# Entity Centric Neural Models for Natural Language Processing

Nicola De Cao

# Entity Centric Neural Models for Natural Language Processing

ILLC Dissertation Series DS-2024-03

Institute for Logic, Language and Computation

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl

The investigations were supported by the Netherlands Organization for Scientific Research (NWO), SAP Innovation Center Network and Facebook AI Research.



Printed and bound by City Printing Ltd (UK).

Entity Centric Neural Models for Natural Language Processing

# ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op dinsdag 14 mei 2024, te 16.00 uur

door Nicola De Cao
geboren te Schio

***Promotiecommissie***

| | | |
|---|---|---|
| *Promotores:* | dr. I.A. Titov | Universiteit van Amsterdam |
| | dr. W. Ferreira Aziz | Universiteit van Amsterdam |
| *Overige leden:* | prof. dr. T. Hofmann | ETH Zürich |
| | prof. dr. L. Zettlemoyer | University of Washington |
| | dr. J. Berant | Universitat Tel Aviv |
| | prof. dr. M. de Rijke | Universiteit van Amsterdam |
| | prof. dr. R. Fernández Rovira | Universiteit van Amsterdam |
| | dr. E.V. Shutova | Universiteit van Amsterdam |

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

# Contents

# List of Tables

# List of Figures

# List of Publications

This thesis is based, for the most part, on the following publications:

- (De Cao et al., 2019b) **De Cao, Nicola**, Aziz, Wilker, & Titov, Ivan. (2019). Question answering by reasoning across documents with graph convolutional networks. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2306–231. https://doi.org/10.18653/v1/N19-1240

- (De Cao et al., 2020) **De Cao Nicola**, Schlichtkrull Michael Sejr, Aziz Wilker, & Titov Ivan. (2020). How do decisions emerge across layers in neural models? interpretation with differentiable masking. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 3243–3255. https://doi.org/10.18653/v1/2020.emnlp-main.262

- (De Cao et al., 2021a) **De Cao Nicola**, Izacard Gautier, Riedel Sebastian, & Petroni Fabio. (2021a). Autoregressive entity retrieval. *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

- (De Cao et al., 2021b) **De Cao Nicola**, Aziz Wilker, & Titov Ivan. (2021b). Editing factual knowledge in language models. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6491–6506. https://doi.org/10.18653/v1/2021.emnlp-main.522

- (De Cao et al., 2021c) **De Cao Nicola**, Aziz Wilker, & Titov Ivan. (2021c). Highly parallel autoregressive entity linking with discriminative correction. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 7662–7669. https://doi.org/10.18653/v1/2021.emnlp-main.604

- (De Cao et al., 2022) **De Cao Nicola**, Wu Ledell, Popat Kashyap, Artetxe Mikel, Goyal Naman, Plekhanov Mikhail, Zettlemoyer Luke, Cancedda Nicola, Riedel Sebastian, & Petroni Fabio. (2022). Multilingual Autoregressive Entity Linking. *Transactions of the Association for Computational Linguistics*, 10, 274–290. https://doi.org/10.1162/tacl_a_00460

During my doctoral studies I also authored, collaborated, and/or published the following articles that are not part of this thesis:

- (De Cao et al., 2019a) **De Cao Nicola**, Aziz Wilker, & Titov Ivan. (2019a). Block neural autoregressive flow. In Amir Globerson & Ricardo Silva (Eds.), *Proceedings of the thirtyfifth conference on uncertainty in artificial intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019* (pp. 1263–1273). AUAI Press.

- (De Cao & Aziz, 2020) **De Cao Nicola**, & Aziz Wilker. (2020). The power spherical distribution. *Proceedings of the 37th International Conference on Machine Learning, ICML 2020 (INNF+ Workshop), 13-18 July 2020, Virtual Event*, 119.

- (Izacard et al., 2020) Izacard Gautier, Petroni Fabio, Hosseini Lucas, **De Cao Nicola**, Riedel Sebastian, & Grave Edouard. (2020). A memory efficient baseline for open domain question answering. *CoRR, abs/2012.15156*.

- (Schlichtkrull et al., 2021) Schlichtkrull Michael Sejr, **De Cao Nicola**, & Titov Ivan. (2021). Interpreting graph neural networks for NLP with differentiable edge masking. *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

- (Petroni et al., 2021) Petroni Fabio, Piktus Aleksandra, Fan Angela, Lewis Patrick, Yazdani Majid, **De Cao Nicola**, Thorne James, Jernite Yacine, Karpukhin Vladimir, Maillard Jean, Plachouras Vassilis, Rocktäschel Tim, & Riedel Sebastian. (2021). KILT: A benchmark for knowledge intensive language tasks. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2523–2544. https://doi.org/10.18653/v1/2021.naacl-main.200

- (Min et al., 2021) Min Sewon, Boyd-Graber Jordan, Alberti Chris, Chen Danqi, Choi Eunsol, Collins Michael, Guu Kelvin, Hajishirzi Hannaneh, Lee Kenton, Palomaki Jennimaria, Raffel Colin, Roberts Adam, Kwiatkowski Tom, Lewis Patrick, Wu Yuxiang, Küttler Heinrich, Liu Linqing, Minervini Pasquale, Stenetorp Pontus, Riedel Sebastian, Yang Sohee, Seo Minjoon, Izacard Gautier, Petroni Fabio, Hosseini Lucas, **De Cao Nicola**, Grave Edouard, Yamada Ikuya, Shimaoka Sonse, Suzuki Masatoshi, Miyawaki Shumpei, Sato Shun, Takahashi Ryo, Suzuki Jun, Fajcik Martin, Docekal Martin, Ondrej

Karel, Smrz Pavel, Cheng Hao, Shen Yelong, Liu Xiaodong, He Pengcheng, Chen Weizhu, Gao Jianfeng, Oguz Barlas, Chen Xilun, Karpukhin Vladimir, Peshterliev Stan, Okhonko Dmytro, Schlichtkrull Michael, Gupta Sonal, Mehdad Yashar, and Yih Wen-tau. 2021. Neurips 2020 efficientqa competition: Systems, analyses and lessons learned. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track, volume 133 of Proceedings of Machine Learning Research*, pages 86–111. PMLR.

- (De Cao et al., 2021d) **De Cao Nicola**, Schmid Leon, Hupkes Dieuwke, & Titov Ivan. (2021d). Sparse interventions in language models with differentiable masking. *CoRR, abs/2112.06837*.

- (Josifoski et al., 2022) Josifoski Martin, **De Cao Nicola**, Peyrard Maxime, Petroni Fabio, & West Robert. (2022). GenIE: Generative information extraction. Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 4626–464

# Acknowledgments

This dissertation would not have been possible without the support and guidance of many individuals. I would like to express my deepest gratitude to:

Most importantly my wife, **Luisa Quarta**: Thank you for your unwavering love and support throughout this journey. You bravely decided to accompany me into the unknown, moving to the UK for my studies. Your unwavering support, even during stressful times, has been my rock and my source of strength.

My supervisors, **Ivan Titov** and **Wilker Aziz**: Your fantastic supervision and mentorship have been invaluable throughout my PhD journey. I am incredibly grateful for your unwavering support, insightful guidance, constant inspiration, and the numerous opportunities you provided me over the years.

**Alex Klementiev**: Thank you for hosting me at Amazon Research in Berlin. This first experience as a researcher outside of academia taught me incredibly valuable lessons that have shaped my approach to research.

**Fabio Petroni** and **Sebastian Riedel**: My internship with you at FAIR London was truly a blast. Your guidance was instrumental in shaping this thesis, and much of it would not have been possible without your expertise and support.

My dear friend, **Michael Schlichtkrull**: You have been a phenomenal collaborator, mentor, and most importantly, a true friend. Your unwavering support and insightful discussions have greatly enriched this research and my overall PhD experience.

I would also like to extend my gratitude to all the colleagues and many friends I made along the way: Caio Corro, Nelly Papalampidi, Christos Baziotis, Elena Voita, Karin Sevegnani, Ivana Balažević, Govert Verkes, Tim Davidson, Bryan Eikema, Patrick Lewis, Douwe Kiela, Nils Reimers and many others.

Finally, I want to thank my parents and brother for their love, encouragement, and understanding throughout this process.

# Chapter 1

# Introduction

Entities[1] are at the center of how we represent and aggregate knowledge. For instance, encyclopedias such as WIKIPEDIA[2] are organized by entities (*e.g.,* one per WIKIPEDIA article). Written encyclopedias have existed for around two thousand years (*e.g.,* the *Naturalis Historia* dates back to 77 AD) and have evolved substantially during such time in form, language, style, and many other aspects. The *Encyclopédie, ou dictionnaire raisonné des sciences, des arts et des métiers* (published in France between 1751 and 1772; Diderot & d'Alembert, 1751) and the *Encyclopædia Britannica* (published in Scotland between 1768 and 1771; Smellie, 1768) are generally considered the first printed Encyclopedias in modern history and defined a radical change in the spread of information in the world. While the content and language may differ, the fundamental organization of information through entities, categories, and cross-references has remained virtually unchanged for centuries. Organizing world knowledge in a such way feels natural and convenient for humans but what about machines? Do machine learning algorithms take advantage of our categorization? Can we build computer algorithms able to connect different pieces of knowledge or to distinguish between ambiguous concepts? Although there are no clear answers to these complex questions yet, in this thesis, we will argue that providing extra information about the nature of entities to Natural Language Processing (NLP) algorithms improves their performance in many useful applications.

Let's start our journey by looking at an example of making use of entities for multi-document question answering. In such a setting, a user asks a question to an information system that needs to search through a library of documents for

[1] entity (noun): something that exists apart from other things, having its own independent existence (https://dictionary.cambridge.org/dictionary/english/entity)

[2] https://www.wikipedia.org

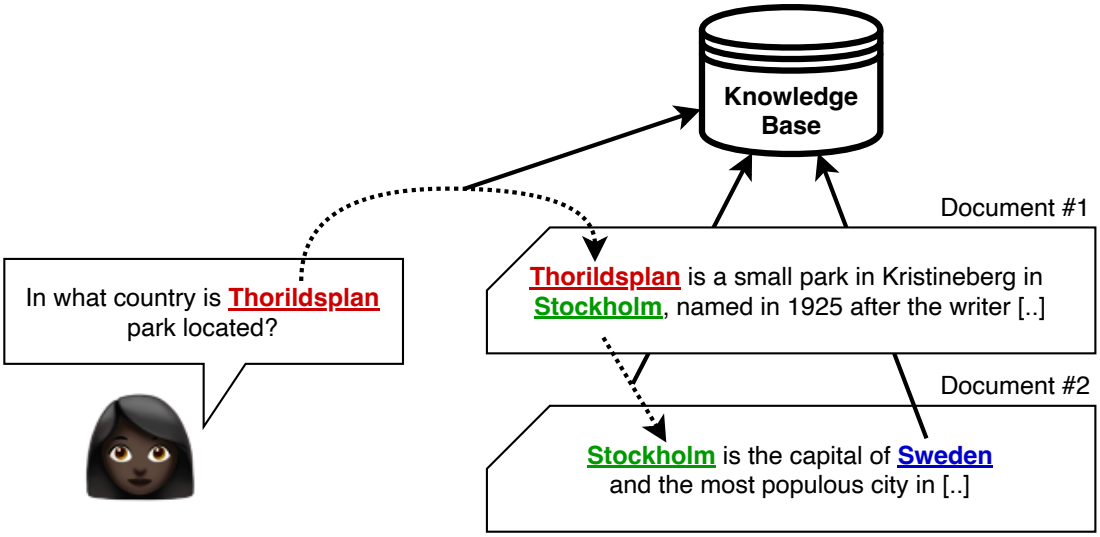



Figure 1.1: Make use of entities for multi-document question answering. A user asks the question "*In what country is Thorildsplan park located?*" **(left)**. The mention of the park is then tagged first and then linked to a specific entity in the Knowledge Base **(top)**. Then, a retrieval system gets one document when another mention of the park is present **(right)** as it is relevant to answer the question. In such a document, another mention of an entity is present ("*Stockholm*") which leads to retrieving another document with the final answer (*i.e.,* "*Sweden*"). The whole process can be split into different tasks and trained for specific parts of the pipeline such as mention detection, entity disambiguation, document retrieval, and reading comprehension.

an answer. In addition, let's assume that there is the need to analyze and reason across multiple documents because no simple answer can be found in a single document. We show a machine-aided process to get an answer in such a setting in figure 1.1. As we can see from there, the process that a machine undertakes aims to mimic what humans would do, and it seems a reasonable strategy. We hypothesize that breaking down such a complex task into learnable sub-steps leads to an overall system improvement and human interpretability of such. We can use objective metrics to see if that is the case on such a task.

**Entities for Natural Language Understanding** To investigate the aforementioned problems, in chapter 3, we investigate how can we exploit entities to tackle Natural Language Understanding (NLU). We introduce a neural model that "reasons"[3] relying on information spread within and across multiple docu-

[3] Here we do not mean we integrate any formal reasoning technique. By "reasoning" here we mean we design the input of the model and its components in such a way as to promote the learning of a multi-step process to generate the output.

ments. Our hypothesis is that making the model output a prediction using loosely "reasoning" (learned) steps through references of entities will allow it to learn a sensible and more generalizable strategy to deal with complex questions. Mentions of entities present in the text are annotated which makes testing our hypothesis much easier. Then, we frame the task as an inference problem on a graph. These mentions are nodes of a graph, while edges encode relations between different mentions (*e.g.,* within- and cross-document co-reference). Graph convolutional networks (GCNs) are applied to these graphs and trained to perform multi-step reasoning. We show that using extra entity information results in a scalable and compact method achieving state-of-the-art results at the time of development (*i.e.,* 2018) on WikiHop, a multi-document question-answering dataset popular back then.

The findings of chapter 3 open the door to more interesting problems since one limiting factor of our contributions is that all mentions of entities are given as inputs to the model. The ability to retrieve mentions of entities in texts is fundamental for knowledge-intensive tasks such as open-domain question answering, and dialog. Thus a natural question arises: how can we exploit language models to identify and disambiguate entities in the text?

**Finding Entities in Text with Language Models** Entity Linking (EL; Bunescu & Paşca, 2006; Cucerzan, 2007; Dredze et al., 2010; Hoffart et al., 2011; Le & Titov, 2018) is a fundamental task in NLP employed as a building block for text understanding (Févry et al., 2020b; Verga et al., 2020). It consists of grounding entity mentions in unstructured texts to Knowledge Base (KB) identifiers (*e.g.,* WIKIPEDIA articles). Entity linking has plenty of applications in multiple domains, spanning open-domain question answering (De Cao et al., 2019b; Nie et al., 2019; Asai et al., 2020), dialogue (Bordes et al., 2017; Wen et al., 2017; Williams et al., 2017; Chen et al., 2017b; Curry et al., 2018; Sevegnani et al., 2021), biomedical systems (Leaman & Gonzalez, 2008; Zheng et al., 2015), information extraction (Sarawagi, 2008; Martinez-Rodriguez et al., 2020), to name just a few. In figure 1.2 we show an example of linking mentions to their relevant entities in a Knowledge Base.

Although there has been extensive previous work on entity retrieval (*e.g.,* Hoffart et al., 2011; Piccinno & Ferragina, 2014; Huang et al., 2015; Le & Titov, 2018; Logeswaran et al., 2019; Broscheit, 2019; Wu et al., 2020, to name just a few) there is a common design choice to most current solutions: entities are associated with a unique atomic label and the retrieval problem can be interpreted as multi-class classification across these labels. The match between input and label is calculated through a bi-encoder (Wu et al., 2020; Karpukhin et al., 2020): a dot product between dense vector encodings of the input and the encoding of an entity's information (such as title and description). Critically, this formulation enables sub-linear search using modern maximum-inner-product-search libraries

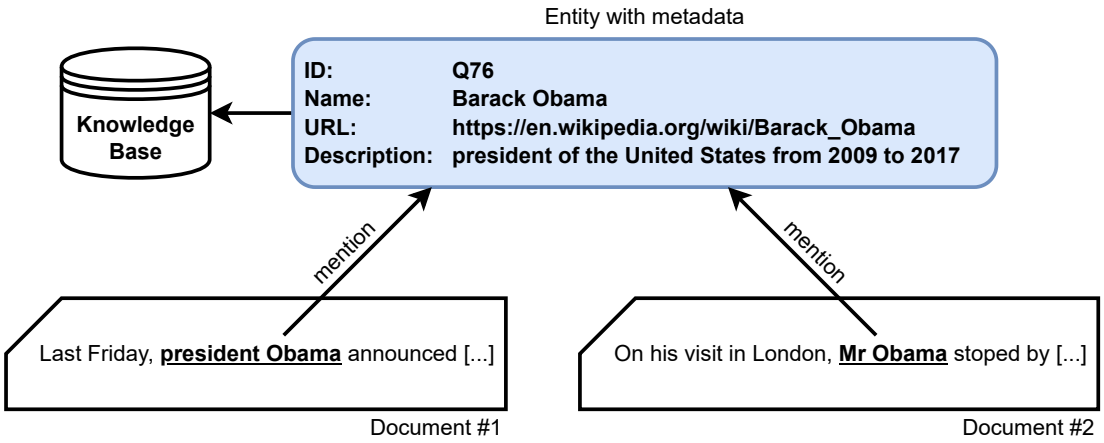



Figure 1.2: An example of two different mentions referring to the same entity. The entity is defined within a Knowledge Base and it has some metadata attached to it such as a name, an URL, and a description.

(Johnson et al., 2019) and hence supports retrieving from large entity databases. In chapter 4, we then propose a novel approach: the first system that retrieves entities by generating their unique names, left to right, token-by-token in an autoregressive fashion. Our model mitigates the limitations of well-established contemporary models[4] that potentially miss fine-grained interactions between text and entities in a Knowledge Base. Additionally, we significantly reduced the memory footprint of current systems (up to 15 times) because the parameters of our encoder-decoder architecture scale with vocabulary size, not with the entity count. We also extend our approach to a large multilingual setting with more than 100 languages (chapter 5). In this setting, we match against entity names of as many languages as possible, which allows exploiting language connections between source input and target name. Finally, we also propose a very efficient approach that parallelizes autoregressive linking across all potential mentions in a text fragment. Such a system relies on a shallow and efficient decoder which allows a >70 faster model with no performance drop (chapter 6).

**Interpretability and Controllability of Language Model** The findings of chapters 4, 5, and 6 open the door to many interesting applications in many sub-domains. One compelling aspect of our study is that it suggests that most of the system gains come from the ability of the model to recall its memories about entity names both obtained during language modeling pre-training and our task-specific fine-tuning. Unfortunately, such ability comes at a price. This is because most (if not all) deep learning-based language models come as black-box functions. Thus, we cannot fully understand their prediction or tell if they reason

[4] Bi-encoders that produce scores via a dot-product (explained in details in section 4.2.2).

or memorize. When they memorize, we also cannot usually control where and how to add, remove, or modify such memories with ease. These reflections lead to our next research question: How can we interpret and control a model's internal knowledge about entities?

To that end, in chapter 7, we introduce a novel post-hoc interpretation technique for inspecting how decisions emerge across layers in neural models. Our system learns to mask out subsets of vectors while maintaining differentiability. This lets us not only plot attribution heatmaps but also analyze how decisions are formed across network layers. We use this system to study BERT models (Devlin et al., 2019a) on sentiment classification and question answering additionally showing that this technique can be applied to the graph-based model presented in chapter 3. Finally, we also propose a method that can be used to edit factual knowledge about entities and, thus, to fix 'bugs' or unexpected predictions without the need for expensive re-training or fine-tuning (chapter 8).

## 1.1 Contributions

The primary contributions of this thesis can be summarised as follows:

1. We introduce a neural model that integrates reasons relying on information spread within and across multiple documents. We frame it as an inference problem on a graph. Mentions of entities are nodes of this graph, while edges encode relations between different mentions.

2. We propose a system that identifies entities from text and links them to an external knowledge base by generating their unique names in an autoregressive fashion in more than 100 languages. We employ constrained generation to use such a generative autoregressive model as a classifier.

3. We present a novel post-hoc interpretation technique for inspecting how decisions emerge across layers in neural models.

4. We develop a method that edits factual knowledge about entities inside language models and, thus, fixes 'bugs' or unexpected predictions without the need for expensive re-training or fine-tuning.

Most if not all the findings suggest a central role of entities in Natural Language Processing and we encourage research towards incorporating entity information across more tasks.

# Chapter 2

# Background

In the following sections, we introduce the background material necessary to understand the work presented in the subsequent chapters. In section 2.1 we introduce a notion of what an entity is while in section 2.2 we define what Knowledge Bases (KB) and Knowledge Graphs (KG) are.

All models used in this thesis are (deep and artificial) Neural Networks (NNs; Hopfield, 1982) primarily dealing with graph or text data. While we do not present an extensive discussion about deep learning and neural network models, we point the reader to Goodfellow et al. (2016) for a thorough introduction to these concepts. In section 2.3 we discuss graph modeling and give an introduction to a class of neural models designed to encode and process graph-structured data known as Graph Neural Networks (GNNs). Then, in section 2.4 we discuss text modeling and the neural network models typically used to process such textual data (*i.e.,* Recurrent Neural Networks (RNNs) and Transformer Networks). Unless stated otherwise, throughout this thesis, we follow the notation and definitions of tables 2.1 and 2.2 for all mathematical equations.

## 2.1 Entities

The term *entity* has several different connotations within the computer science or machine learning community and thus, it has no unique formal definition. Usually, ontologies serve to create a formal representation of the entities within an information system which are based on a taxonomy defined by domain experts. An ontology formally describes categories, properties, and relations between entities. The Web Ontology Language (OWL; Bechhofer et al., 2004; Antoniou & Harmelen, 2004; McGuinness, Van Harmelen, et al., 2004) is an example of a widely adopted

| Description | Notation |
|---|---|
| Function, number or element in a set | $x$ |
| Vector | $\boldsymbol{x}$ |
| Element of a vector | $\boldsymbol{x}_i$ |
| Matrix | $\boldsymbol{X}$ |
| Row of a matrix | $\boldsymbol{X}_i$ |
| Column of a matrix | $\boldsymbol{X}_{:j}$ |
| Element of a matrix | $\boldsymbol{X}_{ij}$ |
| Tensor | $\mathbf{X}$ |
| Element of a tensor | $\mathbf{X}_{i,\ldots,jk}$ |
| Set | $\mathcal{X}$ |
| Element of a set | $x_i$ |
| Element of a set for vectors, matrices and tensors | $\boldsymbol{x}^{(i)}, \boldsymbol{X}^{(i)}, \mathbf{X}^{(i)}$ |
| Standard set of numbers | $\mathbb{R}, \mathbb{N}, \mathbb{Z}$ |
| Set of indices *i.e.,* $\{x \in \mathbb{N}_{>0} : x \leq n\}$ | $[\![1, n]\!]$ |
| Random variable | $\mathrm{x}$ |
| Random vector | $\mathbf{x}$ |
| Random matrix | $\mathbf{X}$ |
| Expected value under distribution $p$ | $\mathbb{E}_p[\ldots]$ |
| Probability mass function of $x$ conditioned on $y$ and parameterized by $\theta$ | $p(x\|y;\theta)$ |

Table 2.1: Notation. Note that vectors and matrices are row-indexed (*i.e.,* $\boldsymbol{X}_i$ is the $i$-th row, $\boldsymbol{X}_{ij}$ is the $j$-th element of the $i$-th row, $\boldsymbol{a}\boldsymbol{b}^\top$ is the inner product, and $\boldsymbol{x}\boldsymbol{A}$ is a vector matrix product).

technology used to define ontologies, that is supported by the World Wide Web Consortium[1].

In this thesis, we focus specifically on entities within a general real-world knowledge base (see next session for more discussion) that describes a particular state of affairs of the world. An example of a knowledge base is WIKIDATA[2] which represents any kind of topic, concept, or object. Entities there are real-world "things" or real-world objects distinguishable from one another. Whether or not a thing should be included in WIKIDATA depends on a notability criteria[3] which implicitly states what an entity is within WIKIDATA.

For example, a *car*[4] is an entity representing an abstract class of vehicles. An entity can have attributes and have relations with one another which gives us

[1] https://www.w3.org
[2] https://www.wikidata.org
[3] https://www.wikidata.org/wiki/Wikidata:Notability
[4] WIKIDATA entity ID Q1420

| Name | Definition |
|---|---|
| Sigmoid function | $\sigma : \mathbb{R} \to (0, 1)$ and $\sigma(x) = (1 + \exp(-x))^{-1}$ |
| Softmax function | $\text{softmax} : \mathbb{R}^d \to \Delta^{d-1}$ and $\text{softmax}(\boldsymbol{x})_i = \frac{\exp(\boldsymbol{x}_i)}{\sum_{j=1}^{d} \exp(\boldsymbol{x}_j)}$ |
| Hadamard product | $\odot : \mathbb{R}^{d \times k} \times \mathbb{R}^{d \times k} \to \mathbb{R}^{d \times k}$ and $(\boldsymbol{A} \odot \boldsymbol{B})_{ij} = \boldsymbol{A}_{ij} \cdot \boldsymbol{B}_{ij}$ |
| Outer product | $\otimes : \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^{d \times k}$ and $(\boldsymbol{a} \otimes \boldsymbol{b})_{ij} = \boldsymbol{a}_i \cdot \boldsymbol{b}_j$ |
| Indicator function | $\mathbf{1}_{\mathcal{A}} : \mathcal{B} \to \{0, 1\}$ and $\mathbf{1}_{\mathcal{A}}(x) = \begin{cases} 1 & \text{if} \quad x \in \mathcal{A} \\ 0 & \text{if} \quad x \notin \mathcal{A} \end{cases}$ |
| Kullback–Leibler divergence | $D_{KL}[p\|q] = \int_{\mathcal{X}} p(\mathrm{d}x) \log \frac{p(\mathrm{d}x)}{q(\mathrm{d}x)}$ |
| Jensen–Shannon divergence | $D_{JS}[p\|q] = \left(D_{KL}[p\|m] + D_{KL}[q\|m]\right)/2$ <br> where $\quad m(x) = (p(x) + q(x))/2$ |

Table 2.2: Definitions. Note that vectors and matrices are row-indexed (*i.e.,* $\boldsymbol{X}_i$ is the $i$-th row, $\boldsymbol{X}_{ij}$ is the $j$-th element of the $i$-th row, $\boldsymbol{a}\boldsymbol{b}^\top$ is the inner product, and $\boldsymbol{x}\boldsymbol{A}$ is a vector matrix product).

information about some of their characteristic. For example, a particular car model like a *Tesla Model S*[5] (an electric sedan produced and sold by Tesla Motors) is also an entity which relates to the *car* entity with the relation *being an instance of*[6]. Entities can also be people, organizations, events or locations such as *Mother's day* and *the United Kingdom*. Numerical proprieties are usually not entities. For example, the capital of the United Kingdom is *London* which is another entity but its population is around 67 million (as of 2021[7]) which is typically denoted as an *attribute*.

## 2.2 Knowledge Bases and Knowledge Graphs

A *knowledge base* is a term that denotes a centralized repository of structured or unstructured data within a computer system. KBs can contain data of multiple modalities, *e.g.,* text, table, graphs, images, audio/video, and others. One of the original uses of the term knowledge base was to describe a system that stores facts about the world which is separate from a reasoning engine that uses those facts to make deductions and produce complex outputs (Hayes-Roth, 1983). The term was originally made up to distinguish between databases (Date,

[5] Wikidata entity ID Q1463050
[6] Wikidata property ID P31
[7] https://data.worldbank.org/indicator/SP.POP.TOTL

1975), which decades ago were just flat, tabular data, and a new way of storing information in a hierarchical and relational form. Nowadays, databases support multi-modal data, semantic meta-data, and hierarchical and relational information thus the terminology is practically equivalent. Knowledge bases are machine-readable resources optimized for information collection, organization, and retrieval. Examples of large and open-source knowledge bases with a mixture of structured and unstructured data are WIKIPEDIA[8], WIKIDATA and the WIKIMEDIA family[9].

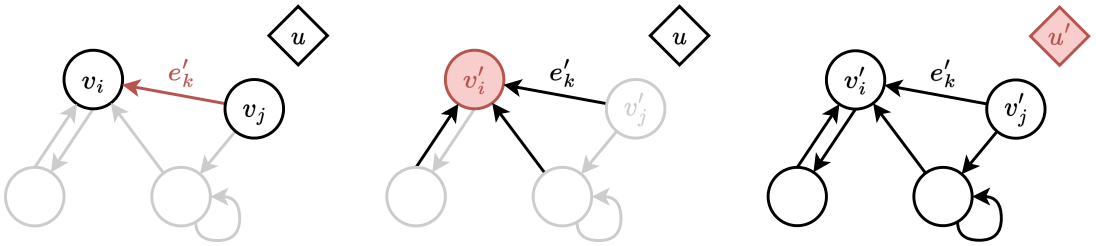
Although some practitioners use the terms knowledge base and *knowledge graph* interchangeably there is a clear distinction between the two. Knowledge graphs are a subset of knowledge bases, *i.e.,* all knowledge graphs are knowledge bases, while not every knowledge base is a knowledge graph. The key difference between the two is that within a knowledge graph, information is represented as a graph characterized by relationships between entities. Knowledge graphs do indeed represent graph-structured data or information topologically equivalent to a graph. They often store interconnected descriptions of entities or general abstract concepts. Examples of large and open-source knowledge graphs are DBpedia[10] and WIKIDATA. Knowledge graphs have also been largely employed by the industry as well (Noy et al., 2019). Search engines such as Google[11], Bing[12] employ algorithms that operated on knowledge graphs. Question-answering services such as WolframAlpha[13], Apple's Siri (Ilyas et al., 2022), and Amazon Alexa[14] extensively rely on this technology as well.

Although we explain their differences, throughout this thesis the terms knowledge base and knowledge graph are essentially interchangeable as we do not rely upon any propriety that is not present in both.

## 2.3 Neural Models for Graphs

Graph Neural Networks (GNNs) are a class of neural network models developed for the processing of graph-structured data (Gori et al., 2005; Scarselli et al., 2009). Generally, each layer of a GNN takes as input a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathcal{R} \rangle$ (*i.e.,* a triple consisting of a set of vertices, a set of edges, and a set of relations) alongside with a set of initial vertex features $\mathcal{X}_v \in \mathcal{V} \times \mathbb{R}^{d_v}$, edge features $\mathcal{X}_e \in \mathcal{E} \times \mathbb{R}^{d_e}$, and global features $\boldsymbol{x}^g \in \mathbb{R}^{d_g}$ (for some dimensionalities $d_v, d_e, d_g \in \mathbb{N}_{>0}$). Then, a layer computes new node, edge, and global embeddings as new hidden states according to a function that takes into account the graph structure (*e.g.,* the neighborhood

[8] https://www.wikipedia.org
[9] https://www.wikimedia.org
[10] https://www.dbpedia.org
[11] https://developers.google.com/knowledge-graph
[12] https://www.microsoft.com/en-us/bing/apis/bing-entity-search-api
[13] https://www.wolfram.com/knowledgebase
[14] https://aws.amazon.com/alexaforbusiness/knowledge-skills




(a) Edge update. (b) Node update. (c) Global update.

Figure 2.1: Updates in a GNN block. $v_i, v_j$, and $v'_i, v'_j$ are two vertices features before and after an update respectively. $e_k$ and $e'_k$ are edge features before and after an update. $u$ and $u'$ are global graph features before and after an update. Red indicates the element that is being updated, and black indicates other elements which are involved in the update (note that the pre-update value of the red element is also used in the update). Figure and caption adapted and inspired from figure 3 in Battaglia et al. (2018).

surrounding each vertex and the relations which connect nodes). GNNs layers can be stacked to create a fully differentiable neural model learnable with standard stochastic gradient descent. They are useful for providing topology-aware node, edge, and graph features for downstream prediction tasks such as classification or regression at node-, edge- and graph-level. The way GNNs aggregate neighborhood information depends on many architectural choices that currently count hundreds of variations. In addition, some variations of GNNs were specifically developed for dealing with simple graphs, multigraphs, directed and undirected graphs, temporal graphs, and hypergraphs (Schlichtkrull et al., 2018; Feng et al., 2019; Yu et al., 2019; Rossi et al., 2020).

The only GNN model used in this thesis is a modification of Relational Graph Convolutional Network (R-GCN; Schlichtkrull et al., 2018) that belongs to a particular class called *graph neural message passing* (Gilmer et al., 2017). In a nutshell, each message passing GNN layers acts in 3 stages (see figure 2.1 for a visual overview of these steps):

1. **edge update:** for each edge, it computes its new attributes taking into account its previous set of features, the global state, and all the node features of the nodes which are connected to the edge (as shown in figure 2.1a);

2. **node update:** for each node, its new attributes taking into account its previous set of features, the global state, and all the edge features of the edges which are connected to the node (as shown in figure 2.1b);

3. **global update:** it aggregates all edge and node attributes globally and then it computes an updated global attribute (as shown in figure 2.1c).

Such formulation is quite general, and each step of the way can be implemented differently depending on the application requirements. Finally, we point the reader to a survey by Battaglia et al. (2018) for a further discussion of GNNs and message passing.

## 2.4 Neural Models for Text

Neural networks for text modeling are a class of neural network models developed for processing natural language (Winograd, 1971; Goldberg, 2016). Generally, they take some text in form or an ordered set (sequence) of discrete tokens $\mathcal{X} \in \mathcal{V}^n$ of size $n$ as inputs. Note that the number of input tokens does not need to be fixed, *e.g.,* a network can operate on sequences of different lengths. Each element in the input sequence belongs to a pre-defined vocabulary $\mathcal{V}$ (*e.g.,* the set of ASCII characters, all Unicode symbols, or a subset of all English words). Although the way a neural model for text operates differs by a wide range, nowadays, they typically follow this general outline:

1. **encode step:** they encode the discrete input sequence $\mathcal{X}$ into a sequence of continuous embedded tokens $\mathcal{H}^{(0)} \in \mathcal{X} \times \mathbb{R}^{d_e}$ (*i.e.,* the 0-th layer hidden state for some $d_e$-dimensional embeddings) using an *input embedding matrix* $\boldsymbol{E}^i \in \mathbb{R}^{|\mathcal{V}| \times d_e}$ which allows the following neural layers to operate in the real coordinate space;

2. **hidden steps:** they process the embedded sequence either via a Recurrent Neural Network, Convolutional Neural Network, or a Transformer model (*i.e.,* ad hoc architectures built for processing sequences—see the next few sections for more details);

3. **output step:** they produce a sequence of $d_o$-dimensional hidden representation $\mathcal{Y} \in \mathcal{X} \times \mathbb{R}^{d_o}$, one for each input token;

4. **post-processing step:** eventually, they either output i) the sequence of hidden states $\mathcal{Y}$, or i) a single vector for the whole sequence (*e.g.,* via an average or max pooling of $\mathcal{Y}$), or iii) token probabilities derived by a projection of $\mathcal{Y}$ (typically used in generative tasks like translation or summarization).

When a model is required to output token probabilities, it also has an *output embedding matrix* $\boldsymbol{E}^o \in \mathbb{R}^{d_o \times |\mathcal{V}|}$ (often $\boldsymbol{E}^o = \boldsymbol{E}^i$), which is used to project $d_o$-dimensional hidden representation predicted by the network to logits[15] of a categorical probability distribution over the token vocabulary $\mathcal{V}$.

[15]Logits parameters (also know as scores) of categorical probability distribution. The distribution over $n$ categories is derived from logits $\theta \in \mathbb{R}^n$ and temperature $\tau \in \mathbb{R}_{>0}$ typically set to $\tau = 1$ as: $p(x_i; \theta, \tau) = \text{softmax}(\theta/\tau)_i = \frac{\exp(\theta_i/\tau)}{\sum_{j=1}^{n} \exp(\theta_j/\tau)}$.

### 2.4.1 Recurrent Neural Networks

A standard approach for text modeling using neural networks is to use Recurrent Neural Networks (RNNs; Jordan, 1986; Rumelhart & McClelland, 1987; Elman, 1990). RNNs apply a form of feedforward neural network recursively for each element in a sequence of inputs while keeping an internal state (memory). For a sequence of $n$ elements, they require to be applied $n$ times. At the end of $n$ calls, the RNN will have an internal representation of the whole sequence. RNN layers can also be stacked to create powerful and fully differentiable neural models learnable with standard stochastic gradient descent when provided with some supervision from which once can derive a loss or objective function. They are useful for processing variable-length sequences of text or audio signals. Long Short Term Memory networks (LSTM; Hochreiter & Schmidhuber, 1997) are a well-established improvement of RNNs designed to prevent their vanishing gradient issues. In recent years, LSTMs have been augmented with *attention* (see next section) and eventually overtaken for most of the tasks by another class of models called transformers neural networks (Vaswani et al., 2017).

### 2.4.2 Attention

The attention mechanisms in neural networks is considered to be an effort to mimic human brain actions in a simplified manner that took inspiration from cognitive attention (James et al., 1890). The core idea behind attention is using a learned weighted average to summarize sequences of inputs (typically vectors, but it can be used with arbitrary multi-dimensional tensors). The application of attention increases the importance of some parts of the input while decreasing others. Typically attention is used as a layer inside a deep neural network trained by gradient descent. Thus, learning which part of the input is more important than another depends entirely on the data, the rest of the architecture, and the loss function. Modern use of attention for natural language processing is credited to Bahdanau et al. (2015) and Luong et al. (2015) who originally proposed to use attention to summarize sequences for neural machine translation. They used it as a component of RNN models allowing the decoder to *attend* (*i.e.,* give attention) to elements in the input sequence differently for each time-step. A simple attention mechanism from $m$ elements over a matrix of $n$ $d_v$-dimensional vectors $\boldsymbol{V} \in \mathbb{R}^{n \times d_v}$

can be summarized by the following equation:

$$\boldsymbol{o}_i = \sum_{j=1}^{n} \boldsymbol{A}_{ij} \boldsymbol{V}_j \quad \forall i \in [\![1, m]\!] \,, \tag{2.1}$$

where each output vector $\boldsymbol{o}_i \in \mathbb{R}^{d_v}$ is a weighted sum of the values $v_{1,\dots,n}$ according to the *attention weights* $\boldsymbol{A}_{ij} \in [0, 1]^{m \times n}$ such that

$$\boldsymbol{A}_{ij} = \text{softmax}(\boldsymbol{S}_i)_j = \frac{\exp(\boldsymbol{S}_{ij})}{\sum_{k=1}^{n} \exp(\boldsymbol{S}_{ik})} \quad \forall i \in [\![1, m]\!], \forall j \times [\![1, n]\!] \,. \tag{2.2}$$

Note that the *scores* (*i.e.,* logits) $\boldsymbol{S}_{ij} \in \mathbb{R}^{m \times n}$ can be computed in an arbitrary way that depends on the implementation and intended use. However, it is important that the attention weights $\boldsymbol{A}_{ij}$ for $j \in [\![1, n]\!]$ sum to 1 (*i.e.,* $\sum_{j=1}^{n} \boldsymbol{A}_{ij} = 1$) since the attention mechanism is a convex rather than a linear combination of vectors (Rockafellar, 1970).

**Scaled Dot-Product Attention** More recently, Vaswani et al. (2017) presented a more complex type of attention known as scaled dot-product attention. The authors used this formulation to introduce the transformer architecture with the aim of replacing the RNN for sequence modeling entirely (see next section for an introduction to transformers). Concretely, the attention function used for the transformer architecture consists of

1. a keys matrix $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$ which contains a set of $n$ key vectors of size $d_k$ used to identify the particular elements in the input set;
2. a values matrix $\boldsymbol{V} \in \mathbb{R}^{n \times d_v}$ which contains a set of $n$ value vectors of size $d_v$ used to store information for each element in the input set;
3. a queries matrix $\boldsymbol{Q} \in \mathbb{R}^{m \times d_k}$ which contains a set of $m$ query vectors of size $d_k$ used to attend the elements in the input set.

Then, each score $\boldsymbol{S}_{ij}$ is computed via a scaled dot-product $\boldsymbol{Q}_i \boldsymbol{K}_j^\top / \sqrt{d_k}$ where $\sqrt{d_k}$ is a normalization factor.[16] In matrix form, scaled dot-product attention can be written as

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \underbrace{\text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)}_{=\boldsymbol{A}\ \text{(attention matrix)}} \boldsymbol{V} \,, \tag{2.3}$$

where the softmax operation is computed in the last matrix dimension. This function gives us an output matrix $\boldsymbol{O} \in \mathbb{R}^{m \times d_v}$.

[16] The dot product $\sum_{i=1}^{d} \boldsymbol{a}_i \boldsymbol{b}_i$ of two independent random $d$-dimensional vectors with mean 0 and variance 1 has has mean 0 and variance $d$. Thus, normalizing corresponds to dividing by the vectors' standard deviation at initialization, assuming normal distributions.

**Multi-head Attention** Vaswani et al. (2017) further extended the limited single attention function with $d_{model}$-dimensional keys, values, and queries (*i.e.,* the hidden dimensionality used a model) concatenating different applications of the attention function. Multi-head attention linearly projects the queries, keys, and values $h$ times with different, learned linear projections to $d_k$, $d_k$, and $d_v$ dimensions, respectively. All of these projected versions of attention perform in parallel. The concatenated output vector still has $d_v$ values. The main intuition behind what is the value of multi-head attention is that it allows a model to jointly attend to the information in the input through different learned attention mixtures and projections. In matrix form, multi-head (with $h$ heads) attention can be written as

$$\begin{aligned} \text{MultiHeadAtt}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Concat}(\boldsymbol{H}^{(1)}, \cdots, \boldsymbol{H}^{(h)})\boldsymbol{W}^{O} \ , \\ \boldsymbol{H}^{(i)} &= \text{Attention}(\boldsymbol{Q}\boldsymbol{W}^{(Q,i)}, \boldsymbol{K}\boldsymbol{W}^{(K,i)}, \boldsymbol{V}\boldsymbol{W}^{(V,i)}) \ , \end{aligned} \tag{2.4}$$

where the projections are parameter matrices

$$\begin{aligned} &\boldsymbol{W}^{(K,i)} \in \mathbb{R}^{d_{model} \times d_k} \quad \boldsymbol{W}^{(Q,i)} \in \mathbb{R}^{d_{model} \times d_k} \\ &\boldsymbol{W}^{(V,i)} \in \mathbb{R}^{d_{model} \times d_v} \quad \boldsymbol{W}^{O} \in \mathbb{R}^{h \cdot d_v \times d_{model}} \end{aligned} \qquad \forall i \in [\![1, h]\!] \ .$$

Multi-head attention is fully differentiable and all weights are typically learned with stochastic gradient descent via the optimization of a loss function.

### 2.4.3 Transformer Neural Networks

The transformer architecture originally proposed by Vaswani et al. (2017) is a rather successful attempt to replace RNNs for dealing with sequential data. The two fundamental differences from RNNs are that 1) the input sequence is processed all at once as opposed to RNNs that build a hidden representation incrementally through time, and 2) the model uses the attention mechanisms to attend the input sequence at each layer incrementally constructing a representation of a sequence through its layers. A simple transformer is essentially a stack of $l$ layers, each implementing the following equation

$$\mathbf{H}^{(\ell+1)} = \text{LayerNorm}(\mathbf{H}^{(\ell)} + \text{Sublayer}(\mathbf{H}^{(\ell)})) \quad \forall \ell \in [\![1, l]\!] \ , \tag{2.5}$$

where $\mathbf{H}^{(\ell)}$ denotes the hidden state tensor at layer $\ell$, and each Sublayer is either a feedforward layer or an attention layer. LayerNorm denotes a Layer Normalization layer (Ba et al., 2016). An attention layer implements equation 2.4 where keys, queries, and values are computed as linear projections of the current hidden state, and a feedforward layer is implemented as

$$\text{FFNN}(\boldsymbol{x}) = \phi(\boldsymbol{x}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)})\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)} \ , \tag{2.6}$$

where $\boldsymbol{W}^{(1)} \in \mathbb{R}^{d_{model} \times d_{ffnn}}$ and $\boldsymbol{W}^{(2)} \in \mathbb{R}^{d_{ffnn} \times d_{model}}$ are weight matrices, $\boldsymbol{b}^{(1)} \in \mathbb{R}^{d_{ffnn}}$ and $\boldsymbol{b}^{(2)} \in \mathbb{R}^{d_{model}}$ are biases terms, and $\phi$ can be any non-linear func-

tion. Vaswani et al. (2017) used Rectified Linear Units (ReLU) activations (*i.e.,* $\text{ReLU}(x) = max(0, x)$).

**Encoder and Decoder models** Transformers can be used as encoders to *encode* (*i.e.,* process) and input sequence as well as decoders to *decode* (*i.e.,* generate) an output sequence. When used as encoders, the transformer's attention is called *self-attention* as the model takes the input sequence to model the attention to itself. When used as decoders, the transformer's attention may be applied twice: one to attend the input sequence (if it was encoded by an encoder model) and another to attend the output sequence generated up to a current time step. The mechanism of attending another set is called *cross-attention*. Transformer decoders only attend past values, *i.e.,* previous steps during generation to generate their outputs. However, during training time, the model can learn from all steps altogether via *masked attention* that prevents the model from attending subsequent positions. Masked attention essentially *masks* subsequent values in the sequence preventing the model from accessing those (concretely, it makes the attention matrix triangular). Masked attention during training allows training Transformers with a single forward pass as opposed to RNNs which need to process every step in a sequence in an iterative fashion.

### 2.4.4 Pre-trained Language Models

**Word embeddings** Using pre-trained models for natural languages processing became popular in the past decade starting with the success of pre-trained word vectors (*e.g.,* word2vec and GloVe; Mikolov et al., 2013; Pennington et al., 2014a). Their success is mainly due to *transfer learning* (Pan & Yang, 2010; Goodfellow et al., 2016): the concept of transferring the previously learned information from other tasks. The objective of transfer learning is to potentially improve the efficacy and sample efficiency while learning new tasks. Pre-trained word embeddings are trained in an self-supervised manner (*i.e.,* they do not require any annotated data but the they are trained with indirect supervision that can be derived from the text corpora itself). Word vectors are typically trained with a language model objective (*i.e.,* a word embedding has to predict its context, or the context embedding has to predict a word). After pre-training, these vectors are used within neural-based sequence models to encode textual inputs providing an initial representation usually called *embedding layer*. Word vectors have been extensively shown to contain semantic information words that are useful for essentially any NLP application.

**Bidirectional LSTMs** Although extremely useful and very efficient to train and use, word embeddings have a major drawback: they store information about single tokens without any context. Indeed, the learning of using the context to

make any prediction is delegated to a sequence model that takes these word vectors as inputs and produces a task-dependent output. However, any sequence model must learn basic sentences' syntax and semantics to accomplish a task. Indeed, this inefficiency led to the development of modern pre-trained language models. Embedding vectors from Language Models (ELMo; Peters et al., 2018) was the first successful work approach that used pre-train word vectors and bidirectional-LSTMs together to produce deep contextualized word representations for downstream application. The key difference from before is that input representations from this model are contextualized, *i.e.,* they already contain information about the sentence and surrounding text. ELMo is a bidirectional language model which optimizes the forward sequence probability of $n$ tokens $\mathcal{T} = \{t_i\}_{i=1}^{n}$:

$$p(\mathcal{T}) = \prod_{i=1}^{n} p(t_i|t_{<i}; \theta_{\overrightarrow{LSTM}}, \theta_w, \theta_s) \ , \tag{2.7}$$

as well as the backward sequence probability

$$p(\mathcal{T}) = \prod_{i=1}^{n} p(t_i|t_{>i}; \theta_{\overleftarrow{LSTM}}, \theta_w, \theta_s) \ , \tag{2.8}$$

where $\theta_{\overrightarrow{LSTM}}, \theta_{\overleftarrow{LSTM}}$ are the parameters of the forward and backward LSTMs respectively, $\theta_w$ are shared learned word vector representation and $\theta_s$ a shared softmax layer (*i.e.,* a linear projection from a hidden state to the output vocabulary dimension for predicting tokens' probabilities). The authors showed consistent improvement in various tasks when using these representations.

**Generative Transformers** Another step forward in this directions was pre-training generative transformers (Transformer-XL, GPT, GPT-2, and GPT-3 models; Dai et al., 2019; Radford et al., 2018a; Radford et al., 2019; Brown et al., 2020). Those models optimize only a forward language objective while relying entirely on the transformer architecture:

$$p(\mathcal{T}) = \prod_{i=1}^{n} p(t_i|t_{<i}; \theta_{transformer}, \theta_w) \ , \tag{2.9}$$

where $\theta_{transformer}$ are the parameter of the transformers and $\theta_w$ the learned word vector representation shared with the softmax layer. The main advantages of transformers over bi-LSTM models were 1) an advantageous inductive bias due to the architecture (see discussion in section 2.4.3), 2) much faster training which allowed training on more data, and 3) the ability to efficiently stack more layers which allows the models to learn more complex information from the data.

**Bidirectional Transformers** Generative transformers can be considered decoder-only models since they are only trained to predict the next word in a sentence. However, there are also pre-training bidirectional transformers (BERT, RoBERTa; Devlin et al., 2019b; Liu et al., 2019) which are denoted as encoder-only models. These encoders should be preferred when one does not need to generate text because unidirectional attention is less expressive than its bidirectional counterpart. Encoder models are typically trained as denoising autoencoders (Vincent et al., 2010; Goodfellow et al., 2016): their training objective is to reconstruct part or all input after it has been corrupted. In particular, these classes of models are trained randomly *masking* (*i.e.,* removing) part of the input and replacing it with a placeholder or a random token from the vocabulary. The *masked* language model objective is then

$$p(\mathcal{T}_{masked}|\mathcal{T}_{unmasked}) = \prod_{t \in \mathcal{T}_{masked}} p(t|\mathcal{T}_{unmasked}; \theta_{transformer}, \theta_w) \ , \tag{2.10}$$

where $\mathcal{T}_{masked}$ is the set of masked tokens and $\mathcal{T}_{unmasked}$ is the set of tokens that it has not been masked ($\mathcal{T}_{masked} \cup \mathcal{T}_{unmasked} = \mathcal{T}$ and $\mathcal{T}_{masked} \cap \mathcal{T}_{unmasked} = \emptyset$). Note that there is no autoregressive decomposition of the likelihood nor modeling of the whole text probability. The probability of masking a token is a hyperparameter. Very different strategies of masking and variation of training have been proposed. We refer the reader to a survey by Rogers et al. (2020) for an in-depth discussion of these.

**Encoder-Decoder Transformers** Another important class of transformers models combines encoder and decoder models. These models use a bidirectional transformer model to encode the input sequence and then a unidirectional (autoregressive) decoder model to generate text. As explained in section 2.4.3 these models use a combination of self-attention and cross-attention. Text-To-Text Transfer Transformer and Denoising Sequence-to-Sequence Pre-training (T5, BART; Raffel et al., 2020; Lewis et al., 2020a) are two popular models that implement the encoder-decoder architecture. These models are typically trained as denoising autoencoders where the input is corrupted (with different strategies), encoded by the encoder model, and trained for an autoregressive generation:

$$p(\mathcal{T}|\mathcal{T}_{corrupted}) = \prod_{i=0}^{n} p(t_i|t_{<i}, \text{enc}(\mathcal{T}_{corrupted}; \theta_{enc}); \theta_{dec}, \theta_w) \ , \tag{2.11}$$

where $T_{corrupted}$ is the set of corrupted tokens, *enc* is the encode function while $\theta_{enc}$ and $\theta_{dec}$ are the encoder and decoder parameters respectively (typically a disjoint set). $T_{corrupted}$ comes as a combination of shuffling and masking and differs by hyperparameter setting and implementation.

**Ongoing Research** The use of transformers models allowed scaling up to orders of magnitude more data while also pre-training with orders of magnitude more parameters. Fine-tuning massively pre-trained consistently brought improvements over training from scratch non-contextualized models (Chowdhery et al., 2022). The huge impact of these models created a new paradigm in NLP that consists of training a few *foundation models*[17] and then fine-tuning them on specific tasks—see Bommasani et al. (2021) for a thorough discussion. Contemporary transformers include thousands of variations, both architectural and in the way they are pre-trained, and we revert the reader to a survey by Rogers et al. (2020) for more discussion. Two significant directions relevant to this thesis are:

1. retrieval augmented language models pre-training (like REALM and RAG; Guu et al., 2020; Lewis et al., 2020b) that employ a latent knowledge retriever, which allows the model to retrieve and attend over documents from a large corpus such as WIKIPEDIA;

2. Enhanced Language Representation with Informative Entities pre-training (ERNIE, ERNIE 2.0, and ERNIE 3.0; Zhang et al., 2019b; Sun et al., 2020a; Sun et al., 2021b) which incorporate and attend entity information from a knowledge base to augment the capabilities of the model—other models that also do that are Févry et al. (2020b) and Verga et al. (2020).

Recently, many variants improved computational and memory efficiency upon the original architecture. We point the reader to Tay et al. (2022a) for a comprehensive overview of existing work and models across multiple domains from the efficiency perspective. Transformer efficiency also enabled the development of extremely large models from a hundred billion to up to half a trillion parameters (Rae et al., 2021; Zhang et al., 2022a; Chowdhery et al., 2022; Smith et al., 2022).

[17] A foundation model is a model extensively trained (both in terms of steps and quantity of data) which results in one that can be adapted to a wide range of downstream tasks (Bommasani et al., 2021).

# Chapter 3

# Question Answering by Reasoning Across Documents with Entity Graph Convolutional Networks

**Chapter Highlights**

Most research in reading comprehension has focused on answering questions based on individual documents or even single paragraphs. In this chapter, we introduce a neural model which integrates and reasons relying on information spread within documents and across multiple documents. We frame it as an inference problem on a graph. *Mentions of entities* are nodes of this graph while edges encode relations between different mentions (*e.g.,* within- and cross-document coreference). A graph convolutional network (GCN; Kipf & Welling, 2017) is applied to these graphs and trained to perform multi-step reasoning. Our Entity-GCN method is scalable and compact, and it achieved state-of-the-art results on a multi-document question answering dataset, WikiHop (Welbl et al., 2018) at the time of writing (2018). Our contributions can be summarized as follows:

- we present a novel approach for multi-hop QA that relies on a (pre-trained) document encoder and information propagation across multiple documents using graph neural networks;
- we provide an efficient training technique which relies on a slower offline and a faster online computation avoiding expensive document processing;
- we empirically show that our algorithm is effective, presenting an improvement over previous results at the time writing (*i.e.,* 2018).

Entities and their mentions are central to the development of Entity-GCN. In this chapter, we focus on using them rather than predicting them (*i.e.,* we do not tackle entity recognition and disambiguation).

## 3.1 Introduction

The long-standing goal of natural language understanding is the development of systems which can acquire knowledge from text collections. Fresh interest in reading comprehension tasks was sparked by the availability of large-scale datasets, such as SQuAD (Rajpurkar et al., 2016a), CNN/Daily Mail (Hermann et al., 2015) and Natural Questions (Kwiatkowski et al., 2019), enabling end-to-end training of neural models (Seo et al., 2017; Xiong et al., 2017; Shen et al., 2017; Karpukhin et al., 2020; Lewis et al., 2020b; Izacard et al., 2020; Asai et al., 2020; Lewis et al., 2021). These systems, given a text and a question, need to answer the query relying on the given document. However, it has been observed that most questions in these datasets do not require reasoning across the document, but they can be answered relying on information contained in a single sentence (Weissenborn et al., 2017). The last generation of large-scale reading comprehension datasets, such as a NarrativeQA (Kočiský et al., 2018), TriviaQA (Joshi et al., 2017), RACE (Lai et al., 2017) and ELI5 (Fan et al., 2019), have been created in such a way as to address this shortcoming and to ensure that systems relying only on local information cannot achieve competitive performance.

Even though these new datasets are challenging and require reasoning within documents, many question answering and search applications require aggregation of information across multiple documents. To fill that research gap, the WikiHop dataset (Welbl et al., 2018) was explicitly created to facilitate the development of systems dealing with these scenarios. Each example in WikiHop consists of a collection of documents, a query and a set of candidate answers (see figure 3.1). Though there is no guarantee that a question cannot be answered by relying just on a single sentence, the authors ensure that it is answerable using a chain of reasoning crossing document boundaries. After we produced the work presented in this chapter, another important multi-hop question answering dataset was published (HotpotQA; Yang et al., 2018b). We do not present experiments or in-depth discussion on that dataset throughout this chapter, but we discuss it in section 3.6 as part of subsequent work (*i.e.,* published after ours).

Though an important practical problem, the multi-hop setting was not receiving much attention in 2018. Indeed, the methods reported by Welbl et al. (2018) approach the task by merely concatenating all documents into a single long text and training a standard RNN-based reading comprehension model, namely, BiDAF (Seo et al., 2017) and FastQA (Weissenborn et al., 2017). Document concatenation in this setting was also used in Weaver (Raison et al., 2018) and MHPGM (Bauer et al., 2018). The only published work which went beyond concatenation was due to Dhingra et al. (2018), where they augment RNNs with jump-links corresponding to co-reference edges. Though these edges provide a structural bias, the RNN states are still tasked with passing the information across the document and performing multi-hop reasoning.

Instead, we frame question answering as an inference problem on a graph

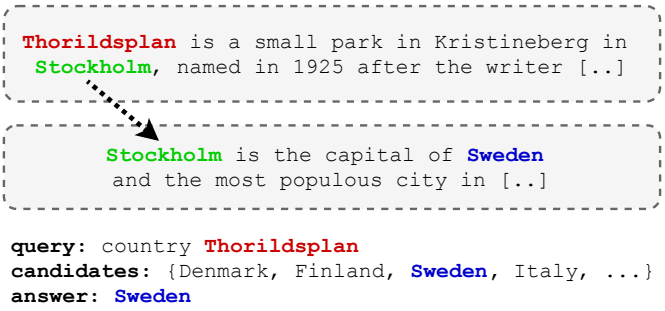



Figure 3.1: A sample from WikiHop where multi-step reasoning and information combination from different documents is necessary to infer the correct answer.

representing the document collection. Nodes in this graph correspond to named entities in a document whereas edges encode relations between them (*e.g.,* cross- and within-document coreference links or simply co-occurrence in a document). We assume that reasoning chains can be captured by propagating local contextual information along edges in this graph using a graph convolutional network (GCN; Kipf & Welling, 2017). We also assume that named entities are already tagged (*i.e.,* we do not rely on any named entity recognition model) and linked to a correct entity (*i.e.,* we do not rely on any entity disambiguation model). This was possible because the dataset we work with (WikiHop) already has this annotation and therefore it was not a required step. Note that in general this annotation is obviously not available and some additional modelling would be required. On that account, in chapters 4, 5, and 6 we explore entity recognition, disambiguation, and linking models filling this gap.

The multi-document setting imposes scalability challenges. In realistic scenarios, a system needs to learn to answer a query for a given collection (*e.g.,* WIKIPEDIA or a domain-specific set of documents). In such scenarios, one cannot afford to run expensive document encoders (*e.g.,* RNN or transformer-like self-attention; Peters et al., 2018; Vaswani et al., 2017), unless the computation can be pre-processed both at train and test time. Even if (similarly to WikiHop creators) one considers a coarse-to-fine approach, where a set of potentially relevant documents is provided, re-encoding them in a query-specific way remains the bottleneck. In contrast to other proposed methods (*e.g.,* Dhingra et al., 2018; Raison et al., 2018; Seo et al., 2017), we avoid training expensive document encoders.

In our approach, only a small query encoder, the GCN layers and a simple feed-forward answer selection component are learned. Instead of training RNN encoders, we use contextualized embeddings (ELMo[1]; Peters et al., 2018) to obtain

[1] The use of ELMo is an implementation choice, and, in principle, any other contextual pre-trained model could be used (*e.g.,* Radford et al., 2018b; Devlin et al., 2019a). At the time of writing, ELMo was the only available option for contextualized embeddings.

initial (local) representations of entity nodes. This implies that only a lightweight computation has to be performed online, both at train and test time, whereas the rest is pre-processed. Even in the somewhat contrived WikiHop setting, where fairly small sets of candidates are provided, the model is at least 5 times faster to train than BiDAF.[2] Interestingly, when we substitute ELMo with simple pre-trained word embeddings, Entity-GCN still performs on par with many techniques that use expensive question-aware recurrent document encoders.

Despite not using recurrent document encoders, the full Entity-GCN model achieves over 2% improvement over the best previously-published results. As our model is efficient, we also reported results of an ensemble which brings further 3.6% of improvement and only 3% below the human performance reported by Welbl et al. (2018).

## 3.2 Background

We first review the dataset we focus on, WikiHop by Welbl et al. (2018), as well as the task abstraction before presenting related work.

### 3.2.1 Data and Task Definitions

**Data** The WikiHop dataset comprises of tuples $\langle q, \mathcal{S}_q, \mathcal{C}_q, a^\star \rangle$ where: $q$ is a query/question, $\mathcal{S}_q$ is a set of supporting documents, $\mathcal{C}_q$ is a set of candidate answers (all of which are entities mentioned in $\mathcal{S}_q$), and $a^\star \in \mathcal{C}_q$ is the entity that correctly answers the question. WikiHop is assembled assuming that there exists a corpus and a knowledge Base (KB) related to each other. The KB contains triples $\langle s, r, o \rangle$ where $s$ is a subject entity, $o$ an object entity, and $r$ a unidirectional relation between them. Welbl et al. (2018) used WIKIPEDIA as corpus and WIKIDATA (Vrandečić, 2012) as KB. The KB is only used for constructing WikiHop: Welbl et al. (2018) retrieved the supporting documents $\mathcal{S}_q$ from the corpus looking at mentions of subject and object entities in the text. Note that the set $\mathcal{S}_q$ (not the KB) is provided to the QA system, and not all of the supporting documents are relevant for the query but some of them act as distractors. Queries, on the other hand, are not expressed in natural language, but instead consist of tuples $\langle s, r, ? \rangle$ where the object entity is unknown and it has to be inferred by reading the support documents. Therefore, answering a query corresponds to finding the entity $a^\star$ that is the object of a tuple in the KB with subject $s$ and relation $r$ among the provided set of candidate answers $\mathcal{C}_q$.

[2] When compared to the 'small' and hence fast BiDAF model reported in Welbl et al. (2018), which is 25% less accurate than our Entity-GCN. Larger RNN models are problematic also because of GPU memory constraints.

**Task** The goal is to learn a model that can identify the correct answer $a^\star$ from the set of supporting documents $\mathcal{S}_q$. To that end, we exploit the available supervision to train a neural network that computes scores for candidates in $\mathcal{C}_q$. We estimate the parameters of the architecture by maximizing the likelihood of observations. For prediction, we then output the candidate that achieves the highest probability.

### 3.2.2 Related Work

**Multi-hop question answering** Many previous systems (from the time of writing), namely BiDAF (Seo et al., 2017), FastQA (Weissenborn et al., 2017), Coref-GRU (Dhingra et al., 2018), MHPGM (Bauer et al., 2018), and Weaver / Jenga (Raison et al., 2018) have been applied to multi-document question answering. The first two mainly focus on single document QA and Welbl et al. (2018) adapted both of them to work with WikiHop. They process each instance of the dataset by concatenating all $d \in \mathcal{S}_q$ in a random order adding document separator tokens. They trained using the first answer mention in the concatenated document and evaluating exact match at test time. Coref-GRU, similarly to us, encodes relations between entity mentions in the document. Instead of using graph neural network layers, as we do, they augment RNNs with jump links corresponding to pairs of corefereed mentions. MHPGM uses a multi-attention mechanism in combination with external commonsense relations to perform multiple hops of reasoning. Weaver is a deep co-encoding model that uses several alternating bi-LSTMs to process the concatenated documents and the query. Subsequent work to ours tackle the multi-hop question answering under different angles using both different version of graph neural networks or more modern architectures like transformers (Vaswani et al., 2017). We discuss some of them in section 3.6.

**Graph neural networks** Graph neural networks have been shown successful on a number of NLP tasks (Marcheggiani & Titov, 2017; Bastings et al., 2017; Zhang et al., 2018a), including those involving document level modeling (Peng et al., 2017). They have also been applied in the context of asking questions about knowledge contained in a knowledge base (Zhang et al., 2018b). In Schlichtkrull et al. (2018), GCNs are used to capture reasoning chains in a knowledge base. Our work and unpublished concurrent work by Song et al. (2018) are the first to study graph neural networks in the context of multi-document QA. Besides differences in the architecture, Song et al. (2018) propose to train a combination of a graph recurrent network and an RNN encoder. We do not train any RNN document encoders for the work on this chapter.

## 3.3 Method

In this section we explain our method presenting the building blocks that make up our Entity-GCN model, namely, an *entity graph* used to relate mentions to entities within and across documents, a *document encoder* used to obtain representations of mentions in context, and a *relational graph convolutional network* that propagates information through the entity graph.

### 3.3.1 Reasoning on an Entity Graph

**Entity graph** In an offline step, we organize the content of each training instance in a graph connecting mentions of candidate answers within and across supporting documents. For a given query $q = \langle s, r, ? \rangle$, we identify mentions in $\mathcal{S}_q$ of the entities in $\mathcal{C}_q \cup \{s\}$ and create one node per mention. This process is based on the following heuristic:

i) we consider mentions spans in $\mathcal{S}_q$ exactly matching an element of $\mathcal{C}_q \cup \{s\}$. Admittedly, this is a rather simple strategy which may suffer from low recall. See chapters 4, 5, and 6 for an in depth exploration of entity recognition, disambiguation, and linking models filling this gap.

ii) we use predictions from a coreference resolution system to add mentions of elements in $\mathcal{C}_q \cup \{s\}$ beyond exact matching (including both noun phrases and anaphoric pronouns). In particular, we use the end-to-end coreference resolution by Lee et al. (2017).

iii) we discard mentions which are ambiguously resolved to multiple coreference chains; this may sacrifice recall, but avoids propagating ambiguity.

To each node $v_i$, we associate a continuous annotation $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ which represents an entity in the context where it was mentioned (details in section 3.3.2). We then proceed to connect these mentions

i) if they co-occur within the same document (`DOC-BASED` edges);

ii) if the pair of named entity mentions is identical (`MATCH` edges—these may connect nodes across and within documents);

iii) if they are in the same coreference chain, as predicted by the external coreference system (`COREF` edges).

Note that `MATCH` edges when connecting mentions in the same document are mostly included in the set of edges predicted by the coreference system. Having the two types of edges lets us distinguish between less reliable edges provided by the coreference system and more reliable (but also more sparse) edges given by the exact-match heuristic. We treat these three types of connections as three

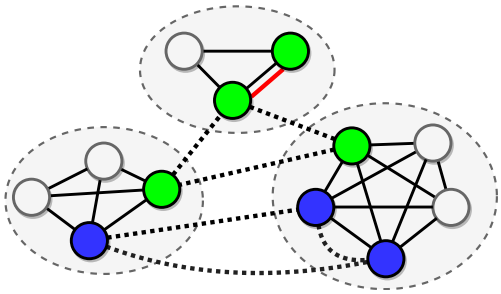


Figure 3.2: Supporting documents (dashed ellipses) organized as a graph where nodes are mentions of either candidate entities or query entities. Nodes with the same color indicates they refer to the same entity (exact match, coreference or both). Nodes are connected by three simple relations: one indicating co-occurrence in the same document (solid edges), another connecting mentions that exactly match (dashed edges), and a third one indicating a coreference (bold-red line).

different types of relations. See figure 3.2 for an illustration. In addition to that, and to prevent having disconnected graphs, we add a fourth type of relation (`COMPLEMENT` edge) between any two nodes that are not connected with any of the other relations. We can think of these edges as those in the complement set of the entity graph with respect to a fully connected graph.

**Multi-step reasoning** Our model then approaches multi-step reasoning by transforming node representations (section 3.3.2 for details) with a differentiable message passing algorithm that propagates information through the entity graph. The algorithm is parameterized by a graph convolutional network (GCN) (Kipf & Welling, 2017), in particular, we employ relational-GCNs (Schlichtkrull et al., 2018), an extended version that accommodates edges of different types. In section 3.3.3 we describe the propagation rule.

Each step of the algorithm (also referred to as a *hop*) updates all node representations in parallel. In particular, a node is updated as a function of messages from its direct neighbours, and a message is possibly specific to a certain relation. At the end of the first step, every node is aware of every other node it connects directly to. Besides, the neighbourhood of a node may include mentions of the same entity as well as others (*e.g.,* same-document relation), and these mentions may have occurred in different documents. Taking this idea recursively, each further step of the algorithm allows a node to indirectly interact with nodes already known to their neighbours. After $l$ layers of R-GCN, information has been propagated through paths connecting up to $l + 1$ nodes.

We start with node representations $\{\boldsymbol{h}^{(0,i)}\}_{i=1}^{n}$ (at the 0-th layer, *i.e.,* input), and transform them by applying $l$ layers of R-GCN obtaining $\{\boldsymbol{h}^{(l,i)}\}_{i=1}^{n}$. Together with a representation $\boldsymbol{q} \in \mathbb{R}^k$ of the query, we define a distribution over candidate answers and we train maximizing the likelihood of observations. The probability

of selecting a candidate $c \in \mathcal{C}_q$ as an answer is then

$$p(c|q, \mathcal{C}_q, \mathcal{S}_q) \propto \exp\left(\max_{i \in \mathcal{M}_c} f_o([\boldsymbol{q}, \boldsymbol{h}^{(l,i)}])\right) , \tag{3.1}$$

where $f_o$ is a parameterized affine transformation, and $\mathcal{M}_c$ is the set of node indices such that $i \in \mathcal{M}_c$ only if node $v_i$ is a mention of $c$. The max operator in equation 3.1 is a design choice but necessary[3] to select the node with highest predicted probability since a candidate answer is realized in multiple locations via different nodes.

### 3.3.2 Node Annotations

Keeping in mind we want an efficient model, we encode words in supporting documents and in the query using only a pre-trained model for contextualized word representations rather than training our own encoder. Specifically, we use ELMo[4] (Peters et al., 2018), a pre-trained bi-directional language model that relies on character-based input representation. ELMo representations, differently from other pre-trained word-based models (*e.g., word2vec* or GloVe; Mikolov et al., 2013; Pennington et al., 2014b), are contextualized since each token representation depends on the entire text excerpt (*i.e.,* the whole sentence).

We choose not to fine tune nor propagate gradients through the ELMo architecture, as it would have defied the goal of not having specialized RNN encoders. In the experiments, we will also ablate the use of ELMo showing how our model behaves using non-contextualized word representations (we use GloVe).

**Documents pre-processing** ELMo encodings are used to produce a set of representations $\{\boldsymbol{x}^{(i)}\}_{i=1}^{n}$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ denotes the $i$-th candidate mention in context. Note that these representations do not depend on the query yet and no trainable model was used to process the documents so far, that is, we use ELMo as a fixed pre-trained encoder. Therefore, we can pre-compute representation of mentions once and store them for later use.

**Query-dependent mention encodings** ELMo encodings are used to produce a query representation $\boldsymbol{q} \in \mathbb{R}^k$ as well. Here, $\boldsymbol{q}$ is a concatenation of the final outputs from a bidirectional RNN layer trained to re-encode ELMo representations of words in the query. The vector $\boldsymbol{q}$ is used to compute a query-dependent representation of mentions $\{\hat{\boldsymbol{x}}^{(i)}\}_{i=1}^{n}$ as well as to compute a probability distribution

[3] During early experimentation we used different formulation and we discovered the max operator to work better.

[4] The use of ELMo is an implementation choice, and, in principle, any other contextual pre-trained model could be used (*e.g.,* Radford et al., 2018b; Devlin et al., 2019a). At the time of writing, ELMo was the only available model for contextualized embedding model.

over candidates (as in equation 3.1). Query-dependent mention encodings $\hat{\boldsymbol{x}}^{(i)} = f_x(\boldsymbol{q}, \boldsymbol{x}^{(i)})$ are generated by a trainable function $f_x$ which is parameterized by a feed-forward neural network.

### 3.3.3 Entity Relational Graph Convolutional Network

Our model uses a gated version of the original R-GCN propagation rule. At the first layer, all hidden node representation are initialized with the query-aware encodings $\boldsymbol{h}^{(0,i)} = \hat{\boldsymbol{x}}^{(i)}$. Then, at each layer $0 \leq \ell \leq l$, the update message $\boldsymbol{u}^{(\ell,i)}$ to the $i$-th node is a sum of a transformation $f_s$ of the current node representation $\boldsymbol{h}^{(\ell,i)}$ and transformations of its neighbours:

$$\boldsymbol{u}^{(\ell,i)} = f_s(\boldsymbol{h}^{(\ell,i)}) + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \sum_{r \in \mathcal{R}_{ij}} f_r(\boldsymbol{h}^{(\ell,j)}) \ , \tag{3.2}$$

where $\mathcal{N}_i$ is the set of indices of nodes neighbouring the $i$-th node, $\mathcal{R}_{ij}$ is the set of edge annotations between $i$ and $j$, and $f_r$ is a parametrized function specific to an edge type $r \in \mathcal{R}$. Recall the available relations from section 3.3.1, namely, $\mathcal{R} = \{\texttt{DOC-BASED}, \texttt{MATCH}, \texttt{COREF}, \texttt{COMPLEMENT}\}$.

A gating mechanism regulates how much of the update message propagates to the next step. This provides the model a way to prevent completely overwriting past information. Indeed, if all necessary information to answer a question is present at a layer which is not the last, then the model should learn to stop using neighbouring information for the next steps. Gate levels are computed as

$$\boldsymbol{a}^{(\ell,i)} = \sigma\left(f_a\left([\boldsymbol{u}^{(\ell,i)}, \boldsymbol{h}^{(\ell,i)}]\right)\right) \ , \tag{3.3}$$

where $\sigma$ is the sigmoid function (*i.e.*, $\sigma : x \mapsto (1 + \exp(-x))^{-1}$) and $f_a$ a parametrized transformation. Ultimately, the updated representation is a gated combination of the previous representation and a non-linear transformation of the update message:

$$\boldsymbol{h}^{(\ell+1,i)} = \phi(\boldsymbol{u}^{(\ell,i)}) \odot \boldsymbol{a}^{(\ell,i)} + \boldsymbol{h}_i^{(\ell,i)} \odot (1 - \boldsymbol{a}^{(\ell,i)}) \ , \tag{3.4}$$

where $\phi(\cdot)$ is any nonlinear function (for our experiments we fixed to be the hyperbolic tangent function tanh) and $\odot$ stands for element-wise multiplication. All transformations $f_*$ are affine and they are not layer-dependent (since we would like to use as few parameters as possible to decrease model complexity promoting efficiency and scalability).

## 3.4 Experimental Setting

**WikiHop** We use WikiHop (Welbl et al., 2018) for training, validation/development and test. The test set is not publicly available and therefore we measure

performance on the validation set in almost all experiments. WikiHop has 43,738/ 5,129/ 2,451 query-documents samples in the training, validation and test sets respectively for a total of 51,318 samples. Authors constructed the dataset as described in section 3.2.1 selecting samples with a graph traversal up to a maximum chain length of 3 documents (see table 3.2 for additional dataset statistics). WikiHop comes in two versions, a standard (unmasked) one and a masked one. The masked version was created by the authors to test whether methods are able to learn lexical abstraction. In this version, all candidates and all mentions of them in the support documents are replaced by random but consistent placeholder tokens. Thus, in the masked version, mentions are always referred to via unambiguous surface forms. We do not use coreference systems in the masked version as they rely crucially on lexical realization of mentions and cannot operate on masked tokens.

**Architecture** See table 3.1 for an outline of Entity-GCN architectural detail. Here the computational steps:

i) ELMo embeddings are a concatenation of three 1024-dimensional vectors resulting in 3072-dimensional input vectors $\{\boldsymbol{x}^{(i)}\}_{i=1}^{n}$.

ii) For the query representation $\boldsymbol{q}$, we apply 2 bi-LSTM layers of 256 and 128 hidden units to its ELMo vectors. The concatenation of the forward and backward states results in a 256-dimensional question representation.

iii) ELMo embeddings of candidates are projected to 256-dimensional vectors, concatenated to the $\boldsymbol{q}$, and further transformed with a 2 layers MLP of 1024 and 512 hidden units in 512-dimensional query aware entity representations $\hat{\boldsymbol{x}}^{(i)} \in \mathbb{R}^{512}$.

iv) All transformations $f_*$ in R-GCN-layers are affine and they do maintain the input and output dimensionality of node representations the same (512-dimensional).

v) Eventually, a 2-layers MLP with $[256, 128]$ hidden units takes the concatenation between $\{\boldsymbol{h}^{(l,i)}\}_{i=1}^{n}$ and $\boldsymbol{q}$ to predict the probability that a candidate node $v_i$ may be the answer to the query $q$ (see equation 3.1).

During preliminary trials, we experimented with different numbers of R-GCN-layers (in the range 1-7). We observed that with WikiHop, for $l \geq 3$ models reach essentially the same performance, but more layers increase the time required to train them. Besides, we observed that the gating mechanism learns to keep more and more information from the past at each layer making unnecessary to have more layers than required.

| **Input:** query $q$ and vertices $\{v_i\}_{i=1}^n$ | |
|---|---|
| query ELMo 3072-dim | candidates ELMo 3072-dim |
| 2 bi-LSTM layers $[256, 128]$-dim | 1 FF layer 256-dim |
| concatenation 512-dim | |
| 2 FF layer $[1024, 512]$-dim: $\{\hat{\boldsymbol{x}}^{(i)}\}_{i=1}^n$ | |
| 3 R-GCN layers 512-dim each (shared parameters) | |
| concatenation with $\boldsymbol{q}$ 768-dim | |
| 3 FF layers $[256, 128, 1]$-dim | |
| **Output:** probabilities over $C_q$ | |

Table 3.1: EntityGCN model architecture.

| | **Minimum** | **Maximum** | **Average** | **Median** |
|---|---|---|---|---|
| # candidates | 2 | 79 | 19.8 | 14 |
| # documents | 3 | 63 | 13.7 | 11 |
| # tokens per document | 4 | 2,046 | 100.4 | 91 |

Table 3.2: WikiHop dataset statistics from Welbl et al. (2018): number of candidates and documents per sample and document length.

**Training Details** We train our models with a batch size of 32 for at most 20 epochs using the Adam optimizer (Kingma & Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate of 1e-4. To help against overfitting, we employ dropout (Srivastava et al., 2014b) testing rates $\in 0, 0.1, 0.15, 0.2, 0.25$ and early-stopping on validation accuracy. We report the best results of each experiment based on accuracy on validation set.

## 3.5 Results

In this section, we compare our method against recent work as well as preforming an ablation study using the WikiHop dataset (Welbl et al., 2018).

### 3.5.1 Comparison

In this experiment, we compare our Enitity-GCN against all the prior work on the same task available at the time of writing—we also report results from subsequent work to ours for completeness but we do not do an in-depth comparison with those in this section. We present test and development results (when present) for both

versions of the dataset in table 3.3. From Welbl et al. (2018), we list an oracle based on human performance as well as two standard reading comprehension models, namely BiDAF (Seo et al., 2017) and FastQA (Weissenborn et al., 2017). We also compare against Coref-GRU (Dhingra et al., 2018), MHPGM (Bauer et al., 2018), and Weaver (Raison et al., 2018). Additionally, we include results of MHQA-GRN (Song et al., 2018), from a recent arXiv preprint describing concurrent work. They jointly train graph neural networks and recurrent encoders. We report single runs of our two best single models and an ensemble one on the unmasked test set (recall that the test set is not publicly available and the task organizers only report unmasked results) as well as both versions of the validation set.

Entity-GCN (best single model without coreference edges) outperforms all previous work by over 2% points. We additionally re-ran BiDAF baseline to compare training time: when using a single Titan X GPU, BiDAF and Entity-GCN process 12.5 and 57.8 document sets per second, respectively. Note that Welbl et al. (2018) had to use BiDAF with very small state dimensionalities (20), and smaller batch size due to the scalability issues (both memory and computation costs). We compare applying the same reductions.[5] Eventually, we also report an ensemble of 5 independently trained models. All models are trained on the same dataset splits with different weight initializations. The ensemble prediction combining each model ($m = 5$) is obtained as

$$\arg\max_{c \in \mathcal{C}_q} \prod_{i=1}^{m} p(c|q, \mathcal{C}_q, \mathcal{S}_q; \theta_i) \ , \tag{3.5}$$

where $\theta_i$ are the parameters of the $i$-th model in the ensemble.

### 3.5.2 Ablation Study

To help determine the sources of improvements, we perform an ablation study using the publicly available validation set (see table 3.4). We perform two groups of ablation, one on the embedding layer, to study the effect of ELMo, and one on the edges, to study how different relations affect the overall model performance.

**Embedding ablation** We argue that ELMo is crucial, since we do not rely on any other context encoder. However, it is interesting to explore how our R-GCN performs without it. Therefore, in this experiment, we replace the deep contextualized embeddings of both the query and the nodes with GloVe (Pennington et al., 2014b) vectors (insensitive to context). Since we do not have any component in our model that processes the documents, we expect a drop in performance. In other words, in this ablation our model tries to answer questions

[5] Besides, we could not run any other method we compare with combined with ELMo without reducing the dimensionality further or having to implement a distributed version.

| Model | Unmasked | | Masked | |
|---|---|---|---|---|
| | **Test** | **Dev** | **Test** | **Dev** |
| Human (Welbl et al., 2018) | 74.1 | – | – | – |
| FastQA (Welbl et al., 2018) | 25.7 | – | 35.8 | – |
| BiDAF (Welbl et al., 2018) | 42.9 | – | 54.5 | – |
| Coref-GRU (Dhingra et al., 2018) | 59.3 | 56.0 | – | – |
| MHPGM (Bauer et al., 2018) | – | 58.2 | – | – |
| Weaver / Jenga (Raison et al., 2018) | 65.3 | 64.1 | – | – |
| MHQA-GRN (Song et al., 2018) | 65.4 | 62.8 | – | – |
| Entity-GCN without coreference (single model) | **67.6** | **64.8** | – | 70.5 |
| Entity-GCN with coreference (single model) | **66.4** | **65.3** | – | – |
| Entity-GCN* (ensemble 5 models) | **71.2** | **68.5** | – | 71.6 |
| Subsequent work | | | | |
| BAG (Cao et al., 2019) | 69.0 | 66.5 | 68.9 | 70.9 |
| HDE (Tu et al., 2019) | 70.9 | 68.1 | – | – |
| DynSAN (Zhuang & Wang, 2019) | 71.4 | 70.1 | – | – |
| Tang et al. (2020) | 72.5 | 70.8 | – | – |
| HDE ensemble (Tu et al., 2019) | 74.3 | 70.9 | – | – |
| Chen et al. (2019) | 76.5 | 72.2 | – | – |
| Tang et al. (2020) (ensemble) | 78.3 | 74.0 | – | – |
| Longformer (Beltagy et al., 2020) | 81.9 | – | – | – |
| BigBird (Zaheer et al., 2020) | 82.3 | 75.9 | – | – |
| RealFormer (He et al., 2021) | 84.4 | 79.2 | – | – |

Table 3.3: Accuracy of different models on WikiHop closed test set and public validation set. Our Entity-GCN outperforms recent prior work without learning any language model to process the input but relying on a pre-trained one (ELMo – without fine-tunning it) and applying R-GCN to reason among entities in the text. * with coreference for unmasked dataset and without coreference for the masked one.

*without reading the context at all.* For example, in figure 3.1, our model would be aware that "Stockholm" and "Sweden" appear in the same document but any context words, including the ones encoding relations (*e.g.,* "is the capital of") will be hidden. Besides, in the masked case all mentions become 'unknown' tokens with GloVe and therefore the predictions are equivalent to a random guess. Once the strong pre-trained encoder is out of the way, we also ablate the use of our R-GCN component, thus completely depriving the model from inductive biases that aim at multi-hop reasoning.

The first important observation is that replacing ELMo by GloVe (GloVe

| **Model** | **Unmasked** | **Masked** |
|---|---|---|
| *full* (ensemble) | **68.5** | **71.6** |
| *full* (single) | **65.1** ± 0.11 | **70.4** ± 0.12 |
| GloVe with R-GCN | 59.2 | 11.1 |
| GloVe w/o R-GCN | 51.2 | 11.6 |
| No R-GCN | 62.4 | 63.2 |
| No relation types | 62.7 | 63.9 |
| No `DOC-BASED` | 62.9 | 65.8 |
| No `MATCH` | 64.3 | 67.4 |
| No `COREF` | 64.8 | – |
| No `COMPLEMENT` | 64.1 | 70.3 |
| Induced edges | 61.5 | 56.4 |

Table 3.4: Ablation study on WikiHop validation set. The *full model* is our Entity-GCN with all of its components and other rows indicate models trained without a component of interest. We also report baselines using GloVe instead of ELMo with and without R-GCN. For the *full model* we report mean ±1 std over 5 runs.

with R-GCN in table 3.4) still yields a competitive system that ranks far above baselines from (Welbl et al., 2018) and even above the Coref-GRU of Dhingra et al. (2018), in terms of accuracy on (unmasked) validation set. The second important observation is that if we then remove R-GCN (GloVe w/o R-GCN in table 3.4), we lose 8.0 points. That is, the R-GCN component pushes the model to perform above Coref-GRU still without accessing context, but rather by updating mention representations based on their relation to other ones. These results highlight the impact of our R-GCN component.

**Graph edges ablation** In this experiment we investigate the effect of the different relations available in the entity graph and processed by the R-GCN module. We start off by testing our stronger encoder (*i.e.,* ELMo) in absence of edges connecting mentions in the supporting documents (*i.e.,* using only self-loops – No R-GCN in table 3.4). The results suggest that WikipHop genuinely requires multihop inference, as our best model is 6.1% and 8.4% more accurate than this local model, in unmasked and masked settings, respectively.[6] However, it also shows that ELMo representations capture predictive context features, without being explicitly trained for the task. It confirms that our goal of getting away with training expensive document encoders is a realistic one.

We then inspect our model's effectiveness in making use of the structure

[6] Recall that all models in the ensemble use the same local representations, ELMo.

| | Relation | Accuracy | Hit@2 | Hit@5 | Avg. $\lvert\mathcal{C}_q\rvert$ | Supports |
|---|---|---|---|---|---|---|
| | overall (ensemble) | 68.5 | 81.0 | 94.1 | 20.4 ± 16.6 | 5129 |
| | overall (single model) | 65.3 | 79.7 | 92.9 | 20.4 ± 16.6 | 5129 |
| **3 best** | member_of_political_party | 85.5 | 95.7 | 98.6 | 5.4 ± 2.4 | 70 |
| | record_label | 83.0 | 93.6 | 99.3 | 12.4 ± 6.1 | 283 |
| | publisher | 81.5 | 96.3 | 100.0 | 9.6 ± 5.1 | 54 |
| **3 worst** | place_of_birth | 51.0 | 67.2 | 86.8 | 27.2 ± 14.5 | 309 |
| | place_of_death | 50.0 | 67.3 | 89.1 | 25.1 ± 14.3 | 159 |
| | inception | 29.9 | 53.2 | 83.1 | 21.9 ± 11.0 | 77 |

Table 3.5: Accuracy and hit at K analysis overall and per query type. Avg. $\lvert\mathcal{C}_q\rvert$ indicates the average number of candidates with one standard deviation.

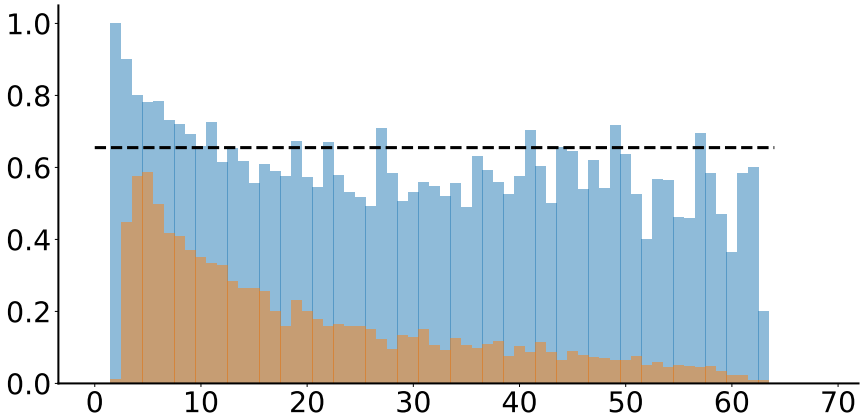
encoded in the graph. We start naively by fully-connecting all nodes within and across documents without distinguishing edges by type (No relation types in table 3.4). We observe only marginal improvements with respect to ELMo alone (No R-GCN in table 3.4) in both the unmasked and masked setting suggesting that a GCN operating over a naive entity graph would not add much to this task and a more informative graph construction and/or a more sophisticated parameterization is indeed needed.

Next, we ablate each type of relations independently, that is, we either remove connections of mentions that co-occur in the same document (`DOC-BASED`), connections between mentions matching exactly (`MATCH`), or edges predicted by the coreference system (`COREF`). The first thing to note is that the model makes better use of `DOC-BASED` connections than `MATCH` or `COREF` connections. This is mostly because i) the majority of the connections are indeed between mentions in the same document, and ii) without connecting mentions within the same document we remove important information since the model is unaware they appear closely in the document. Secondly, we notice that coreference links and complement edges seem to play a more marginal role. Though it may be surprising for coreference edges, recall that the `MATCH` heuristic already captures the easiest coreference cases, and for the rest the out-of-domain coreference system may not be reliable. Still, modelling all these different relations together gives our Entity-GCN a clear advantage. This is our best system evaluating on the development. Since Entity-GCN seems to gain little advantage using the coreference system, we report test results both with and without using it. Surprisingly, with coreference, we observe performance degradation on the test set. It is likely that the test documents are harder for the coreference system.[7]

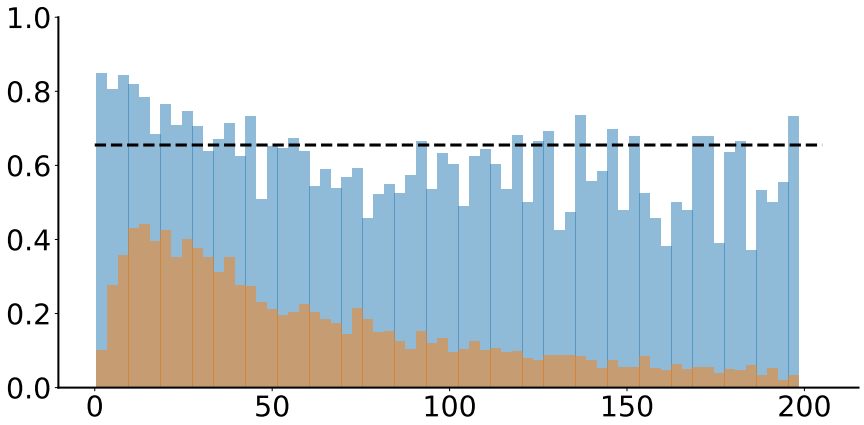
We do perform one last ablation, namely, we replace our heuristic for assigning edges and their labels by a model component that predicts them. The last row of table 3.4 (Induced edges) shows model performance when edges are not predetermined but predicted. For this experiment, we use a bilinear function $f_e(\hat{\boldsymbol{x}}^{(i)}, \hat{\boldsymbol{x}}^{(j)}) = \sigma\left((\hat{\boldsymbol{x}}^{(i)})^\top \boldsymbol{W}_e\, \hat{\boldsymbol{x}}^{(j)}\right)$ that predicts the importance of a single edge connecting two nodes $i, j$ using the query-dependent representation of mentions (see section 3.3.2). The performance drops below 'No R-GCN' suggesting that it cannot learn these dependencies on its own.

Most results are stronger for the masked settings even though we do not apply the coreference resolution system in this setting due to masking. It is not surprising as coreferred mentions are labeled with the same identifier in the masked version, even if their original surface forms did not match (Welbl et al. (2018) used WIKIPEDIA links for masking). Indeed, in the masked version, an entity is always referred to via the same unique surface form (*e.g.,* `MASK1`) within and across documents. In the unmasked setting, on the other hand, mentions to an entity may differ (*e.g.,* "US" vs "United States") and they might not be

[7] Since the test set is hidden from us, we cannot analyze this difference further.




(a) Candidates set size (x-axis) and accuracy (y-axis). Pearson's correlation of $-0.687$ ($p$ <1e-7).




(b) Nodes set size (x-axis) and accuracy (y-axis). Pearson's correlation of $-0.385$ ($p$ <1e-7).

Figure 3.3: Accuracy (blue) of our best single model with respect to the candidate set size (on the *top*) and nodes set size (on the *bottom*) on the validation set. Data distributions (orange) per number of candidate (*top*) and nodes (*bottom*). Dashed lines indicate average accuracy.

retrieved by the coreference system we are employing, making the task harder for all models. Therefore, as we rely mostly on exact matching when constructing our graph for the masked case, we are more effective in recovering coreference links on the masked rather than unmasked version.[8]

[8] Though other systems do not explicitly link matching mentions, they similarly benefit from masking (*e.g.,* masks essentially single out spans that contain candidate answers).

### 3.5.3 Error Analysis

In this section we provide an error analysis for our best single model predictions. First of all, we look at which type of questions our model performs well or poorly. There are more than 150 query types in the validation set but we filtered the three with the best and with the worst accuracy that have at least 50 supporting documents and at least 5 candidates. We show results in table 3.5. We observe that questions regarding places (birth and death) are considered harder for Entity-GCN. We then inspect samples where our model fails while assigning highest likelihood and noticed two principal sources of failure i) a mismatch between what is written in WIKIPEDIA and what is annotated in WIKIDATA, and ii) a different degree of granularity (*e.g.,* born in "London" vs "UK" could be considered both correct by a human but not when measuring accuracy).

Secondly, we study how the model performance degrades when the input graph is large. In particular, we observe a negative Pearson's correlation ($-0.687$) between accuracy and the number of candidate answers. However, the performance does not decrease steeply. The distribution of the number of candidates in the dataset peaks at 5 and has an average of approximately 20. Therefore, the model does not see many samples where there are a large number of candidate entities during training. Differently, we notice that as the number of nodes in the graph increases, the model performance drops but more gently (negative but closer to zero Pearson's correlation). This is important as document sets can be large in practical applications. In figure 3.3, we show how the model performance goes when the input graph is large. In particular, how Entity-GCN performs as the number of candidate answers or the number of nodes increases.

Finally, in table 3.6, we report three samples from WikiHop development set where out Entity-GCN fails. In particular, we show two instances where our model presents high confidence on the answer, and one where is not. We commented these samples explaining why our model might fail in these cases.

## 3.6 Subsequent Work

Subsequent work to ours tackles the multi-hop question answering in different ways using different versions of graph neural networks or more modern architectures like transformers (Vaswani et al., 2017). There has also been progress from a data perspective. Results on WikiHop from subsequent work are shown in the bottom part of table 3.3.

**Advances in datasets** Because WikiHop was automatically generated, the dataset has some limitations. Namely:

i) *questions are not expressed in natural language*: this becomes problematic when want to deploy the model in the real world as queries are unrealistic for

| | |
|---|---|
| **ID** | WH_dev_2257 |
| **Gold answer** | 2003 ($p(c\|q, \mathcal{C}_q, \mathcal{S}_q) = 0.141$) |
| **Query** | inception (of) Derrty Entertainment |
| **Predicted answer** | 2000 ($p(c\|q, \mathcal{C}_q, \mathcal{S}_q) = 0.158$) |
| **Support 1** | **Derrty Entertainment** is a record label founded by [...]. The first album released under **Derrty Entertainment** was Nelly 's **Country Grammar**. |
| **Support 2** | **Country Grammar** is the debut single by American rapper Nelly. The song was produced by Jason Epperson. It was released in **2000**, [...] |

(a) In this example, the model predicts the answer correctly. However, there is a mismatch between what is written in WIKIPEDIA and what is annotated in WIKIDATA. In WikiHop, answers are generated with WIKIDATA.

| | |
|---|---|
| **ID** | WH_dev_2401 |
| **Gold answer** | Adolph Zukor ($p(c\|q, \mathcal{C}_q, \mathcal{S}_q) = 7.1\text{e-}6$) |
| **Query** | producer (of) Forbidden Paradise |
| **Predicted answer** | Jesse L. Lask ($p(c\|q, \mathcal{C}_q, \mathcal{S}_q) = 0.999$) |
| **Support 1** | **Forbidden Paradise** is a [...] drama film produced by **Famous Players-Lasky** [...] |
| **Support 2** | **Famous Players-Lasky** Corporation was [...] from the merger of **Adolph Zukor**'s Famous Players Film Company [..] and the **Jesse L. Lasky** Feature Play Company. |

(b) In this sample, there is ambiguity between two entities since both are correct answers reading the passages but only one is marked as correct. The model fails assigning very high probability to only on one of them.

| | |
|---|---|
| **ID** | WH_dev_3030 |
| **Gold answer** | Scania ($p(c\|q, \mathcal{C}_q, \mathcal{S}_q)$ =2.9e-4) |
| **Query** | place_of_birth (of) Erik Penser |
| **Predicted answer** | Eslöv ($p(c\|q, \mathcal{C}_q, \mathcal{S}_q) = 0.973$) |
| **Support 1** | Nils Wilhelm **Erik Penser** (born August 22, 1942, in **Eslöv**, **Skåne**) is a Swedish [...] |
| **Support 2** | **Skåne** County, sometimes referred to as " **Scania** County " in English, is the [...] |

(c) In this sample, there is ambiguity between two entities since the city Eslöv is located in the Scania County (English name of Skåne County). The model assigning high probability to the city and it cannot select the county.

Table 3.6: Samples from WikiHop set where Entity-GCN fails. $p$ indicates the predicted likelihood.

humans, and additional work would be needed to convert a natural question into a structured query.

ii) *questions are generated using a KG*: as we explain in section 3.2.1 Welbl et al. (2018) constructed the question and evidence set of WikiHop using the relations from a KB. Unfortunately, although queries are realistic relations that humans annotated in the KB, there is no certainty that the relation(s) will be expressed in the text. Therefore, it might be impossible for both humans and machines to answer them. This is also evident from table 3.3 were human performance is just below 75%.

To address these issues, Yang et al. (2018b) proposed HotpotQA: an alternative dataset for multi-hop question answering that has become the standard benchmark for this task for the past few years. HotpotQA uses both automation and human annotation to construct the dataset. The authors used hyperlinks from WIKIPEDIA to automatically select pairs of pages. Then they asked human annotators to come out with natural language questions that needed to involve information from both sources. Additionally, they added some extra questions that involved comparing information between pages. Most of the recent literature on this task focused on HotpotQA rather than WikiHop. The resulting dataset is considerably large, containing 90,564 training, 7,405 validation, and 14,810 test examples, respectively. HotpotQA is freely available, and it has a public leaderboard.[9]

[9] https://hotpotqa.github.io

**Advances in graph neural networks** Some works after ours applied other versions of graph neural networks to the multi-hop question-answering problem. Here are some interesting examples:

- *Bi-directional attention Entity-GCN* (Cao et al., 2019) extends our approach leveraging relationships between nodes in an entity graph but it additionally employs a bi-attention layer between the query and the entity graph (Seo et al., 2017; Xiong et al., 2017). They also use a combination of NER features, POS features, ELMo, and Glove embeddings to initialize the node representation as well as R-GCN layers to process them.

- *Heterogeneous Document-Entity* (Tu et al., 2019) constructs a graph with different granularity levels of information including candidates, documents, and entities (similar to ours). They also use a combination of R-GCN, bi-attention, and self-attention (Vaswani et al., 2017) layers to process and combine the information from different sources.

- *Hierarchical Graph Network* (Fang et al., 2020) aggregates the information from texts across multiple paragraphs, building first a hierarchical graph by constructing nodes on different levels of granularity (questions, paragraphs, sentences, entities). Then, for graph propagation, they use Graph Attention Network (Velickovic et al., 2018). The advantage of a hierarchical formulation is that it encodes a structural bias into the processing of the graph. Finally, they initialize the representations of nodes which are initialized with a pre-trained model.

**Path-based methods** Some works after ours investigated the reasoning problem from a path perspective: they sequentially retrieve evidence paragraphs in a reasoning path by conditioning on the previously retrieved documents. In particular:

- Chen et al. (2019) uses distant supervision constructing reasoning chains via heuristics relying on named entity recognition and coreference resolution. The model learns to extract chains from the raw text; thus, chains at test time are predicted. As a document encoder, they use BERT (Devlin et al., 2019b) and then use LSTM-based pointer network (Vinyals et al., 2015) to process the reasoning chain sequentially.

- Tang et al. (2020) uses a gated R-GCN that is the same GCN layers we proposed while adding distant supervision with reasoning chains. Similarly to Cao et al. (2019), they apply bi-directional attention on top of the node and query representations.

- Asai et al. (2020) introduces a graph-based recurrent retrieval model that learns to retrieve reasoning paths over the WIKIPEDIA hyperlink graph.

Their model is also backed by BERT (Devlin et al., 2019b) model that encodes queries and paragraphs. Then, the authors used beam search for candidate selection of paths of paragraphs. Finally, paragraphs are retrieved with a dot-product search, and paths are scored with an RNN.

**Transformer architectures** As discussed in section 3.1 approaches that concatenate all documents into a single long text and training a standard RNN-based reading comprehension model did not work well for multi-hop question answering (Welbl et al., 2018). That was mainly due to i) the limitation of a single layer of bi-attention and ii) the unavailability of large pre-trained language models at that time. Indeed, transformers-based models (Vaswani et al., 2017) would have been a solid baseline. However, full self-attention has quadratic time and space complexity with respect to the input sequence length making it impractical to use on very long sequences. Zhuang and Wang (2019) were one of the first to propose using only self-attention to encode the passages (*i.e.,* practically using a transformer), employing efficient cross-passage attention. However, only recently have some models been able to approach this task just taking all documents as inputs at once: Longformer (Beltagy et al., 2020), BigBird (Zaheer et al., 2020), and RealFormer (He et al., 2021). Although there are many more versions of sub-quadratic attention layers, an in-depth discussion of such is beyond the scope of this section (see Tay et al. (2022a) for a survey on efficient transformers). All three variants mentioned above have linear time and space complexity on the input size. Thus, these methods made it possible to use all the candidate documents as inputs achieving "super-human" performance on WikiHop (see table 3.3). In this case, some models get accuracy scores higher than human raters even though the it does not mean it was superior to humans. This very high performance was not achieved only because of the architectural design but is largely due to large pre-training on big corpora. If human annotators could not answer some questions based on the evidence of the candidate documents, it is hardly possible that a model would achieve that. Higher than annotators performance in those chases is achieved because models memorized factual information during the pre-training stage and used such to answer correctly even without evidence. Finally, Xiong et al. (2021) used a version dense passage encoding (Karpukhin et al., 2020) to learn to retrieve chains of documents within a large text database (*i.e.,* WIKIPEDIA). That is similar to Asai et al. (2020) but Xiong et al. (2021) learn all the steps end-to-end where Asai et al. (2020) do not train the document encoder. Indeed, on HotpotQA they reported improvements over Asai et al. (2020).

## 3.7 Conclusion

In this chapter, we designed a graph neural network that operates over a compact graph representation of a set of documents where nodes are mentions to entities

and edges signal relations such as within and cross-document coreference. The model learns to answer questions by gathering evidence from different documents via a differentiable message passing algorithm that updates node representations based on their neighbourhood. Our model outperforms published results where ablations show substantial evidence in favour of multi-step reasoning. Moreover, we make the model fast by using pre-trained (contextual) embeddings.

Entities and their mentions are central to the development of Entity-GCN. One main limitation of this work is that we restrict the application of our method to the case where named entity recognition and entity disambiguation are considered as solved steps (*i.e.,* our model receives as inputs a set of tagged and resolved mention-entity pairs). Therefore, relying on a solid performance of mention-entity resolution is crucial for deploying this class of models in the wild. NER and entity disambiguation are becoming mature technologies, but further improvements can be done. In the following three chapters, we tackle both of these tasks proposing methods that improve and extend the capabilities of current algorithms.

# Chapter 4

# Autoregressive Entity Linking

**Chapter Highlights**

Entity linking is the task of finding mention of entities in text and linking them to their corresponding entity identifier in a knowledge base. Current approaches to entity liking, as know as bi-encoders, can be understood as classifiers among atomic labels. They use dot-product search for retrieving from a predicted vector space (index). In this chapter, we propose GENRE, the first system that retrieves entities by generating their unique names, left to right, token-by-token in an autoregressive fashion and conditioned on the context. This enables us to mitigate some technical issues that typical models have and in particular:

i) the autoregressive formulation allows us to directly capture relations between context and entity name, effectively cross encoding both;

ii) the memory footprint is greatly reduced because the parameters of our encoder-decoder architecture scale with vocabulary size, not entity count;

iii) the exact cross entropy loss can be efficiently computed without the need to subsample negative data.

We show the efficacy of the approach, experimenting with more than 20 datasets on entity disambiguation, end-to-end entity linking and document retrieval tasks, achieving new state-of-the-art or very competitive results while using a tiny fraction of the memory footprint of competing systems. Finally, we demonstrate that new entities can be added by simply specifying their unambiguous name which was not effective in traditional bi-encoder approaches.[1]

[1] Source code available at https://github.com/facebookresearch/GENRE

## 4.1 Introduction

The ability to retrieve the correct entity from large Knowledge Bases (KBs) given a textual input is a fundamental building block for several applications (Ferrucci, 2012; Slawski, 2015; Yang et al., 2018a). Indeed, in the previous chapter, we explored how a mention-entity graph extracted from the text can be exploited by a graph neural network to reason across multiple documents. Nevertheless, entity retrieval is useful not just for multi-document reading comprehension. For instance, most commercial recommendation systems include in their pipelines components to detect and disambiguate entity mentions in open text, in order to isolate relevant concepts from non-meaningful data (Slawski, 2015; Yang et al., 2018a). Another example are chat-bots and question answering systems, that are often equipped with retrieval components to surface specific KB entries (*e.g.,* WIKIPEDIA articles) to find knowledge for sustaining a conversation or answering a question (Ferrucci, 2012; Chen et al., 2017a; Lewis et al., 2020b; Roller et al., 2021; Sevegnani et al., 2021).

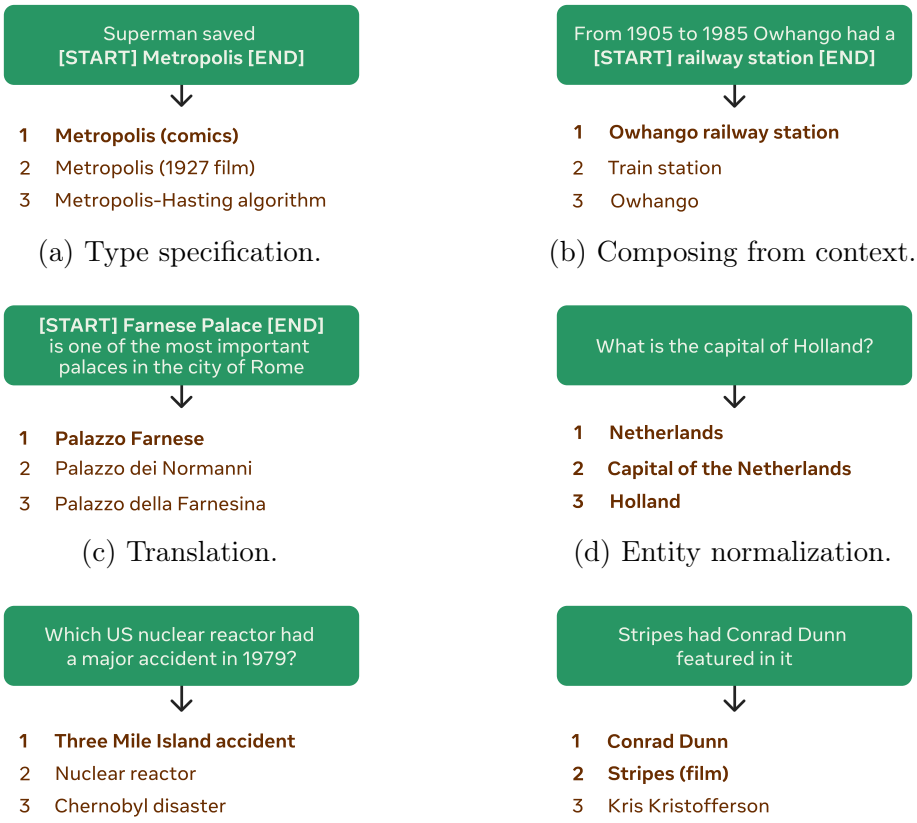
Although there has been extensive previous work on entity retrieval (*e.g.,* Hoffart et al., 2011; Piccinno & Ferragina, 2014; Huang et al., 2015; Le & Titov, 2018; Logeswaran et al., 2019; Broscheit, 2019; Wu et al., 2020, to name just a few) there is a common design choice to most current solutions: entities are associated with a unique atomic label and the retrieval problem can be interpreted as multi-class classification across these labels. The match between input and label is calculated through a bi-encoder (Wu et al., 2020; Karpukhin et al., 2020): a dot product between dense vector encodings of the input and the entity's meta information (such as title and description). Critically, this formulation enables sub-linear search using modern maximum-inner-product-search libraries (Johnson et al., 2019) and hence supports retrieving from large entity databases.

Unfortunately, the classifier approach to entity retrieval also has several shortcomings:

i) unless a costly cross-encoder is used for re-ranking (Wu et al., 2020), the dot-product can miss fine-grained interactions between input and entity meta information (Humeau et al., 2020);

ii) storing dense vectors for the whole KB requires a large memory footprint, especially in real-world scenarios (*i.e.,* $\approx$24GB to store 1024-dimensional vectors for all of the $\approx$6M WIKIPEDIA pages), and the size grows linearly with the addition of new entities;

iii) computing an exact cross entropy loss over all entities is very expensive, hence current solutions need to subsample negative data (Logeswaran et al., 2019; Karpukhin et al., 2020) at training time. Tuning an appropriately hard set of negative instances can be challenging and time-consuming;




(a) Type specification.

(b) Composing from context.

(c) Translation.

(d) Entity normalization.

(e) Implicit factual knowledge.

(f) Exact copy.

Figure 4.1: Examples of entities correctly retrieved from GENRE (we show only the top-3 rank). On the *top* three entity disambiguation instances and on the *bottom* three document retrieval instances, two for open-domain question answering and one for fact checking. All of them are cast as sequence-to-sequence problems while inference is done using constrained beam search. Gold entities in **bold**. Sub-captions indicate the type of interaction between the input context and the entity names required.

iv) existing systems can suffer from a cold-start problem since they cannot represent entities about which they have not yet gathered sufficient information, in the form, for instance, of a textual description or a set of relations with the existing entities.

The treatment of entity identifiers as atomic labels in a classifier ignores the fact that we often have unambiguous, highly structured and compositional entity names. WIKIPEDIA, for instance, associates unique titles to articles,[2] that may be the name of the subject or a description of its topic, as well as potential distinctive

[2] We use *entity name* to refer to the corresponding WIKIPEDIA article title throughout the rest of the chapter.

information to disambiguate [3] (see figure 4.1 for some examples). These entity names often interact with mention contexts in a predictable and regular fashion. For example, often entity names are identical with the mention strings that refer to them (*e.g.,* figure 4.1f). When this is not possible, they might be composed of tokens in the context (*e.g.,* figure 4.1b), include a type specification that can be inferred (*e.g.,* figure 4.1a), be the translation of the string mention (*e.g.,* figure 4.1c), require 'normalization' such as referring to the correct alias of a mention (*e.g.,* figure 4.1d), or require factual knowledge that might be stored in the parameters of a model (*e.g.,* figure 4.1e). These observations suggest that textual inputs could be *translated* into unique entity names, word by word, instead of being classified among a huge set of options.

In this chapter, we propose GENRE (for *Generative ENtity REtrieval*), the first entity retriever that exploits a sequence-to-sequence architecture to generate entity names in an autoregressive fashion conditioned on the context. Concretely, GENRE uses a transformer-based architecture, pre-trained with a language modeling objective (*i.e.,* we use BART weights; Lewis et al., 2020a) and fine-tuned to generate entity names. This architecture has been shown to retain factual knowledge to some extent (Petroni et al., 2019) and language translation skills (Radford et al., 2019) among other things, both desirable properties for an entity retriever. Naturally, the generated output might not always be a valid entity name. To solve this problem, GENRE employs a constrained decoding strategy that forces each generated name to be in a predefined candidate set.

The autoregressive formulation allows us to directly capture the aforementioned relations between context and entity name, effectively cross encoding both. Also, the memory footprint required is orders of magnitude smaller than current systems, since the parameters of a sequence-to-sequence model scale linearly with the vocabulary size, not entity count. Moreover, the exact cross entropy loss can be computed efficiently for each output token (*i.e.,* all non-gold tokens are considered negative), thereby eliminating the need for negative data downsampling. Finally, our model never accesses any explicit meta-information about the entity beyond their title, hence new entities can be added by simply appending their unambiguous name to the candidate set (*e.g.,* figure 4.1b refers to an entity added after training).

We empirically evaluate the performance of GENRE on more than 20 datasets, spanning three families of tasks:

i) entity disambiguation, using popular datasets and settings (both in- and out-of-domain);

ii) end-to-end entity linking, with the GERBIL benchmarking tool (Röder et al., 2018), by using a novel dynamically markup-constrained decoding strategy which transform plain text into structured markup;

[3] often in the form of a description in parentheses after the name. Wikipedia naming conventions are described in https://en.wikipedia.org/wiki/Wikipedia:Article_titles.

iii) document retrieval, with the recently proposed KILT benchmark (Petroni et al., 2021) which spans 5 different sub-tasks.

Our models achieve state-of-the-art or very competitive results on nearly all datasets, often with substantial improvement (+13.7 precision points on KILT for retrieval on average). Further, we show that compared with recent models, GENRE requires substantially less memory ($\approx 20$ times smaller footprint on average). Finally, we demonstrate that our model can be applied in scenarios where the only entity information available is its name.

## 4.2 Background

In the following sections, we first provide a formal definition of the entity retrieval task. Secondly, we present a brief discussion on bi-encoders for retrieval, and finally, we present related work.

### 4.2.1 Task Definition

We assume to have a collection of entities $\mathcal{E}$ (*e.g.,* WIKIPEDIA articles) where each entity is an entry in a Knowledge Base (KB). We want to approach the following retrieval problem: given a textual input source $\mathcal{X}$ (*i.e.,* a sequence of tokens, *e.g.,* a question), a model has to return the most relevant entities from $\mathcal{E}$ with respect to $\mathcal{X}$. We assume that each $e \in \mathcal{E}$ is uniquely assigned to a textual representation (*i.e.,* its name): a sequence of tokens $\mathcal{Y}_e$ specific to $e$ (*e.g.,* WIKIPEDIA pages are uniquely identified by their titles).

A particular instance of this problem is Entity Disambiguation (ED) (see figure 4.1 for an example) where an input $\mathcal{X}$ is annotated with a mention and a system has to select either its corresponding entity from $\mathcal{E}$, or to predict that there is no corresponding entry in the KB. Another instance is page-level Document Retrieval (DR) where the input $\mathcal{X}$ is intended as a query and $\mathcal{E}$ as a collection of documents identified by their unique titles (*e.g.,* WIKIPEDIA articles).

### 4.2.2 Bi-encoders for Retrieval

Bi-encoders (also know as two-tower models, dual-encoder models or Siamese networks; Reimers & Gurevych, 2019; Chicco, 2021) consist of two (usually architecturally identical) encoders (with possibly different parametrizations) which encode textual inputs into vectors. The objective is to create a vector space such that relevant pairs of inputs (which in the case of entity linking are a context and entity meta information but in the case of question answering is a passage and a question) will have a smaller distance (*i.e.,* higher similarity) than the irrelevant ones. These encoders may be trained with different losses, but the overall goal is that they become good ranking functions for retrieval, which is essentially a

metric learning problem (Kulis, 2013). The distance or similarity function is an architectural choice. However, due to computational advantages, the inner-product is often used, which allows search at test time to be an instance of the maximum inner product search (MIPS) problem (Wu et al., 2020). Training these type of models require the availability of a set of relevant and irrelevant pairs. Usually, all the other datapoints except the relevant ones are considered irrelevant, and in this case, in-batch negatives (Yih et al., 2011; Henderson et al., 2017; Gillick et al., 2019; Lerer et al., 2019; Humeau et al., 2020) are used. If available, using hard negatives (*i.e.,* datapoints that are irrelevant but very similar to the relevant one) helps training (Karpukhin et al., 2020).

In the case of entity linking, we have only one relevant entity (*i.e.,* the correct one) and all the others are irrelevant. That is why we consider this problem a classification problem with many labels. However, in most cases, KBs do really consist of millions of entities. Therefore, any loss we use will never contain all the irrelevant entities due to computational constraints. The loss will then be an approximation of the actual loss. After training, bi-encoders produce an index of vectors that will later be used at test time for nearest neighbors search–although being a simple operation, doing an inner-product search among millions of vectors is slow. Fortunately, approximate nearest neighbor (ANN) offers fast alternatives to full search via product quantization (Jegou et al., 2010), hierarchical navigable small world graphs (Malkov & Yashunin, 2018), and other techniques. Modern systems make use of freely available software such as FAISS[4] (Johnson et al., 2019) or Scalable Nearest Neighbors[5] (ScaNN; Guo et al., 2020).

### 4.2.3 Related Work

**Structured Tasks as Seq2Seq** Casting NLP tasks with a structured input or output into sequence-to-sequence problems has been explored for different problems, including semantic parsing (Rongali et al., 2020), semantic role labelling (Daza & Frank, 2018), discourse representation structure parsing (Liu et al., 2018), generation of fluent natural language responses from structured semantic representations (Balakrishnan et al., 2019), generation and parsing of abstract meaning representation (Konstas et al., 2017). In these works a structured representation, a tree or a graph for instance, is linearized into a sequence of symbols compatible with a seq2seq architecture. To the best of our knowledge, we are the first to cast entity retrieval as a sequence-to-sequence problem while decoding with an autoregressive formulation during inference.

**Constrained Generation** Related to our constrained generation mechanism, (Daza & Frank, 2018; Rongali et al., 2020) use a copying mechanism in order to

[4] https://github.com/facebookresearch/faiss
[5] https://github.com/google-research/google-research/tree/master/scann

limit lexical deviations between the input and output strings. In these tasks, as well as for our problem, it is natural to promote a copying mechanism due to the input and the output proximity. A different type of constraint, a structural constraint, is used in (Balakrishnan et al., 2019) to maintain a valid tree structure. Our constrained beam search encompasses both aspects, a copying mechanism that restricts the vocabulary and a structural constraint to obtain a well-formed annotated output. In addition to these tasks with close input and output, the integration of a mechanism to guide the output of neural networks has been explored in various settings. Lexically constrained decoding has been used to force the inclusion of pre-specified words for machine translation (Hokamp & Liu, 2017; Post & Vilar, 2018), and image captioning (Anderson et al., 2017). To the best of our knowledge, we are the first to exploit constrained generation for entity disambiguation, end-to-end entity linking, and query-based entity retrieval.

**Bi-encoders** Bi-encoders are successfully used in many applications, including entity linking. Building upon the work of Gillick et al. (2019), dense entity retrieval (BLINK; Wu et al., 2020) is a competitive system that uses both a dual-encoder approach in combination with approximated nearest neighbors to fast retrieve a set of candidates and a cross-encoder to re-rank them:

$$\log p(e|\mathcal{X};\theta) \propto f(\mathcal{X};\theta)^\top f(e;\theta) \ , \tag{4.1}$$

where $f$ is a neural network and $\theta$ its parameters. BLINK uses a classification loss with in-batch negatives to approximate the full cross-entropy loss:

$$\log p(e|\mathcal{X};\theta) = f(\mathcal{X};\theta)^\top f(e;\theta) - \log \sum_{e' \in \mathcal{E}} \exp f(\mathcal{X};\theta)^\top f(e';\theta) \tag{4.2}$$

$$\approx f(\mathcal{X};\theta)^\top f(e;\theta) - \log \sum_{e' \in \mathcal{B}} \exp f(\mathcal{X};\theta)^\top f(e';\theta) \ , \tag{4.3}$$

where $\mathcal{B}$ is a mini-batch of examples during training. BLINK encodes inputs using the last layer of BERT (Devlin et al., 2019b) output at the BOS token. It encodes the context with a special separator token to delimit the mention to disambiguate. It uses entities' meta information, such as titles and descriptions, to encode them.

Dense passage retrieval (DPR; Karpukhin et al., 2020) similarly employs bi-encoders to tackle the open-domain question-answering task (*i.e.,* the text collection is large and before applying any reading model it has to retrieve the top-k documents that may answer the query). DPR also uses inner-product as a scoring function and a cross-entropy loss. Additionally, DPR uses hard negatives mined from BM25 (Robertson, Zaragoza, et al., 2009) to learn a better vector space and thus boost the performance. Other models such as retrieval augmented language model pre-training (REALM; Guu et al., 2020) and retrieval augmented generation (RAG; Lewis et al., 2020b) used bi-encoders to augment language models. They optimize an LM objective while retrieving text to help the generation

learn end-to-end a model that reads, retrieves, and generates text. Those models have been successfully fine-tuned for open-domain question answering.

**Generative Retrieval** Nogueira et al. (2020) propose to use a sequence-to-sequence model to re-rank documents. Given a query and a document the model is trained to output the words "true" or "false" depending on whether the document is relevant or not. Differently from our approach for entity retrieval, it requires a limited list of candidate documents, obtained with BM25 for instance, in order to be computationally possible. Massarelli et al. (2019) and Petroni et al. (2020) explore the idea of using an autoregressive language model as a neural retriever, by exploiting the implicit knowledge stored in their parameters to generate relevant sentences given a query. While intriguing, such solutions still lag behind retrievers with an explicit knowledge access (*e.g.,* an explicit WIKIPEDIA index). The idea of using a generative model for entity disambiguation was proposed in Petroni et al. (2021) as they trained both BART and T5 in a seq2seq fashion on all KILT tasks (including ED). We expanded that intuition generalizing on multiple tasks (end-to-end EL and page-level retrieval) as well as introducing constrained decoding for an efficient and effective search.

## 4.3 Method

We address the retrieval problem with a sequence-to-sequence model that generates textual entity identifiers (*i.e.,* entity names). Concretely, GENRE ranks each $e \in \mathcal{E}$ by computing a score with an autoregressive formulation:

$$\text{score}(e|\mathcal{X};\theta) = p(\mathcal{Y}_e|\mathcal{X};\theta) = \prod_{i=1}^{n} p(y_i|y_{<i},\mathcal{X};\theta) \ , \tag{4.4}$$

where $\mathcal{X}$ is the input (context, mention and delimiters) sequence, $\mathcal{Y}_e = \{y_i\}_{i=1}^{n}$ is the sequence of $n$ tokens in the identifier of $e$, and $\theta$ the parameters of the model. We take advantage of fine-tuning the BART (Lewis et al., 2020a) pre-trained language model. We train GENRE using a standard seq2seq objective, *i.e.,* maximizing the log-probability of the output as a function of model parameters (Sutskever et al., 2011; Sutskever et al., 2014) and regularized with dropout (Srivastava et al., 2014a) and label smoothing (Szegedy et al., 2016). Concretely, we use the objective that is typically used for neural machine translation (NMT, Wu et al., 2016), that is maximizing $\log p(\mathcal{Y}|\mathcal{X};\theta)$ with respect to model's parameters $\theta$ which, due to the factorized formulation, can be calculated exactly. The model is already normalized without the need for any expensive normalization procedure or negative sampling.

### 4.3.1 Inference with Constrained Beam Search

Naturally, at test time, we could compute a score for every element in $\mathcal{E}$ and then sort them. Unfortunately, this might be prohibitively expensive when $\mathcal{E}$ is very large (*e.g.,* WIKIPEDIA has ≈6M entities). Hence, we exploit Beam Search (BS, Sutskever et al., 2014), an established approximate decoding strategies to efficiently navigate the search space. Instead of explicitly scoring all entities in $\mathcal{E}$, we search for the top-$k$ entities in $\mathcal{E}$ decoding from our model using BS with $k$ beams. Note that using BS implies that the time cost of our retriever does not depend on the size of $\mathcal{E}$, but only on the size of the beams and the average length of entity representations as we do autoregressive generation. The average length of entity representations is tractable (*e.g.,* WIKIPEDIA titles have 6 BPE tokens on average) and we follow standard NMT settings where $k$ is small (*e.g.,* 10).

Since we want to output only entities from $\mathcal{E}$ we cannot use traditional BS while decoding. Indeed, allowing to generate any token from the vocabulary at every decoding step might lead the model to generate output strings that are not valid identifiers. Hence, we resort to Constrained BS, forcing to only decode valid entity identifiers. BS only considers one step ahead during decoding so we can only constrain the generation of a single next token conditioned on the previous ones. Thus, we define our constrain in terms of a prefix tree $\mathcal{T}$ (known as *trie*); Cormen et al., 2009) where nodes are annotated with tokens from the vocabulary. For each node $t \in \mathcal{T}$, its children indicate all the allowed continuations from the prefix defined traversing the trie from the root to $t$.

See figure 4.2 for an exampled of a trie. When the number of allowed outputs is tractable (*e.g.,* generating a WIKIPEDIA title among ≈6M) the trie is relatively small it can be pre-computed and stored into memory (*e.g.,* constraining on WIKIPEDIA titles using the BART tokenizer produces a trie with $\approx 6M$ leaves, ≈17M internal nodes that occupied ≈600MB of disk space). We employed the constraints masking the log-probabilities of the invalid tokens and not their logits (*i.e.,* we do not re-normalize the probability over the vocabulary).[6]

### 4.3.2 Autoregressive End-to-End Entity Linking

We additionally extend our autoregressive framework to address end-to-end Entity Linking (EL) where, given a document, a system has to both detect entity mentions and link those mentions to their respective KB entities. In this setting, we train the model to predict the source input again but with annotated spans. We use a *Markup* annotation where spans boundaries are flagged with special tokens and accompanied by their corresponding entity identifiers.

Differently from a setting where the output space is relatively small (*e.g.,* a pre-defined set $\mathcal{E}$), the space of annotated outputs is exponentially large. Hence, it is intractable to pre-compute a trie for decoding, and we compute it dynamically

[6] We experimented with both versions and we find masking the log-probability more effective.

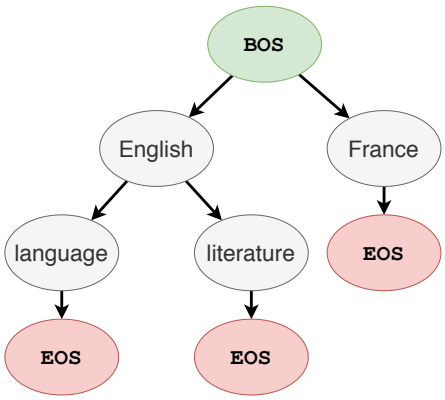


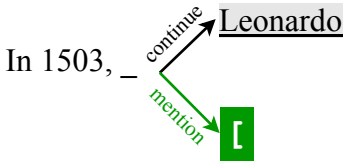

Figure 4.2: Example of prefix tree (trie) structure where the allowed entities identifiers are 'English language', 'English literature' and 'France'. Note that at the root there is the start-of-sequence token `SOS` and all leaves are end-of-sequence tokens `EOS`. Since more that one sequence has the same prefix (*i.e.,* 'English'), this end up being an internal node where branches are the possible continuations.

instead. In figure 4.3 we show an example. At each generation step, the decoder is either generating a mention span, generating a link to a mention, or continuing from the input source. When outside a mention/entity step the decoder has only two options: (i) to continue by copying the next token from the input source, or (ii) to generate the *start of mention* token (*i.e.,* '[') which makes the decoder enter the mention generating phase. While generating a mention, the decoder has either to continue with the next token in the input source or to generate the *end of mention* token (*i.e.,* ']') which makes the decoder enter the entity generating phase. Finally, when generating an entity, the decoder employs the entities trie such that it can only output a valid entity identifier as in Constrained Beam Search explained above.

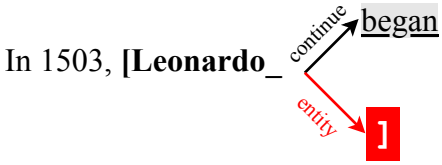
## 4.4 Experimental Settings

We extensively evaluate GENRE on more than 20 datasets across 3 tasks: Entity Disambiguation, end-to-end Entity Linking (EL), and page-level Document Retrieval. Here we describe the experimental settings and we discuss results in section 4.5. All experiments are in English.

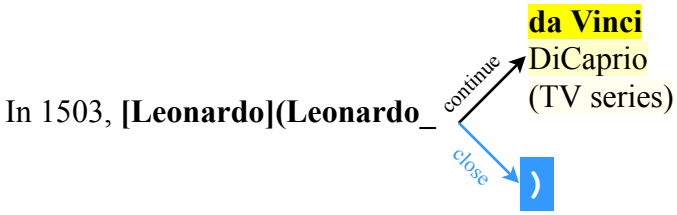
We implemented, trained, and evaluate our model using the `fariseq` library (Ott et al., 2019). We trained GENRE for every task using Adam (Kingma & Ba, 2015) with a learning rate $3e-5$ with a linear warm-up for 500 steps and then liner decay. The objective is sequence-to-sequence categorical cross-entropy loss with 0.1 of label smoothing. We used dropout (Srivastava et al., 2014b) probability of 0.1 and attention dropout of 0.1.




(a) Outside: we can either continue to generate the input or start a new mention.




(b) Inside a mention: we can either continue to generate the input or end the current mention.




(c) Inside an entity link: we can either generate from the entities prefix trie or close if the generated prefix is a valid entity.

Figure 4.3: Example of dynamically constrained *Markup* decoding for entity linking using "*In 1503, Leonardo began painting the Mona Lisa.*" as input. There are 3 cases: when we are outside a mention/entity (a), inside a mention generation step (b), and inside an entity link generation step (c). The model is supposed to output the input source annotating mentions and pointing them to the respective entities: "*In 1503,* ***[Leonardo]****(Leonardo da Vinci) began painting the* ***[Mona Lisa]****(Mona Lisa)*".

### 4.4.1 Entity Disambiguation (ED)

**Setting** We reproduce the setting of Le and Titov (2018) using the same candidate sets, *in-domain* and *out-of-domain* datasets, and evaluating using the *InKB* micro-$F_1$. Given a document $d$ (*e.g.,* a sentence) containing a set of entity mentions $\mathcal{M}_d = \{m_i\}_{i=1}^n$, a system $f$ has to assign, to each mention $m_i$, either a KB entity (*i.e.,* $f(m_i, d) \in \mathcal{E}$), or to predicts that there is no corresponding entry in the KB (*i.e.,* $f(m_i, d) = \text{NIL}$). Moreover, a restricted candidates set $C_{m_i} \subseteq \mathcal{E} \cup \{\text{NIL}\}$ for each mention $m_i$ is provided (*i.e.,* we can restrict the search to $\mathcal{C}_{m_i}$ instead of $\mathcal{E}$—typically $\mathcal{C}_{m_i}$ contains between tens to thousands of candidates which is still order of magnitude less than $|\mathcal{E}|$).

**Training** We train GENRE feeding each document where a single mention is flagged with two special start and end tokens and the target output is the textual representation of the corresponding entity. As large generative models benefit from large amount of data, we first pre-train GENRE on the BLINK data (Wu et al., 2020), *i.e.,* 9M unique triples document-mention-entity from WIKIPEDIA. We pre-trained for 200k steps and then we do model selection on the validation set. Afterward, we fine-tuned on AIDA-CoNLL dataset (Hoffart et al., 2011) without resetting the learning rate nor the optimizer statistics for 10k steps and we do model selection on the validation set. Following previous works (Yamada et al., 2016; Ganea & Hofmann, 2017; Le & Titov, 2018), we considered only mentions that have entities in the KB (*i.e.,* WIKIPEDIA). Training was done on 32 GPUs (with 32GB of memory) and it completed in ≈24h for a total of ≈32 GPU days.

**Inference** We evaluate on five test sets: MSNBC, AQUAINT, ACE2004, WNED-CWEB (CWEB) and WNED-WIKI (WIKI) (Gabrilovich et al., 2013; Guo & Barbosa, 2018). At test time, we decode using constrained beam search with a trie obtained using the provided candidate sets (*i.e.,* a subsets of $\mathcal{E}$). We use Constrained Beam Search with 10 beams, and maximum decoding steps of 15. We restrict the input sequence to be at most 384 tokens cutting the left, right, or both parts of the context around a mention. We normalize the log-probabilities by sequence length.

### 4.4.2 End-to-End Entity Linking (EL)

**Setting** For EL, we reproduce the setting of Kolitsas et al. (2018) using the same *in-domain* and *out-of-domain* datasets as well as evaluating the *InKB* micro-$F_1$ on the GERBIL benchmark platform (Röder et al., 2018). Given a document $d$ (*e.g.,* a sentence) a system $f$ has to return a set of tuples $f(d) = \{\langle m_i, e_i \rangle\}_{i=1}^{n}$ where each $m_i$ is a entity mentions (a span contained in $d$) and each $e_i \in \mathcal{E}$ is a corresponding entity in the KB. Following Kolitsas et al. (2018), we considered only mentions that have entities in the KB (*i.e.,* WIKIPEDIA) and we used their candidate sets with the additions of the table computed by Hoffart et al. (2011)—*i.e.,* we can restrict the search to $\mathcal{C}_{m_i}$ instead of $\mathcal{E}$ that is the same as for ED.

**Training** Similarly to the ED setting, we first pre-trained GENRE on all abstract sections from WIKIPEDIA[7] enriched by a string matching heuristic to solve co-references (*i.e.,* if there is a string that matches exactly with another hyperlink we also add it to the dataset as a mention/entity pairs) data for 200k steps. Then we do model selection on the validation set. Afterward, we fine-tuned on AIDA resetting the learning rate and the optimizer statistics for 10k steps and we do model selection on the validation set. Again, following previous

[7] It is based on the 2019/08/01 WIKIPEDIA dump pre-processed by Petroni et al. (2021).

works (Kolitsas et al., 2018), we considered only mentions that have entities in WIKIPEDIA. Training was done on 64 GPUs (with 32GB of memory) and it completed in ≈30h for a total of ≈80 GPU days.

**Inference** We evaluate on seven *out-of-domain* test sets: MSNBC, Derczynski (Der, Derczynski et al., 2015), KORE 50 (K50, Hoffart et al., 2012), N3-Reuters-128, N3-RSS-500 (R128 and R500, Röder et al., 2014), and OKE challenge 2015 and 2016 (OKE15 and OKE16, Nuzzolese et al., 2015). At test time, we use Constrained Beam Search with 6 beams, and a maximum decoding step of 384. When the input sequence is too long, we split the input into multiple chunks of equal size. We normalize the log-probabilities by sequence length.

### 4.4.3 Page-level Document Retrieval (DR)

**Setting** For this setting, we test GENRE on all the KILT benchmark tasks (Petroni et al., 2021). Here, whole WIKIPEDIA is used as the candidate set and we evaluate using averaged R-precision[8] (Beitzel et al., 2009). KILT consists of five tasks that use the same WIKIPEDIA dump as a knowledge source: fact checking with FEVER (Thorne et al., 2018); open domain question answering using Natural Questions (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018b), TriviaQA (Joshi et al., 2017), ELI5 (Fan et al., 2019); slot filling with T-REx (Elsahar et al., 2018), Zero Shot RE (Levy et al., 2017); entity disambiguation on AIDA CoNLL-YAGO, WNED-WIKI and WNED-CWEB; dialogue with Wizard of WIKIPEDIA (Dinan et al., 2019). Given a query $q$ (*e.g.,* a question) and a collection of documents $\mathcal{D}$ (in KILT are WIKIPEDIA pages), a system has to rank documents in $\mathcal{D}$ based on their relevance to $q$.

**Training** We train GENRE on BLINK (Wu et al., 2020) and all KILT data simultaneously with a single model.[9] We train for 200k steps and we do model selection on the validation set averaging the score across tasks. Training was done on 128 GPUs (with 32GB of memory) and it completed in ≈33h for a total of ≈176 GPU days.

**Inference** At test time, we use Constrained Beam Search with 10 beams. For the ED sub-task, we restrict the input sequence to be at most 384 tokens cutting the left, right, or both parts of the context around a mention. We normalize the log-probabilities by sequence length. As we are only interested in measuring

[8] R-Precision is the precision after $R$ items have been retrieved, where $R$ is the number of relevant items for the given input, *i.e.,* $|\{\text{retrieved items}\} \cap \{\text{relevant items}\}|/|\{\text{retrieved items}\}|$ where $|\{\text{retrieved items}\}|$ is not the same for every input in the dataset.

[9] Note that not all dataset available in KILT have a training set. Concretely, we train on FEVER, Natural Questions, HotpotQA, TriviaQA, T-REx, Zero Shot RE, AIDA CoNLL-YAGO, and Wizard of WIKIPEDIA.

retrieval, for all of those task and datasets we do not perform and evaluate the downstream performance of our system but only the *provenance* that is the WIKIPEDIA page needed as supporting evidence for the answer.

## 4.5 Results

Overall, GENRE achieves very competitive results in all of the three settings being the best performing system on average across all of them. See appendix A.1 for examples of inputs, ground truth and model predictions for all of the three tasks. In the following, we discuss how GENRE compares to state of the art systems as well as showing some quantitative analysis on its memory footprint, how it exploits the structured of the entity name space, and how it behaves on a *cold-start* scenario where new unseen entities are added to the KB (descriptions of those entities are unobserved).

### 4.5.1 Comparisons

**Comparing GENRE to state of the art systems** In ED the difference in average $F_1$ score between GENRE and the second best performing system is small (*i.e.,* +0.8) however, ED is an established task with more than a decade of research that benchmarked on those datasets. Indeed all systems reported in table 4.1 achieved high and similar results even if they were taken from three years back.

The improvements on EL are instead more evident. GENRE is the best in-domain system for AIDA while performing remarkably well also on the out-of-domain setting (*e.g.,* +13 $F_1$ points on Derczynski, and +4.7 on KORE50). Noticeably, in two datasets (OKE15 and OKE16) our model performs poorly. However, these datasets are annotated with coreference (pronouns and common nouns are linked to entities) while our model was not specifically trained for that. Conversely, most of the other systems, have a mention detection component in their pipelines that can be trained or biased to also solve these cases. We considered out of the aim of this study to additional train and evaluate on coreference and we leave it for future work.

On page-level DR, the superiority of GENRE is remarkable. Our model is the best performing system across all 5 KILT tasks and all datasets except on Natural Questions where it is the second best. We achieve +13.7 R-precision points on average with respect to the best performing baseline. In table 4.2 we compare GENRE against all methods reported in the public leaderboard:[10] DPR (Karpukhin et al., 2020), DPR+BERT (Devlin et al., 2019a), DPR+BART,

[10] https://evalai.cloudcv.org/web/challenges/challenge-page/689 accessed on 16-09-2020. We additionally report subsequent published works that have public score we accessed on 11-07-2022.

TF-IDF (Leskovec et al., 2014), RAG (Lewis et al., 2020b), and BLINK+flair (Wu et al., 2020; Akbik et al., 2019). No model except ours was trained on the entire KILT dataset at the same time. A RAG model was trained for every single task as well as for DPR+BERT. Note that this gives and advantage to RAG and DPR+BERT to specialize on single tasks where we have only a single model to solve all of them which still performs better. We speculate that multi-task training could have helped since the all tasks share a common objective to retrieve entities. Both DPR and BLINK+flair were not trained specifically on KILT. However, DPR was trained using several QA datasets which include Natural Question and TriviaQA.

**Memory Footprint** GENRE is not only performing better than other state of the art models on DR but it has a significant reduction of memory footprint (disk space). In figure 4.4 we compare the number of model/index parameter against DPR, RAG, and BLINK. GENRE uses an order of magnitude less parameters (millions instead of billions) to store the entity index because it just has to use a prefix tree of the entity names as opposed to a dense vector for each entity. Concretely, GENRE occupied 14 times less memory than BLINK and 34 times less memory than DPR.

### 4.5.2 Ablations

**Ablation study on ED** In table 4.1, GENRE only AIDA or BLINK data indicates the ablation for which we only train on one of the two datasets (*i.e.,* only fine-tuning). GENRE (full) is also used with constrained decoding (see section 4.3) and in combination with a candidate set (as provided by Le & Titov, 2018). GENRE without candidate set denotes ablating the provided (and small) candidate set and therefore using all the entities in the KB (in our case Wikipedia) as candidates. GENRE without constraints indicates ablating constrained decoding which implies no use of the provided candidates set but also unconstrained generation (*i.e.,* the model may generate entity names that are not in the KB). Eventually, using constrained generation and exploiting the candidate sets proved useful. Training only on AIDA data is insufficient to get high $F_1$ (but AIDA is quite small compared to the 9M datapoints of BLINK data).

**Ablation study on DR** Table 4.2 also extends a comparison with baselines with ablation results (*i.e.,* training GENRE on the numerical identifiers). The purpose of the experiment is to see whether GENRE benefits from the entity names to be meaningful as well as compositional. Numerical IDs do not have that property. In both cases, the model uses its memorizing capabilities but when using IDs the performance is significantly low. Indeed, with IDs the model has no way to generalize nor to use *implicit knowledge* acquired during the unsupervised

| | In-domain | Out-of-domain | | | | | |
|---|---|---|---|---|---|---|---|
| **Method** | **AIDA** | **MSNBC** | **AQUAINT** | **ACE2004** | **CWEB** | **WIKI*** | **Avg.** |
| Yamada et al. (2016) | 91.5 | - | - | - | - | - | - |
| Ganea and Hofmann (2017) | 92.2 | 93.7 | 88.5 | 88.5 | 77.9 | 77.5 | 86.4 |
| Guo and Barbosa (2018) | 89.0 | 92.0 | 87.0 | 88.0 | 77.0 | 84.5 | 86.2 |
| Yang et al. (2018a) | **95.9** | 92.6 | 89.9 | 88.5 | **81.8** | 79.2 | 88.0 |
| Raiman and Raiman (2018) | 94.9 | - | - | - | - | - | - |
| Shahbazi et al. (2019) | 93.5 | 92.3 | 90.1 | 88.7 | 78.4 | 79.8 | 87.1 |
| Yang et al. (2019) | 93.7 | 93.8 | 88.2 | 90.1 | 75.6 | 78.8 | 86.7 |
| Le and Titov (2019) | 89.6 | 92.2 | **90.7** | 88.1 | 78.2 | 81.7 | 86.8 |
| Fang et al. (2019) | 94.3 | 92.8 | 87.5 | **91.2** | 78.5 | 82.8 | 87.9 |
| BLINK w/o candidate set** | 79.6 | 80.0 | 80.3 | 82.5 | 64.2 | 75.5 | 77.0 |
| Chen et al. (2020) | 93.5 | - | - | - | - | - | - |
| Mulang' et al. (2020) | 94.9 | - | - | - | - | - | - |
| **GENRE** | 93.3 | **94.3** | 89.9 | 90.1 | 77.3 | **87.4** | **88.8** |
| Ablations | | | | | | | |
| GENRE only AIDA data | 88.6 | 88.1 | 77.1 | 82.3 | 71.9 | 71.7 | 80.0 |
| GENRE only BLINK data | 89.3 | 93.3 | 90.9 | 91.1 | 76.0 | 87.9 | 88.1 |
| GENRE w/o candidate set | 91.2 | 86.9 | 87.2 | 87.5 | 71.1 | 86.4 | 85.1 |
| GENRE w/o constraints | 86.4 | 80.0 | 81.7 | 82.1 | 66.0 | 81.1 | 79.6 |
| Subsequent work | | | | | | | |
| Barba et al. (2022) | 92.6 | 94.7 | 91.6 | 91.8 | 77.7 | 88.8 | 89.5 |
| Aghajanyan et al. (2022) | 94.8 | 94.8 | 91.1 | 91.4 | 78.4 | 88.7 | 89.8 |

Table 4.1: Micro $F_1$ (InKB) on the in-domain test set and five out-of-domain test sets for the named entity disambiguation task. **Bold** indicates best model and underline indicates second best (not for ablations). *WIKI is usually considered out-of-domain but note that all methods use a part of WIKIPEDIA to train. **results taken from https://github.com/facebookresearch/BLINK and normalized to accommodate entities not in KB.

| Model | Fact Check. FEV | Entity Disambiguation AY2 | WnWi | WnCw | Slot Filling T-REx | zsRE | Open Domain QA NQ | HoPo | TQA | ELI5 | Dial. WoW | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DPR + BERT | 72.9 | - | - | - | - | 40.1 | **60.7** | 25.0 | 43.4 | - | - | - |
| DPR | 55.3 | 1.8 | 0.3 | 0.5 | 13.3 | 28.9 | 54.3 | 25.0 | 44.5 | 10.7 | 25.5 | 23.6 |
| tf-idf | 50.9 | 3.7 | 0.24 | 2.1 | 44.7 | 60.8 | 28.1 | 34.1 | 46.4 | 13.7 | 49.0 | 30.5 |
| DPR + BART | 55.3 | 75.5 | 45.2 | 46.9 | 13.3 | 28.9 | 54.3 | 25.0 | 44.4 | 10.7 | 25.4 | 38.6 |
| RAG | 61.9 | 72.6 | 48.1 | 47.6 | 28.7 | 53.7 | 59.5 | 30.6 | 48.7 | 11.0 | 57.8 | 47.3 |
| BLINK + flair | 63.7 | 81.5 | 80.2 | 68.8 | 59.6 | 78.8 | 24.5 | 46.1 | 65.6 | 9.3 | 38.2 | 56.0 |
| **GENRE full** | **83.6** | **89.9** | **87.4** | **71.2** | **79.4** | **95.8** | 60.3 | **51.3** | **69.2** | **15.8** | **62.9** | **69.7** |
| Ablations | | | | | | | | | | | | |
| GENRE only BLINK *IDs* | 1.8 | 65.0 | 63.5 | 58.6 | 0.1 | 0.2 | 0.4 | 0.3 | 5.4 | 0.3 | 13.3 | 19.0 |
| GENRE only BLINK data | 28.1 | 82.5 | 88.1 | 69.9 | 44.8 | 66.1 | 15.0 | 16.4 | 25.6 | 6.8 | 38.7 | 43.8 |
| GENRE only DPR data | 70.8 | 9.7 | 1.9 | 7.3 | 60.0 | 79.7 | 58.3 | 40.3 | 69.6 | 13.2 | 52.6 | 42.1 |
| GENRE w/o constraints | 78.9 | 87.2 | 83.2 | 36.5 | 74.4 | 93.6 | 53.3 | 45.2 | 63.7 | 14.3 | 62.7 | 63.0 |
| Subsequent Work | | | | | | | | | | | | |
| multi-DPR + BART | 74.5 | 26.5 | 4.9 | 1.9 | 69.5 | 80.9 | 59.4 | 42.9 | 61.5 | 15.5 | 41.1 | 43.6 |
| SEAL | 81.5 | - | - | - | 67.8 | 98.0 | 63.2 | 58.8 | 68.4 | - | - | - |

Table 4.2: R-Precision for page-level retrieval on KILT test data–baselines are taken from Petroni et al. (2021). **Bold** indicates the best model and underline indicates the second best. Our GENRE outperforms previous results in almost all datasets. For the ablations we indicated what datasets we used for training and the test datasets that are similar are highlighted in blue. GENRE only BLINK *IDs* denotes training on BLINK (Wu et al., 2020) data where instead of using the textual entity representation as target we used a numerical ID. multi-DPR (Maillard et al., 2021) is a DPR model (Karpukhin et al., 2020) trained on multiple KILT tasks. SEAL denotes the system by Bevilacqua et al. (2022).

| | In-domain | Out-of-domain | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Method** | **AIDA** | **MSNBC** | **Der** | **K50** | **R128** | **R500** | **OKE15*** | **OKE16*** | **Avg.** |
| Hoffart et al. (2011) | 72.8 | 65.1 | 32.6 | 55.4 | 46.4 | **42.4** | **63.1** | 0.0 | 47.2 |
| Daiber et al. (2013) | 57.8 | - | - | - | - | - | - | - | - |
| Steinmetz and Sack (2013) | 42.3 | 30.9 | 26.5 | 46.8 | 18.1 | 20.5 | 46.2 | 46.4 | 34.7 |
| Moro et al. (2014) | 48.5 | 39.7 | 29.8 | 55.9 | 23.0 | 29.1 | 41.9 | 37.7 | 38.2 |
| Piccinno and Ferragina (2014) | 73.0 | - | - | - | - | - | - | - | - |
| Kolitsas et al. (2018) | 82.4 | 72.4 | 34.1 | 35.2 | **50.3** | 38.2 | 61.9 | 52.7 | 53.4 |
| Peters et al. (2019) | 73.7 | - | - | - | - | - | - | - | - |
| Févry et al. (2020a) | 76.7 | - | - | - | - | - | - | - | - |
| Broscheit (2019) | 79.3 | - | - | - | - | - | - | - | - |
| Martins et al. (2019) | 81.9 | - | - | - | - | - | - | - | - |
| van Hulst et al. (2020)† | 80.5 | 72.4 | 41.1 | 50.7 | 49.9 | 35.0 | **63.1** | **58.3** | 56.4 |
| **GENRE** | **83.7** | **73.7** | **54.1** | **60.7** | 46.7 | 40.3 | 56.1 | 50.0 | **58.2** |
| Subsequent work | | | | | | | | | |
| Kannan Ravi et al. (2021) | 83.1 | - | - | - | - | - | - | - | |
| De Cao et al. (2021c)‡ | 85.5 | - | - | - | - | - | - | - | |
| Zhang et al. (2022b) | 85.8 | 72.1 | 52.9 | 64.5 | 54.1 | 41.9 | 61.1 | 51.3 | 60.5 |
| Mrini et al. (2022) | 85.7 | - | - | - | - | - | - | - | |

Table 4.3: Micro $F_1$ (InKB) on the in-domain test set and four out-of-domain test sets for the entity linking task. **Bold** indicates best model and underline indicates second best. *annotated with coreference (note that we do not train/evaluate our model to link pronouns and common nouns). †results from the WIKIPEDIA 2019 setting as opposed to the 2014 setting (older dump and fewer entities). ‡ discussed in chapter 6.

| Model | Memory | Parameters | Index |
|---|---|---|---|
| DPR (Karpukhin et al., 2020) | 70.9GB | 220M | 15B |
| RAG (Lewis et al., 2020b) | 40.4GB | 626M | 15B |
| BLINK (Wu et al., 2020) | 30.1GB | 680M | 6B |
| **GENRE** | **2.1GB** | **406M** | **17M** |

Table 4.4: Comparison between retrieval models on memory (disk space) footprint and number of model/index parameters.

pre-training. We also ablate the training data. DPR data corresponds to training only on Natural Questions (NQ) and TriviaQA (TQA) as DPR was trained only for QA tasks on those datasets and two extra ones. Note that training on BLINK data corresponds to only training for entity disambiguation. However, every other task share similarities with entity disambiguation and thus the model is also capable to address the other tasks with non-zero performance. For the ablations, underlined cells indicate what are the results on the respective task on which a model was trained for (*i.e.,* GENRE *only BLINK data* was trained only for ED where GENRE *only DPR data* was trained only for QA). The ablation on data suggests that it is beneficial to train on all tasks simultaneously. GENRE without constraints indicates ablating constrained decoding which implies unconstrained generation (*i.e.,* the model may generate entity names that are not in the KB).

### 4.5.3 Analysis

**Exploiting the Structured Name Space** We investigated some properties of GENRE, comparing two variants of our model to BLINK on the ED task (using WNED-KILT validation set): one trained to generate entity names and another to generate numerical identifiers (IDs). All models are trained on the same data and we report results in table 4.5. When there is an exact match between a mention and its entity name, both BLINK and GENRE almost always make an accurate prediction. Different is the case of partial and no match: GENRE performance is much higher suggesting that our model uses the context more effectively, as the autoregressive formulation allows to cross-encode mention context and entity candidates directly capturing fine-grained interactions between the two. Moreover, when we switch to predicting IDs, the performance drops drastically (-20.3 points on average) indicating that it is important that entity names are meaningful, structured and compositional (as they are in Wikipedia) conversely to atomic IDs. Surprisingly, when there is no overlap between a mention-entity pair, performance are still relatively high by using IDs. This suggests that the model is good at memorizing and recalling identifiers even if numeric.

| **Type** (support) | **BLINK** | **GENRE** | **IDs*** |
|---|---|---|---|
| Exact match (1543) | 97.8 | 96.6 | 76.0 |
| Partial match (1531) | 70.7 | 86.9 | 63.8 |
| No match (322) | 49.4 | 59.9 | 55.0 |
| Total (3396) | 81.0 | 88.8 | 68.5 |

Table 4.5: Different types of matches between mentions and their entity names on the WNED-KILT. *indicates GENRE trained on numerical identifiers.

**Entity frequency** The performance of a model naturally depends on how many times entities appear in the training data. We show the data distribution of the mention-entity frequency in figure 4.4. Most of the pairs appears in Wikipedia (10931 / 13354) where 2423 do not (first bin). The average accuracy is 82.5% but noticeable it is higher for mention-entity pairs that are more frequent (right side of the plot). The accuracy for pairs that do not appear in Wikipedia is substantially lower than the average suggesting that those are harder cases (the very end tail of the distribution). The degradation in performance is minimal indicating that our model is good at predicting rare entities.

**Incoming links frequency** We also show the data distribution of the number of incoming links (see figure 4.5). Intuitively, a page/entity with few incoming links has been observed less than highly connected pages/entities. Indeed, for pages/entities never linked (first bin on the left) the average accuracy is 20% lower than the global average (78.6%). However, for pages/entities linked at least once it is above the global average. This indicates that GENRE seems effective on linking rare entities.

**Entity name tokens length frequency** We show the data distribution of title entity name tokens length in figure 4.6. Most of the titles have less than 15 BPE tokens while the mode of the distribution is 5. Here GENRE has an average accuracy of 78.6% but it is higher for short titles (*e.g.,* $<10$) and it is lower for long titles (*e.g.,* $\geq 10$). Degradation in performance does not directly follow the data distribution of the token lengths. Indeed, even if long titles are rare performance is not heavily affected (*e.g.,* for length $>15$).

**Cold-start** We manually collect 50 Wikipedia articles that were created in 2020[11] to simulate a *cold-start* setting where new entities are added to the KB and the only entity information available is their names. To create ED instances we resort to hyperlinks pointing to those entities in other Wikipedia articles. 19

[11]Note that both pre-training and fine-tuning use dumps from 2019.

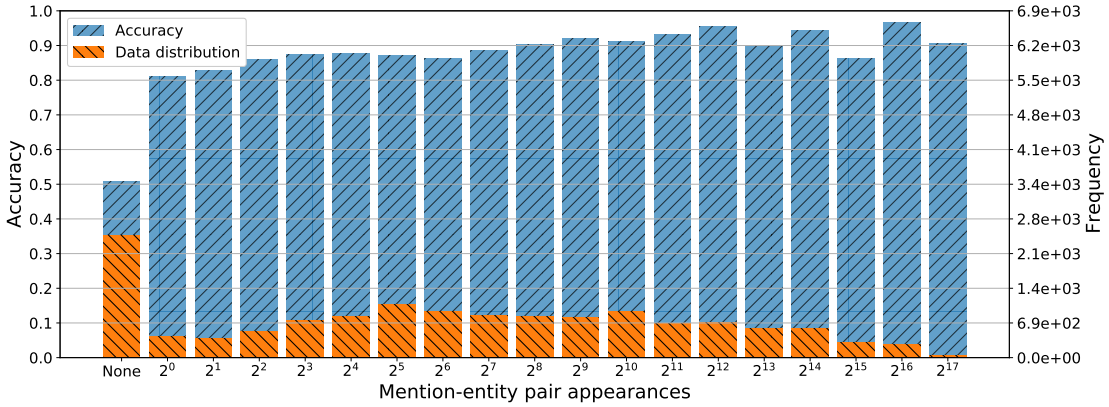



Figure 4.4: Accuracy per mention-entity pair frequency (in WIKIPEDIA) on the validation sets of all Entity Disambiguation tasks in KILT.

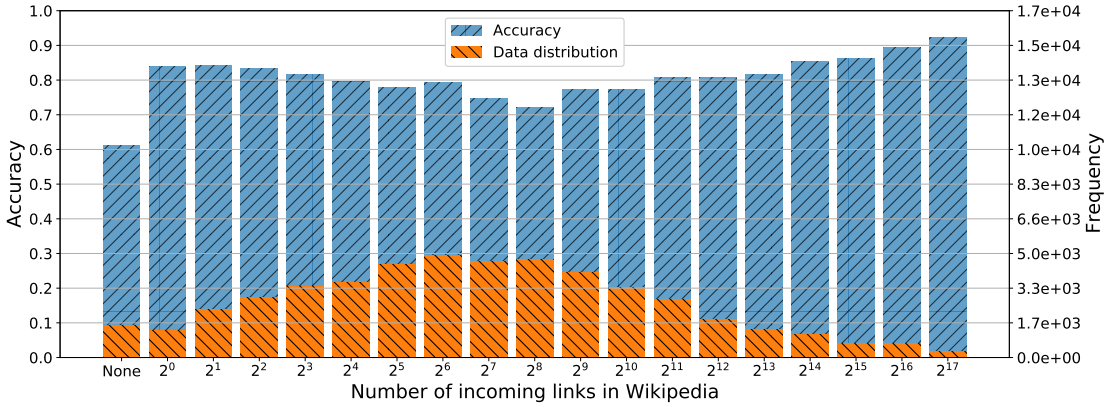



Figure 4.5: Accuracy per number of incoming links in WIKIPEDIA on the validation sets of all KILT datasets except ELI5 (as it is fundamentally different from the others).

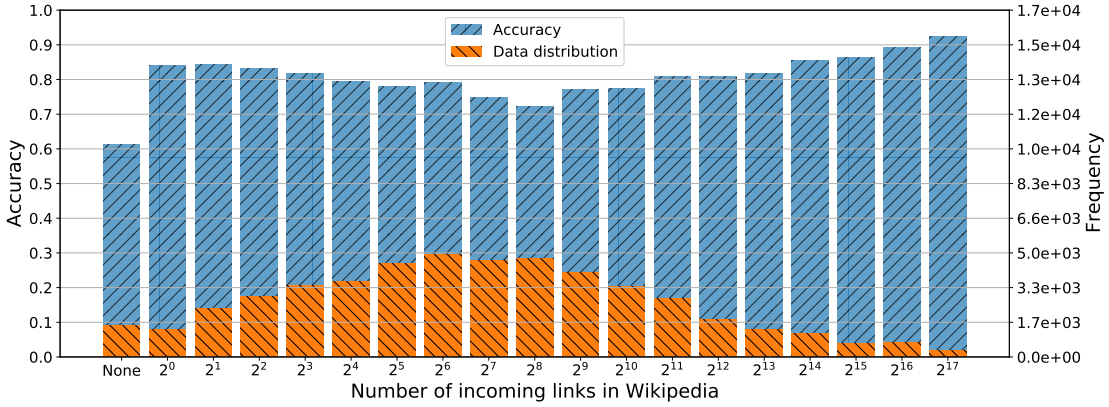



Figure 4.6: Accuracy per number of BPE tokens of the WIKIPEDIA title to generate on the validation sets of all KILT datasets except ELI5 (as it is fundamentally different from the others).

| | **Seen** | **Unseen** | **Total** |
|---|---|---|---|
| Exact match | 87.48 (751) | 70.36 (2227) | 74.68 (2978) |
| Partial match | 56.39 (1566) | 61.47 (4838) | 60.23 (6404) |
| No match | 41.46 (205) | 45.04 (413) | 43.85 (618) |
| **Total** | 64.43 (2522) | 63.21 (7478) | 63.52 (10k) |

Table 4.6: Evaluation of GENRE on WikilinksNED Unseen-Mentions data (Onoe & Durrett, 2020). We train on the provided train set and we report accuracy scores (*i.e.,* precision at 1) alongside with the number of supporting datapoints. We report scores splitting the test set in seen and unseen entities as well as in three different matchings between a mention and its gold entity.

out of 50 mentions have an exact match with their respective entity names and all of them were correctly classified by GENRE. In combination with the results from table 4.5 we can conclude that GENRE has a bias on exactly copying the mention, and this helps on unseen data. GENRE also correctly classified 14/31 of the remaining mentions (45.2%). This demonstrates the ability of our solution to be applied in scenarios where entity metadata is unavailable (apart his name), a setting where, to the best of our knowledge, no existing system is capable to operate.

We additionally test how GENRE performs on unseen mention-entity pairs on WikilinksNED Unseen-Mentions data (Onoe & Durrett, 2020) and we report all results in table 4.6. Surprisingly, GENRE performs almost the same for seen and unseen entity pairs (64.4 vs 63.2 accuracy) However, in the Onoe and Durrett (2020) setting we cannot guarantee entity descriptions have not been seen by BART during pre-training (given his training data contains Wikipedia).

## 4.6 Subsequent Work

A natural extension of the work presented in this chapter is a multilingual version (mGENRE; De Cao et al., 2022). mGENRE allows doing entity linking in more than 100 languages as well as considering languages as a latent variable allowing marginalizing the predictions on all of them. mGENRE is presented in chapter 5. Another work of ours addresses the high computational cost of the end-to-end linking task due to a complex (deep) decoder and the non-parallelizable decoding that scales with the source sequence length (De Cao et al., 2021c). In chapter 6, we propose a very efficient approach that parallelizes autoregressive linking across all potential mentions and relies on a shallow and efficient decoder. Although not part of this thesis, the author also contributed to generative information extraction (GenIE; Josifoski et al., 2022) which extends our work to closed information extraction. Closed information extraction is the problem of extracting

an exhaustive set of ⟨subject, relation, object⟩ triplets from a text snippet that are consistent with a predefined set of entities and relations in a KB. GenIE is a sequence-to-sequence model that exploits the knowledge from a pre-trained transformer by autoregressively generating relations and entities in textual form. GenIE employs a bi-level constrained generation strategy that produces only valid sets of triplets consistent with the predefined KB.

Our work also inspired extensions on other domains, which led to improvements over standard methods. Generative multi-hop retrieval (GMR; Lee et al., 2022) uses a sequence-to-sequence model which iteratively predicts the next paragraph needed as evidence for answering a query. Authors show comparable or higher performance than state-of-the-art bi-encoders while demonstrating better memory and storage footprint. They also show better performance than GENRE in multi-hop setting since since GMR specifically targets that while we do not. Generative evidence retrieval for fact verification (GERE; Chen et al., 2022) employs and extends a version of GENRE for fact-checking. GENE retrieves documents in the same way GENRE does but it additionally retrieves evidence in a generative fashion (*i.e.,* generating the document titles as well as evidence sentence identifiers). Authors evaluated GENE on the complete fact verification task of FEVER (differently, we only measured recall at document level where FEVER requires a sentence level evidence and a label prediction) showing significant improvements over baselines with both time- and memory efficiency.

Generalizations of our framework have also been proposed. Differentiable search index (DSI; Tay et al., 2022b) is a system that uses hierarchical clustering on contextualized embeddings to create identifiers for arbitrary spans of text. The main advantage of DSI is that it does not require title annotation of documents or paragraphs, and therefore, it can be generally applied outside of Wikipedia. Furthermore, it treats the selection of identifiers as a fully unsupervised pre-processing step. DSI is applied to moderate-sized corpora (up to 320k documents), showing that it is a promising direction for further exploration. Search engines with autoregressive LMs (SEAL; Bevilacqua et al., 2022) also avoids the use of titles. SEAL uses all n-grams in a text span as its possible identifiers. SEAL uses compressed full-text substring index (an FM-Index; Ferragina & Manzini, 2000) to efficiently store all the corpus/index. SEAL is trained to generate multiple n-grams conditioning on a query while applying some re-weighing to avoid repetitions and to re-balance infrequent n-grams. See table 4.2 for results on KILT.

Additional relevant subsequent work includes CM3 (Aghajanyan et al., 2022): masked autoregressive generative language and vision model. CM3 is trained with denoising over a large corpus of structured multi-modal documents which includes HTML tags and the whole Wikipedia. Training is self-supervised, and at test time, when it is prompted with the HTML link tag, it can generate correct Wikipedia identifiers. When fine-tuned for entity linking specifically, CM3 outperforms all known models to the best of our knowledge (see table 4.1). Entity names can also be used in an extractive fashion. Extractive entity disambiguation

(ExtEnD; Barba et al., 2022) proposed to concatenate to the input text all the entity names of the candidates and use an extractive model to select the span of the correct label (results reported in table 4.1). Finally, Mrini et al. (2022) improved autoregressive entity linking by adding auxiliary tasks to obtain a better model. The authors trained to re-rank the generated samples at inference time and mention detection. We report relevant results from subsequent work at the bottom of figures 4.1, 4.2, and 4.2.

## 4.7 Conclusions

In this chapter, we propose GENRE, a novel paradigm to addresses entity retrieval: generate entity names autoregressively. Entity names have several properties that might help (even humans) retrieve them, including a compositional structure and a predictable interaction with the context. The autoregressive formulation allows us to directly capture some of these properties, leading to several advantages with respect to current solutions, including an efficient way to cross encode mention context and entity candidates, a much smaller memory footprint, and the ability to compute an exact cross entropy loss without the need to subsample negative data. We empirically show that these characteristics, combined with constrained decoding strategies, led to state-of-the-art performance on a plethora of entity retrieval datasets, spanning entity disambiguation, end-to-end entity linking, and page-level document retrieval, while resulting in systems with a remarkably contained memory footprint, a space reduction by a factor of twenty on average. We additionally demonstrate that new entities can be effectively considered in our system by simply appending their unambiguous name to the candidate set.

# Chapter 5

# Multilingual Autoregressive Entity Linking

**Chapter Highlights**

In this chapter, we present mGENRE, a sequence-to-sequence system for the Multilingual Entity Linking (MEL) problem—the task of resolving language-specific mentions to a multilingual Knowledge Base (KB). mGENRE is a multilingual evolution of GENRE presented in chapter 4. For a mention in a given language, mGENRE predicts the name of the target entity left-to-right, token-by-token in an autoregressive fashion. The autoregressive formulation allows us to effectively cross-encode mention string and entity names to capture more interactions than the standard dot product between mention and entity vectors. It also enables fast search within a large KB even for mentions that do not appear in mention tables and with no need for large-scale vector indices. While prior MEL works use a single representation for each entity, we match against entity names of as many languages as possible, which allows exploiting language connections between source input and target name. Moreover, in a zero-shot setting on languages with no training data at all, mGENRE treats the target language as a latent variable that is marginalized at prediction time. This leads to over 50% improvements in average accuracy. We show the efficacy of our approach through extensive evaluation including experiments on three popular MEL benchmarks where we establish new state-of-the-art results.[1]

[1] Source code available at https://github.com/facebookresearch/GENRE

## 5.1 Introduction

The multilingual version of the EL problem has been for a long time tied to a purely cross-lingual formulation (XEL, McNamee et al., 2011; Ji et al., 2015), where mentions expressed in one language are linked to a KB expressed in another (typically English). Recently, Botha et al. (2020) made a step towards a more inherently multilingual formulation by defining a language-agnostic KB, obtained by grouping language-specific descriptors per entity. Such a formulation has the power of considering entities that do not have an English descriptor (*e.g.,* a WIKIPEDIA article in English) but have one in some other languages.

A common design choice to most current solutions, regardless of the specific formulation, is to provide a unified entity representation, either by collating multilingual descriptors in a single vector or by defining a canonical language. For the common bi-encoder approach (Wu et al., 2020; Botha et al., 2020), this might be optimal. However, in the recently proposed GENRE model (see chapter 4; De Cao et al., 2021a), an autoregressive formulation to the EL problem leading to stronger performance and considerably smaller memory footprints than bi-encoder approaches on monolingual benchmarks, the representations to match against are entity names (*i.e.,* strings) and it's unclear how to extend those beyond a monolingual setting.

In this context, we find that maintaining as much language information as possible, hence providing multiple representations per entity (*i.e.,* one for each available language), helps due to the connections between source language and entity names in different languages. We additionally find that using all available languages as targets and aggregating over the possible choices is an effective way to deal with a zero-shot setting where no training data is available for the source language.

Concretely, in this chapter, we present mGENRE, the first multilingual EL system that exploits a sequence-to-sequence architecture to generate entity names in more than 100 languages left to right, token-by-token in an autoregressive fashion and conditioned on the context (see figure 5.1 for an outline of our system). While prior works use a single representation for each entity, we maintain entity names for as many languages as possible, which allows exploiting language connections between source input and target name. To summarize, this chapter makes the following contributions:

- we consider, in the catalog of entity names, all languages for each entry in the KB. In particular, we use WIKIDATA as target KB and consider all available WIKIPEDIA titles — each entity is represented with multiple names, one for each available language. Storing the multilingual names index is feasible and cheap (*i.e.,* 2.2GB for ≈89M names);

- we design a novel objective function that marginalizes over all languages to perform a prediction. This approach is particularly effective in dealing with

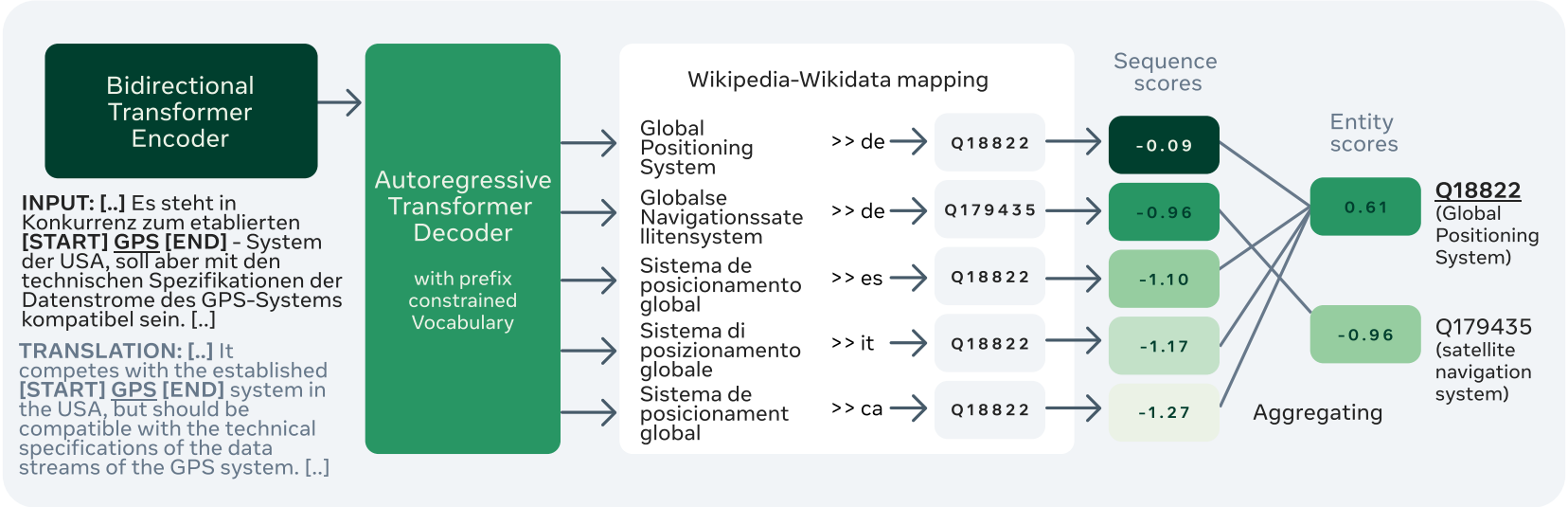



Figure 5.1: mGENRE: the input is text where an entity mention is signaled with special separator tokens and the model outputs predictions for the entity identifier. We use an autoregressive transformer decoder to generate language IDs as well as entity names (*i.e.*, WIKIPEDIA titles). The combination of language ID and a entity name uniquely identify a WIKIDATA ID (with a N-to-1 mapping). We use Beam Search for efficient inference and we marginalize the probability scores for different languages to score entities. This example is a real output from our system and scores values are length normalized log-probabilities.

languages not seen during fine-tuning (≈50% improvements);

- we establish new state-of-the-art performance for the Mewsli-9 (Botha et al., 2020), TR2016$^{\text{hard}}$ (Tsai & Roth, 2016) and TAC-KBP2015 (Ji et al., 2015) MEL datasets;
- we present extensive analysis of modeling choices, including the usage of candidates from a mention table, frequency-bucketed evaluation, and performance on a held out set including low-resource languages.

## 5.2 Background

We first introduce Multilingual Entity Linking in section 5.2.1 highlighting its difference with monolingual and cross-lingual linking. We address the MEL problem with a sequence-to-sequence model that generates textual entity identifiers (*i.e.,* entity names). Our formulation generalizes the GENRE model by De Cao et al. (2021a) to a multilingual setting (mGENRE) discussed in chapter 4.

### 5.2.1 Task Definition

*Multilingual Entity Linking* (MEL, Botha et al., 2020) is the task of linking a given entity mention $m$ in a given context $c$ of language $l \in \mathcal{L}_C$ to the corresponding entity $e \in \mathcal{E}$ in a multilingual Knowledge Base (KB). See figure 5.1 for an example: there are textual inputs with entity mentions (in bold) and we ask the model to predict the corresponding entities in the KB. A language-agnostic KB includes an entity descriptor (at least the name) of each entity in one or more languages. Note that there is no guarantee that an entity descriptor matching the context language is always available. We assume that descriptors in multiple languages for the same entity are mapped to a unique entry in the KB (*e.g.,* as in WIKIDATA). and that each $e \in \mathcal{E}$ has a descriptor in at least a language. Concretely, in this work, we use WIKIDATA (Vrandečić, 2012) as our KB. Each item lists a set of WIKIPEDIA pages in multiple languages linked to it and in any given language each page has a unique name (*i.e.,* its title).

The MEL formulation is a generalization of both monolingual Entity Linking EL and cross-lingual EL (XEL, McNamee et al., 2011; Ji et al., 2015). The monolingual EL formulation considers a KB where each entity descriptor is expressed in the context language — mention and KB language always match, descriptors in other languages are discarded. One problem of this formulation is that the KB might miss several entries for languages with limited coverage of entity descriptors. The XEL formulation tries to mitigate this problem by considering the language with the highest descriptors coverage as canonical (typically English) — mentions in multiple languages are mapped to a single canonical language. Therefore, both the MEL and XEL formulations exploit inter-language links to

identify entities in other languages. However, given that XEL requires the target KB to be monolingual it might still miss several entries in the KB. For instance, Botha et al. (2020) reported that ≈25% of hyperlinks in the Japanese WIKINEWS do not point to a page that have a corresponding one in English.

In this work we assume that each entity descriptor contains a name that concisely describes an entity. In particular, we consider WIKIPEDIA titles (in multiple languages) as entity names. Note that such entity names might not be available for other KBs. We consider the definition of meaningful entity names when not available an interesting future research direction. Finally, see chapter 4 for references on autoregressive entity linking via generation and ranking with constrained beam search.

### 5.2.2 Related Work

The most related works to ours are De Cao et al. (2021a), that proposed to use an autoregressive language model for monolingual EL (see chapter 4), and Botha et al. (2020) that proposes to extend the cross-lingual EL task to multilingual EL with a language-agnostic KB. GENRE was applied not only to EL but also for joint mention detection and entity linking (still with an autoregressive formulation) as well as to page-level document retrieval for fact-checking, open-domain question answering, slot filling, and dialog (Petroni et al., 2021). Botha et al.'s (2020) Model $F^+$ is a *bi-encoder* model: it is based on two BERT-based (Devlin et al., 2019a) encoders that outputs vector representations for contet and entities. Similar to Wu et al. (2020) they rank entities with a dot-product between these representations. Model $F^+$ uses the description of entities as input to the entity encoder and title, document and mention (separated with special tokens) as inputs to the context encoder. Bi-encoders solutions may be memory inefficient since they require to keep in memory big matrices of embeddings, although memory-efficient dense retrieval has recently received attention (Izacard et al., 2020; Min et al., 2021; Lewis et al., 2021).

Another widely explored line of work is Cross-Language Entity Linking (XEL; McNamee et al., 2011; Cheng & Roth, 2013). XEL considers contexts in different languages while mapping mentions to entities in a monolingual KB (*e.g.,* English WIKIPEDIA). Tsai and Roth (2016) used alignments between languages to train multilingual entity embeddings. They used candidate selection and then they re-rank them with an SVM using these embeddings as well as a set of features (based on the multilingual title, mention, and context tokens). Sil et al. (2018) explored the use of more sophisticated neural models for XEL as well as Upadhyay et al. (2018) who jointly modeled type information to boost performance. Zhou et al. (2019) propose improvements to both entity candidate generation and disambiguation to make better use of the limited data in low-resource scenarios. Note that in this work we focus on *multilingual* EL, not cross-lingual. XEL is limiting to a monolingual KB (usually English), where MEL is more general

since it can link to entities that might not be necessary represented in the target monolingual KB but in any of the available languages.

## 5.3 Method

To extend GENRE to a multilingual setting, we need to define what are the unique identifiers of all entities in a language-agnostic fashion. This is not trivial since we rely on text representations that are by their nature grounded in some language. Concretely, for each entity $e$, we have a set of identifiers $\mathcal{I}_e$ that consists of pairs $\langle l, \mathcal{Y}_e^l \rangle$ where $l \in \mathcal{L}_{KB}$ indicates a language and $\mathcal{Y}_e^l$ the sequence of tokens representing the name of the entity $e$ in the language $l$. We extract these identifiers from our KB—each WIKIDATA item has a set of WIKIPEDIA pages in multiple languages linked to it, and in any given language, each page has a unique name. We identify 3 strategies to employ these identifiers:

i) define a *canonical* textual identifier for each entity such that there is a 1-to-1 mapping between the two (*i.e.,* for each entity, select a specific language for its name—see section 5.3.1);

ii) define an $n$-to-1 mapping between textual identifier and entities concatenating a language ID (*e.g.,* a special token) followed by its name in that language—alternatively concatenating its name first and then a language ID (see section 5.3.2);

iii) treat the selection of an identifier in a particular language as a latent variable (*i.e.,* we let the model learn a conditional distribution of languages given the input and we marginalize over those—see section 5.3.3).

All of these strategies define a different way we compute the underlining likelihood of our model. In figure 5.1 we show an outline of mGENRE. The following subsections will present detailed discussions of the above 3 strategies.

### 5.3.1 Canonical entity representation

Selecting a single textual identifier for each entity corresponds to choosing its name among all the available languages of that entity. We employ the same data-driven selection heuristic as in Botha et al. (2020): for each entity $e$ we sort all its names $\mathcal{Y}_e^l$ for each language $l$ according to the number of mentions of $e$ in documents of language $l$. Then we take the name $\mathcal{Y}_e^l$ in the language $l$ that has the most mentions of $e$. In case of a tie, we select the language that has the most number of mentions across all entities (*i.e.,* the language for which we have more training data). Having a single identifier for each entity corresponds to having a 1-to-1

mapping between strings and entities.[2] Thus,

$$\text{score}(e|\mathcal{X};\theta) = p(\mathcal{Y}_e|\mathcal{X};\theta) = \prod_{i=1}^{n} p(y_i|y_{<i},\mathcal{X};\theta) \ , \tag{5.1}$$

where $\mathcal{X}$ is the input (context, mention and delimiters) sequence, $\mathcal{Y}_e = \{y_i\}_{i=1}^{n}$ is the sequence of $n$ tokens in the *canonical* identifier of $e$, and $\theta$ the parameters of the model. Note that this is the same as equation 4.4 of GENRE. A downside of this strategy is that most of the time, the model cannot exploit the lexical overlap between the context and entity name since it has to translate it in the canonical one (*e.g.,* if the canonical name for the entity potato is "Potato" Q10998 and the model encounters "patata"—that is potato in Spanish—it needs to learn that one is the translation of the other).

### 5.3.2 Multilingual entity representation

To accommodate the canonical representation issues, we can predict entity names in any language. Concatenating a language ID $l$ and an entity name $\mathcal{Y}_e^l$ in different orders induces two alternative factorizations. We train maximizing the scores for all our training data:

$$\text{score}(e|\mathcal{X};\theta) = \begin{cases} p(l|\mathcal{X};\theta) \cdot p(\mathcal{Y}_e^l|\mathcal{X},l;\theta) & \text{for 'language+name (L+N)'} \\ p(\mathcal{Y}_e^l|\mathcal{X};\theta) \cdot p(l|\mathcal{Y}_e^l,\mathcal{X};\theta) & \text{for 'name+language (N+L'} \end{cases} \tag{5.2}$$

The former corresponds to first predicting a distribution over languages and then predicting a title *conditioning* on the language $l$ where the latter corresponds to the opposite. Predicting the language first conditions the generation to a smaller set earlier during beam search (*i.e.,* all names in a specific language). However, it might exclude some targets from the search too early if the beam size is too small. Predicting the language last does not condition the generation of names in a particular language but it *asks* the model to disambiguate the language of the generated name whenever it is ambiguous (*i.e.,* when the same name in different languages corresponds to possibly different entities). Only 1.65% of the entity names need to be disambiguated with the language. In practice, we observe negligible difference in performance between the two approaches. Both strategies define an N-to-1 mapping between textual identifiers and entities and then at test time we just use a lookup table to select the correct KB item. This N-to-1 mapping is an advantage compared to using canonical names because the model can predict in any available language and therefore exploit synergies between source and target language as well as avoiding translation.

[2] As this approach chooses one language per entity it might happen that two entities have the same canonical name in the two different languages. We address this issues appending the used language ID so that the combination of the two is always unique.

### 5.3.3 Marginalization

Differently from the plain generation strategies described above, we can treat the textual identifiers as a latent variable and express $\text{score}(e|\mathcal{X};\theta)$ as the probability of the entity name in all languages and marginalizing over them:

$$\text{score}(e|\mathcal{X};\theta) = \sum_{\langle l,\mathcal{Y}_e^l\rangle \in \mathcal{I}_e} p(\mathcal{Y}_e^l, l|\mathcal{X};\theta) = \sum_{\langle l,\mathcal{Y}_e^l\rangle \in \mathcal{I}_e} p(l|\mathcal{X};\theta) \cdot p(\mathcal{Y}_e^l|\mathcal{X}, l;\theta) \ . \quad (5.3)$$

Marginalization exposes the model to all representations in all languages of the same entity and it requires a minor modification of the training procedure. Unfortunately, because computing $\text{score}(e|\mathcal{X};\theta)$ requires a sum over all languages, both training, and inference with marginalization are more expensive than with simple generation (scaling linearly with the number of languages). However, at least during inference, we can still apply BS to only marginalize using the top-$k$ generations. For this reason, we test this training strategy only on few languages but we evaluate marginalization even when training with the other generation strategies described above.

### 5.3.4 Candidate selection

Modern EL systems that employ cross-encoding between context and entities usually do not score all entities in a KB as it is too computational expensive (Wu et al., 2020). Instead, they first apply candidate selection to reduce the number of entities before scoring (with a less expensive method or a non-parametric mention table). In our formulation, there is no need to do that since mGENRE uses Beam Search to generate efficiently. However, using candidates might help, and thus, we also experiment with that. Scoring all candidates might not be always possible (sometimes there are thousands of candidates for a mention) and especially when using an N-to-1 mapping between textual identifiers there will be names to rank in all languages available for each candidate. Then, when we use candidates, it is to constrain BS steps further, rather than to rank all of them. Concretely, candidate selection is made with an alias table. Using the training data, we build a mention table where we record all entities indexed by the names used to refer to them in any language. Additionally, we also use Wikipedia titles as additional mentions (useful for entities that never appear as links), redirects, Wikidata labels, and aliases.

## 5.4 Experimental Setting

We use Wikidata (Vrandečić, 2012) as our KB while exploiting the supervision signal from Wikipedia hyperlinks. For evaluation, we test our model on two established cross-lingual datasets, $\text{TR2016}^{\text{hard}}$ and TAC-KBP2015 (Ji et al., 2015;

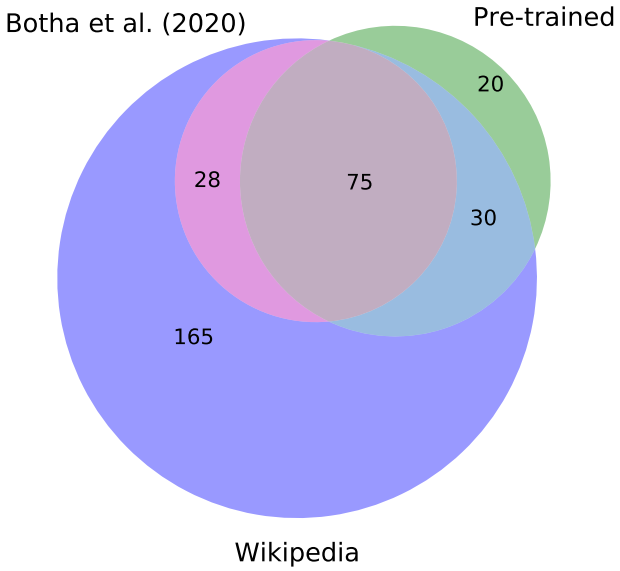



Figure 5.2: Venn diagram on the overlap of languages used during multilingual language modeling (pre-training), the languages available on WIKIPEDIA (as of 2019-10-01), and the languages used by Botha et al. (2020). After pre-training on 125 languages, we fine-tune on the 105 that overlap with the one available in WIKIPEDIA.

Tsai & Roth, 2016), as well as the recently proposed Mewsli-9 MEL dataset (Botha et al., 2020). Additionally, we propose a novel setting extracted from WIKINEWS[3] where we train a model on a set of languages, and we test it on unseen ones.

### 5.4.1 Pre-training

We used a pre-trained mBART (Lewis et al., 2020a; Liu et al., 2020) model on 125 languages—see figure 5.2 for a visual overview of the overlap with these languages, WIKIPEDIA and the languages used by Botha et al. (2020). mBART has 24 layers of hidden size is 1,024 and it has a total of 406M parameters. We pre-trained on an extended version of the `cc100` (Conneau et al., 2020; Wenzek et al., 2020) corpora available here[4] where we increased the number of common crawl snapshots for low resource languages from 12 to 60. The dataset has ≈5TB of text. We pre-trained for 500k steps with max 1,024 tokens per GPU on a variable batch size (≈3000).

[3] https://www.wikinews.org
[4] http://data.statmt.org/cc-100

### 5.4.2 Data for supervision

**Wikidata** We use Wikidata as the target KB to link to. Wikidata contains tens of millions of items but most of them are scholarly articles or they correspond to help and template pages in Wikipedia (*i.e.,* not entities we want to retain).[5] Following (Botha et al., 2020), we only keep Wikidata items that have an associated Wikipedia page in at least one language, independent of the languages we actually model. Moreover, we filter out items that are a subclass (P279) or instance of (P31) some Wikimedia organizational entities (*e.g.,* help and template pages—see table 5.1). Our entity set $\mathcal{E}$ contains 20,277,987 items (as a reference, English Wikipedia has just $\approx$6M items). Using the corresponding Wikipedia titles as textual identifiers in all languages leads to a table of 53,849,351 entity names. We extended the identifiers including redirects which leads to a total of 89,270,463 entity names. Although large, the number of entity names is not a bottleneck as the generated prefix tree occupies just 2.2GB for storage (Botha et al. (2020) systems need $\approx$10 times more storage).

**Wikipedia** We exploit Wikipedia hyperlinks as the source of supervision for MEL. We used Wikipedia in 105 languages out of the $>$300 available. These 105 are all the languages for which our model was pre-trained on that overlaps with the one available in Wikipedia (see full language list in figures 5.3 and 5.4). We aligned each Wikipedia hyperlink to its respective Wikidata item using a custom script. Note that each Wikipedia page maps to a Wikidata item. For the alignment we use i) direct reference when the hyperlink point directly to a Wikipedia page, ii) a re-directions table if the hyperlink points to an alias page, and iii) a Wikidata search among labels and aliases of items if the previous two alignment strategies failed. The previous two alignment strategies might fail when i) authors made a mistake linking on a non-existing page, ii) authors linked to a non-existing page on purpose hoping it will be created in the future, or iii) the original title of a page changed over time and no redirection was added to accommodate old hyperlinks. This procedure successfully aligns 91% of the hyperlinks. We only keep unambiguous alignments since, when using Wikidata search (*i.e.,* the third alignment strategy), the mapping could be ambiguous (*e.g.,* multiple items may share the same labels and aliases). We use a standard Wikipedia extractor `wikiextractor`[6] by Attardi (2015) and a redirect extractor[7]. We use both Wikipedia and Wikidata dumps from 2019-10-01. Eventually, we extracted a large-scale dataset of 734,826,537 datapoints (*i.e.,* mention-entity pairs). For the plain generation strategy, we selected as the ground

[5] https://www.wikidata.org/wiki/Wikidata:Statistics
[6] https://github.com/attardi/wikiextractor
[7] https://code.google.com/archive/p/wikipedia-redirect

| WIKIDATA ID | Label |
|---|---|
| Q4167836 | category |
| Q24046192 | category stub |
| Q20010800 | user category |
| Q11266439 | template |
| Q11753321 | navigational template |
| Q19842659 | user template |
| Q21528878 | redirect page |
| Q17362920 | duplicated page |
| Q14204246 | project page |
| Q21025364 | project page |
| Q17442446 | internal item |
| Q26267864 | KML file |
| Q4663903 | portal |
| Q15184295 | module |

Table 5.1: WIKIDATA identifiers used for filtering out items from Botha et al. (2020).

truth the name in the source language. When such entity name is not available[8] we randomly select 5 alternative languages and we use all of them as datapoints. To enable model selection, we randomly selected 1k examples from each language for validation.

`cc125` For all experiments, we do not train a model from scratch, but we fine-tune a multilingual language model trained on 125 languages. `cc125` is an extension of `cc100` (Conneau et al., 2020; Wenzek et al., 2020) corpora available here[9] where we increased the number of common crawl snapshots for low resource languages from 12 to 60. Unfortunately, this extension is not public.[10]

### 5.4.3 Data for test

**Mewsli-9** (Botha et al., 2020) contains 289,087 entity mentions appearing in 58,717 originally written news articles from WIKINEWS, linked to WIKIDATA. The corpus includes documents in 9 languages.[11] Differently from the cross-lingual

[8] This happens when there are broken links or links that points to pages in prospect of being created.

[9] http://data.statmt.org/cc-100

[10] The work of this chapter was produced during an internship at Facebook AI https://ai.facebook.com; some proprietary data was used.

[11] Arabic, English, Farsi, German, Japanese, Serbian, Spanish, Tamil, and Turkish.

setting, this is a truly multilingual dataset since 11% target entities in Mewsli-9 do not have an English WIKIPEDIA page.

**TR2016$^{\text{hard}}$** (Tsai & Roth, 2016) is a WIKIPEDIA based cross-lingual dataset specifically constructed to contain difficult mention-entity pairs. Authors extracted WIKIPEDIA hyperlinks for which the corresponding entity is not the most likely when using an alias table. Since we train on WIKIPEDIA, to avoid an overlap with this test data, we removed all mentions from our training data that also appear in TR2016$^{\text{hard}}$. Note that this pruning strategy is more aggressive than Tsai and Roth (2016) and Botha et al. (2020) strategies. Tsai and Roth (2016) assured to not have mention-entity pairs overlaps between training and test, but a mention (with a different entity) might appear in training. Botha et al. (2020)[12] split at the page-level only, making sure to hold out all Tsai and Roth (2016) test pages (and their corresponding pages in other languages), but they trained on any mention-entity pair that could be extracted from their remaining training page partition (*i.e.,* they have overlap between training and text entity-mention pairs). To compare with previous works (Tsai & Roth, 2016; Upadhyay et al., 2018; Botha et al., 2020) we only evaluate on German, Spanish, French and Italian (a total of 16,357 datapoints).

**TAC-KBP2015** To evaluate our system on documents out of the WIKIPEDIA domain, we experiment on the TAC-KBP2015 Tri-Lingual Entity Linking Track (Ji et al., 2015). To compare with previous works (Tsai & Roth, 2016; Upadhyay et al., 2018; Sil et al., 2018; Zhou et al., 2019), we use only Spanish and Chinese (*i.e.,* we do not evaluate in English). Following previous work, we only evaluate *in-KB* links (Yamada et al., 2016; Ganea & Hofmann, 2017), i.e, we do not evaluate on mentions that link to entities out of the KB. Previous works considered Freebase (Bollacker et al., 2008) as KB, and thus we computed a mapping between Freebase ID and WIKIDATA ID. When we cannot solve the match, our system gets zero scores (*i.e.,* it counts as a wrong prediction). TAC-KBP2015 contains 166 Chinese documents (84 news and 82 discussion forum articles) and 167 Spanish documents (84 news and 83 discussion forum articles) for a total of 12,853 mention-entity datapoints.

**WIKINEWS-7** For the purpose of testing a model on languages unseen during training, we extract mention-entities pairs from WIKINEWS in 7 languages that are not in the Mewsli-9 language set.[13] Table 5.5 reports statistics of this dataset. WIKINEWS-7 is created in the same way as Mewsli-9, but we used our own implementation to extract data from raw dumps.[14]

[12]Information provided by private correspondence with the authors.
[13]Chinese, Czech, French, Italian, Polish, Portuguese, and Russian.
[14]Botha et al. (2020) did not release code for extracting Mewsli-9 from a WIKINEWS dump.

### 5.4.4 Training

We implemented, trained, and evaluate our model using the `fariseq` library (Ott et al., 2019). We trained mGENRE using Adam (Kingma & Ba, 2015) with a learning rate $1e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and with a linear warm-up for 5,000 steps followed by liner decay for maximum 2M steps. The objective is sequence-to-sequence categorical cross-entropy loss with 0.1 of label smoothing and 0.01 of weight decay. We used dropout (Srivastava et al., 2014b) probability of 0.1 and attention dropout of 0.1. We used max 3,072 tokens per GPU and variable batch size ($\approx 12,500$). Training was done on 384 GPUs (Tesla V100 with 32GB of memory) and it completed in $\approx$72h for a total of $\approx 27,648$ GPU hours or $\approx 1,152$ GPU days. Since TAC-KBP2015 contains noisy text (*e.g.,* XML/HTML tags), we further fine-tune mGENRE for 2k steps on its training set when testing on it. For evaluation we use the recent multilingual-EL dataset Mewsli-9 (Botha et al., 2020), the cross-lingual TAC-KBP2015 Tri-Lingual Entity Linking (Ji et al., 2015) and TR2016$^{\text{hard}}$ (Tsai & Roth, 2016). We refer to the original works for details on the data.

### 5.4.5 Inference

At test time, we use Constrained Beam Search with 10 beams, length penalty of 1, and maximum decoding steps of 32. We restrict the input sequence to be at most 128 tokens cutting the left, right, or both parts of the context around a mention. When employing marginalization, we normalize the log-probabilities by sequence length using $\log p(\mathcal{Y}|\mathcal{X})/L^{\alpha}$, where $\alpha = 0.5$ was tuned on the development set (testing values between 0 and 2 at 0.1 intervals).

## 5.5 Results

The main results of this chapter are reported in table 5.2 for Mewsli-9, and in table 5.3 for TR2016$^{\text{hard}}$, and TAC-KBP2015 respectively. Our mGENRE (trained with 'Name+Language' and used with marginalization and candidates) outperforms all previous works in all those datasets. We show the accuracy of mGENRE on the 105 languages in our Wikipedia validation set against an alias table baseline in figures 5.3 and 5.4.

### 5.5.1 Performance evaluation

**Mewsli-9** In table 5.2 we compare our mGENRE against the best model from Botha et al.'s (2020) Model F$^{+}$ as well as with their alias table baseline. We report results from mGENRE with and without constraining the beam search to the candidates from the table (see section 5.3.4) as well as with and without marginalization (see section 5.3.3). All of these alternatives outperform Model F$^{+}$

| | ar | de | en | es | fa | ja | sr | ta | tr | **micro-avg** | **macro-avg** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Alias Table | 89.0 | 86.0 | 79.0 | 82.0 | 87.0 | 82.0 | 87.0 | 79.0 | 80.0 | 83.0 | 83.0 |
| Botha et al. (2020) | 92.3 | 91.5 | **87.2** | 89.0 | 91.8 | 87.8 | 92.6 | 87.6 | 88.9 | 89.4 | 89.8 |
| **mGENRE** | 94.7 | 91.5 | 86.7 | 90.0 | **94.6** | 89.9 | 94.9 | 92.9 | 90.7 | 90.2 | 91.8 |
| + marg. | 95.3 | 91.8 | 87.0 | **90.1** | 94.2 | 90.2 | **95.0** | 93.1 | 90.9 | 90.4 | 92.0 |
| + cand. | 94.8 | 91.8 | 87.1 | **90.1** | **94.6** | 91.1 | 94.4 | 93.3 | 91.4 | 90.5 | 92.1 |
| + cand. + marg. | **95.4** | **92.0** | **87.2** | **90.1** | 94.4 | **91.4** | 94.5 | **93.8** | **91.5** | **90.6** | **92.3** |
| Subsequent Work | | | | | | | | | | | |
| FitzGerald et al. (2021) | 93.4 | 91.5 | 87.4 | 88.7 | 92.9 | 88.5 | 92.5 | 91.3 | 89.1 | 89.6 | 90.6 |

Table 5.2: Accuracy on Mewsli-9 dataset with micro and macro averages. The results of mGENRE are with and without top-k candidates from the alias table as well as with and without marginalization. **Bold** indicate best result.

| Method | TAC-KBP2015 | | | TR2016$^{hard}$ | | | | |
|---|---|---|---|---|---|---|---|---|
| | **es** | **zh** | **macro-avg** | **de** | **es** | **fr** | **it** | **macro-avg** |
| Tsai and Roth (2016) | 82.4 | 85.1 | 83.8 | 53.3 | 54.5 | 47.5 | 48.3 | 50.9 |
| Sil et al. (2018)* | 83.9 | 85.9 | 84.9 | - | - | - | - | - |
| Upadhyay et al. (2018) | 84.4 | 86.0 | 85.2 | 55.2 | 56.8 | 51.0 | 52.3 | 53.8 |
| Zhou et al. (2019) | 82.9 | 85.5 | 84.2 | - | - | - | - | - |
| Botha et al. (2020) | - | - | - | **62.0** | 58.0 | 54.0 | 56.0 | 57.5 |
| **mGENRE** | 86.3 | 64.6 | 75.5 | 56.3 | 57.1 | 50.0 | 51.0 | 53.6 |
| **mGENRE** + marg. | **86.9** | 65.1 | 76.0 | 56.2 | 56.9 | 49.7 | 51.1 | 53.5 |
| **mGENRE** + cand. | 86.5 | 86.6 | 86.5 | 61.8 | **61.0** | **54.3** | **56.9** | **58.5** |
| **mGENRE** + cand. + marg. | 86.7 | **88.4** | **87.6** | 61.5 | 60.6 | **54.3** | 56.6 | 58.2 |
| Subsequent Work | | | | | | | | |
| FitzGerald et al. (2021) | - | - | - | 64.0 | 59.0 | 58.0 | 59.0 | 60.0 |

Table 5.3: Accuracy on TAC-KBP2015 Entity Linking dataset (only datapoints linked to FreeBase) and TR2016$^{hard}$ of mGENRE (trained with 'title+lang') with and without top-k candidates from the table as well as with and without marginalization. *as reported by Upadhyay et al. (2018).

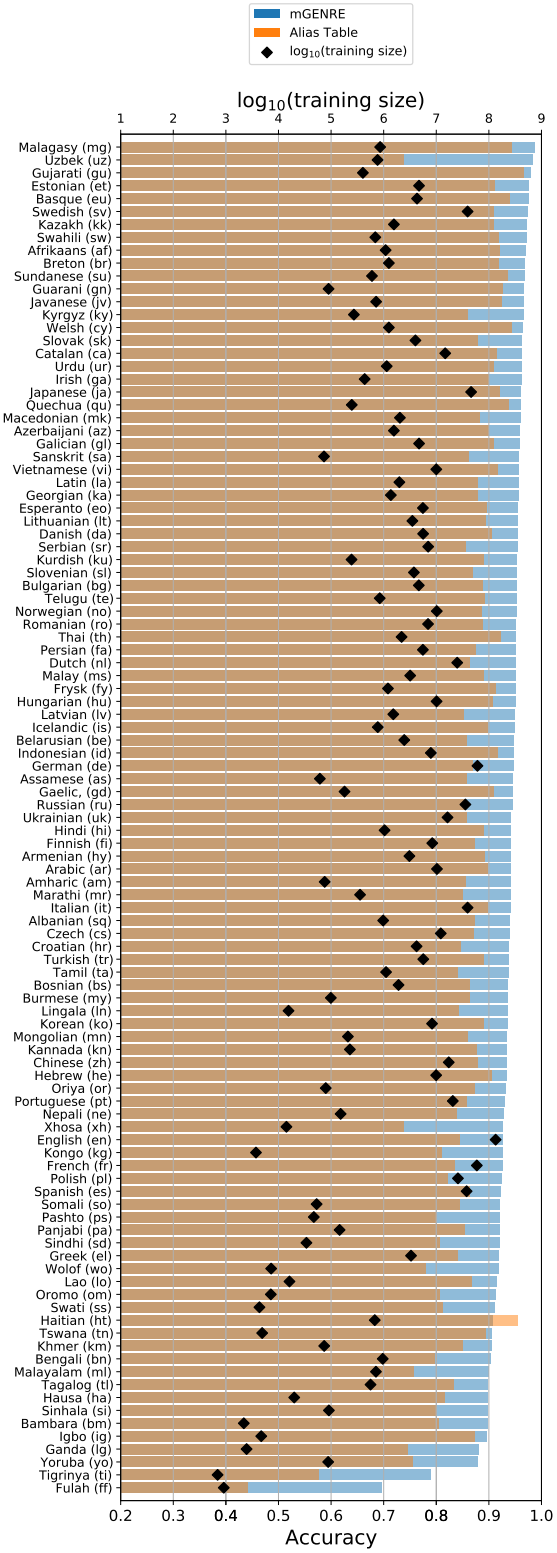



Figure 5.3: Accuracy of mGENRE and alias table on the 105 languages in our Wikipedia validation set sorted by accuracy. See Table A.2 for all precise values.

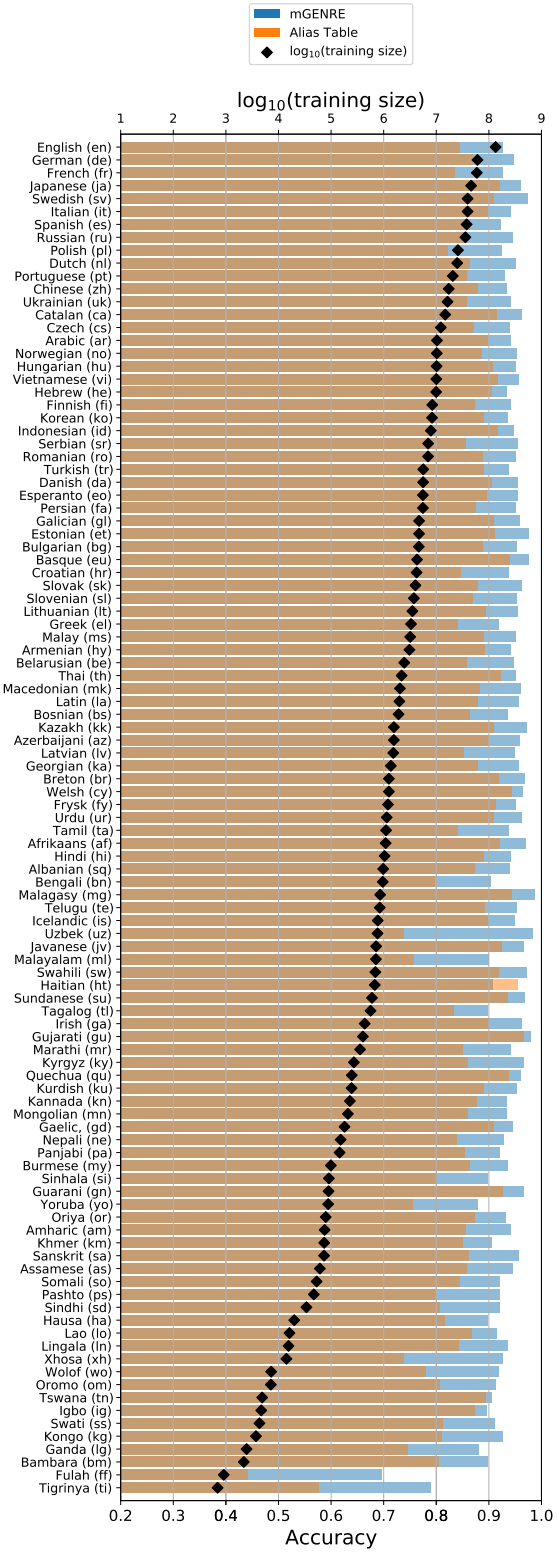



Figure 5.4: Accuracy of mGENRE on the 105 languages in our Wikipedia validation set sorted by training set size. See Table A.2 for all precise values.

on both micro and macro average accuracy across the 9 languages. Our base model (without candidates or marginalization) has a 10.9% error reduction in micro average and 18.0% error reduction for macro average over all languages. The base model has no restrictions on candidates so it is effectively classifying among all the ≈20M entities. The base model performs better than Model $F^+$ on each individual language except English and German. Note that these languages are the ones for which we have more training data (≈134M and ≈60M datapoints each) but also the languages that have the most entities/pages (≈6.1M and ≈2.4M). Therefore these are the hardest languages to link. When enabling candidate filtering to restrict the space for generation, we further improve error reduction to 13.6% and 21.0% for micro and macro average respectively. Although candidate selection is not required by our general formulation, it definitely helps to restrict the search space when candidates are available (note that recall@k using all the candidates is >98% for all languages and on average using candidates reduces the search space form ≈20M entities to a few hundreds—*e.g.,* see figure 5.5 for a breakout of results by the number of retrieved candidates). Marginalization reduces the error by the same amount as candidate filtering but combining search with candidates and marginalization leads to our best model: it improves error reduction to 14.5% and 23.0% on micro and macro average respectively. Our best model is also better than Model $F^+$ in English and on par with it in German.

**$\text{TR2016}^{\text{hard}}$ and TAC-KBP2015** We compared our mGENRE against cross-lingual systems (Tsai & Roth, 2016; Sil et al., 2018; Upadhyay et al., 2018; Zhou et al., 2019) and Model $F^+$ by Botha et al. (2020) in table 5.3. Differently from Meswli-9, the base mGENRE model does not outperform previous systems. Using marginalization brings minimal improvements. Instead, using candidates gives +11% absolute accuracy on TAC-KBP2015 and +5% on $\text{TR2016}^{\text{hard}}$ effectively making mGENRE state-of-the-art in both datasets. The role of candidates is very evident on TAC-KBP2015 where there is not much of a difference for Spanish but a +22% absolute accuracy for Chinese. TAC-KBP2015 comes with a training set and we used it to expand the candidate set. Additionally, we also included all simplified Chinese versions of the entity names because we used traditional Chinese in pre-training, and TAC-KBP2015 uses simplified Chinese. Many mentions in TAC-KBP2015 were not observed in Wikipedia, so the performance gain mostly comes from this but including the simplified and alternative Chinese names also played an important role (+5% comes from this alone).[15]

[15] We speculate that including different version (*e.g.,* different dialects for Arabic) of entity names could improve performance in all languages. Since this is not in the scope of this chapter, we will leave it for future.

| | Botha et al. (2020) | | **mGENRE** | |
|---|---|---|---|---|
| **Bin** | Support | Acc. | Support | Acc. |
| [0, 1) | 3,198 | 8.3 | 1,244 | 22.1 |
| [1, 10) | 6,564 | 57.7 | 5,777 | 47.3 |
| [10, 100) | 32,371 | 80.4 | 28,406 | 77.3 |
| [100, 1k) | 66,232 | 89.6 | 72,414 | 89.9 |
| [1k, 10k) | 78,519 | 92.9 | 84,790 | 93.2 |
| [10k, +) | 102,203 | 94.1 | 96,456 | 96.3 |
| **micro-avg** | 289,087 | 89.4 | 289,087 | 90.6 |
| **macro-avg** | - | 70.5 | - | 71.0 |

Table 5.4: Accuracy on the Mewsli-9 dataset, by entity frequency in training (full-model, *i.e.,* trained with 'Language+Name' and used with marginalisation and candidates). The support is slightly different because training data differ (*i.e.,* the set of languages from WIKIPEDIA is different).

### 5.5.2 Analysis

**By entity frequency** Table 5.4 shows a breakdown of Mewsli-9 accuracy by entity frequency in training for Botha et al. (2020) Model $F^+$ and mGENRE. Interestingly, our model has much higher accuracy (22% vs 8%) on unseen entities (*i.e.,* the $[0, 1)$ bin). This is because our formulation can take advantage of copying names from the source, translating them or normalizing them. For example, an unseen person name should likely be linked to the entity with the same name. This powerful bias gives the model advantage in these cases. On very rare entities (*i.e.,* the $[1, 10)$ bin) our model performs worse than Model $F^+$. Note that Model $F^+$ was trained specifically to tackle those cases (*e.g.,* with hard negatives and frequency-based mini-batches) whereas our model was not. We argue that similar strategies can be applied to mGENRE to improve performance on rare entities, and we leave that to future work. The performance gap between Model $F^+$ and mGENRE on entities that appear more than 100 times in the training set is minimal.

**By candidate frequency** We additionally measure the accuracy on Mewsli-9 by the number of candidates retrieved from the alias table (details in figure 5.5). When there are no candidates ($\approx 4\%$ of Mewsli-9) an alias table would automatically fail, but mGENRE uses the entire KB as candidates and has 63.9% accuracy. For datapoints with few candidates (*e.g.,* less than 100), we could use mGENRE as a *ranker* and score all of the options without relying on constrained beam search. However, this approach would be computationally infeasible when there are no candidates (*i.e.,* we use all the KB as candidates) or too many candidates (*e.g.,* thousands). Constrained BS allows us to efficiently explore the

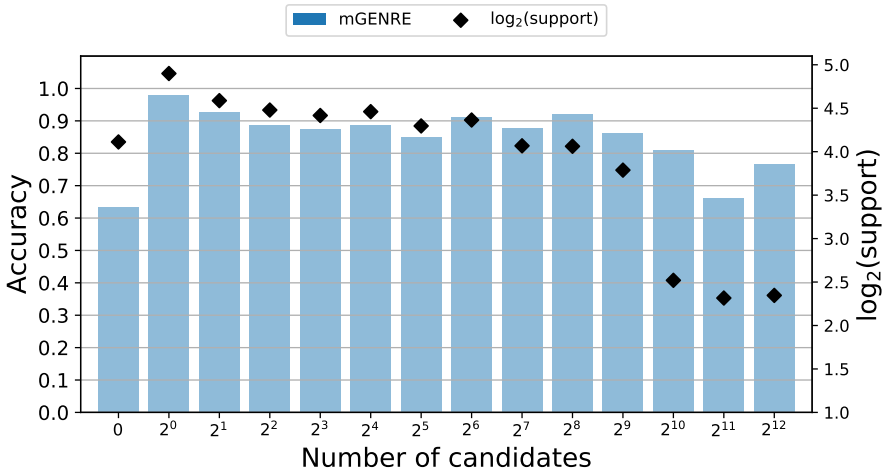



Figure 5.5: Accuracy on Mewsli-9 by the number of retrieved candidates (full-model, *i.e.,* trained with 'Language+Name' and used with marginalisation and candidates). We also indicate the support of each bin (in log-scale). No candidates (0 bin) corresponds to considering all items in the KB as candidates.

space of entity names, whatever the number of candidates.

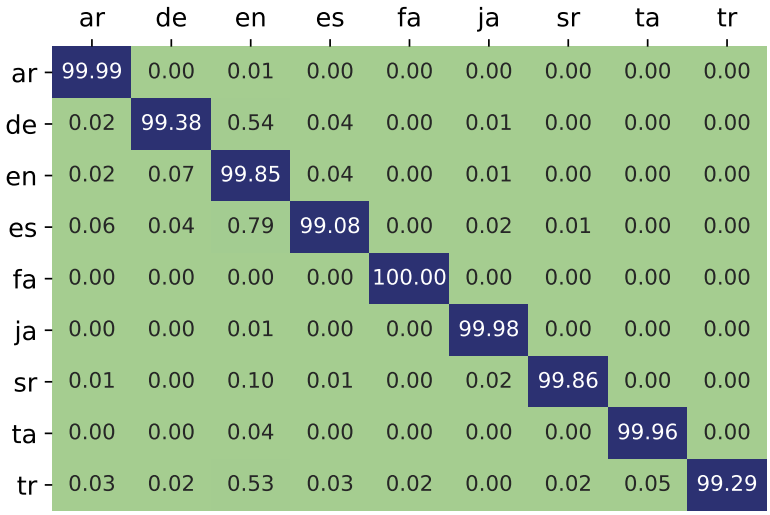
**By mention frequency** We show a breakdown of the accuracy of mGENRE on Mewsli-9 by mention frequency in table 5.6. The accuracy of unseen mentions is 66.7% and increases up to 93.6% for mentions seen more than 10k times. For extremely common mentions (*i.e.,* seen more than 1M times) the accuracy drops to 73.2%. These mentions correspond to entities that are harder to disambiguate (*e.g.,* 'United States' appears 3.2M times but can be linked to the country as well as any sports team where the context refers to sports).

**Unseen Languages** We use our WIKINEWS-7 dataset to evaluate mGENRE capabilities to deal with languages not seen during training (*i.e.,* the set of languages in train and test are disjoint). This zero-shot setting implies that no mention table is available during inference; hence we do not consider candidates for test mentions. We train our models on the nine Mewsli-9 languages and compare all strategies exposed in section 5.3. To make our ablation study feasible, we restrict the training data to the first 1 million hyperlinks from WIKIPEDIA abstracts. Results are in table 5.5.

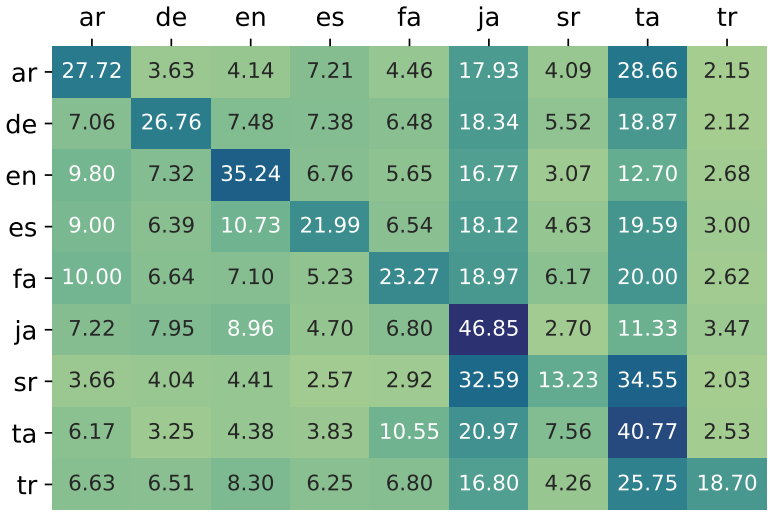
Using our marginalization strategy that aggregates (both at training and inference time) over all seen languages to perform the linking brings an improvement of over 50% with respect to considering a single language. To deeper investigate the behaviour of the model in this setting, we compute the probability mass distribution over languages seen at training time for the top-1 prediction (reported in figure 5.7). When marginalization is enabled (figure 5.7b) the distribution

| Language | Canonical | N+L | L+N | L+N$^{M}$ |
|---|---|---|---|---|
| ar | 90.5 | 92.8 | **92.9** | 89.2 |
| de | **84.6** | 86.4 | 86.4 | 85.3 |
| en | 77.6 | **79.3** | 79.2 | 76.5 |
| es | 83.4 | **85.5** | 85.2 | 83.4 |
| fa | 91.6 | 90.7 | **91.8** | 88.2 |
| ja | 81.3 | 82.3 | **82.8** | 81.3 |
| sr | 91.5 | 92.7 | **92.9** | 92.5 |
| ta | **92.8** | 91.8 | 91.9 | 91.3 |
| tr | **88.0** | 87.7 | 87.3 | 86.0 |
| **micro-avg** | 83.20 | 84.77 | **84.80** | 83.05 |
| **macro-avg** | 86.82 | 87.68 | **87.82** | 85.97 |
| | | + candidates | | |
| ar | 94.4 | 94.5 | **94.7** | 93.0 |
| de | 89.4 | **89.8** | **89.8** | 89.3 |
| en | 83.6 | 83.8 | **83.9** | 82.4 |
| es | 87.7 | 88.2 | **88.3** | 87.3 |
| fa | **93.6** | 93.3 | **93.6** | 93.3 |
| ja | 87.9 | 88.0 | **88.4** | 87.9 |
| sr | 93.1 | 93.4 | **93.5** | 93.2 |
| ta | **93.0** | 92.2 | 92.5 | 92.5 |
| tr | **91.1** | 90.4 | 89.9 | 89.1 |
| **micro-avg** | 87.95 | 88.22 | **88.32** | 87.43 |
| **macro-avg** | 90.42 | 90.41 | **90.51** | 89.78 |
| | | Unseen languages | | |
| cs | 36.3 | 30.2 | 34.0 | **69.7** |
| fr | 62.9 | 57.0 | 53.3 | **73.4** |
| it | 44.8 | 43.7 | 42.9 | **56.8** |
| pl | 31.9 | 21.2 | 25.6 | **68.8** |
| pt | 60.8 | 61.7 | 59.5 | **76.2** |
| ru | 34.9 | 32.4 | 35.1 | **65.8** |
| zh | 35.1 | 41.1 | 44.0 | **52.8** |
| **micro-avg** | 41.6 | 38.3 | 39.5 | **65.9** |
| **macro-avg** | 43.8 | 41.0 | 42.1 | **66.2** |

Table 5.5: Accuracy on the Mewsli-9 and WIKINEWS-7 datasets. Models are trained only on the Mewsli-9 languages (1M datapoints per language: `ar`, `de`, `en`, `es`, `fa`, `ja`, `sr`, `ta`, and `tr`). N+L stands for Name+Language and L+N is for Language+Name. $^{M}$ indicates marginalization.




(a) Language+Name.

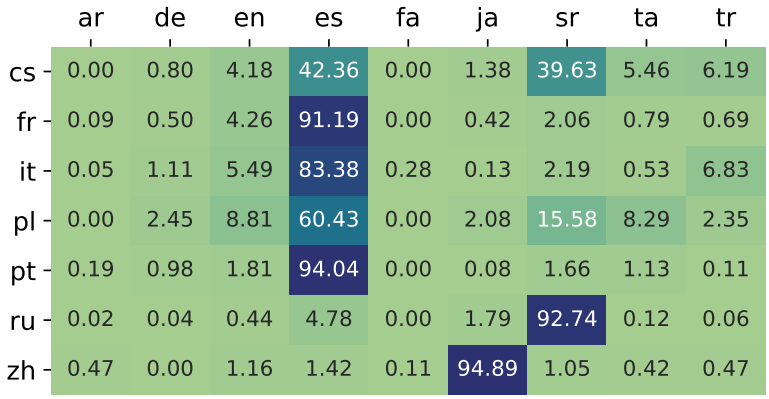



(b) Language+Name$^{M}$.

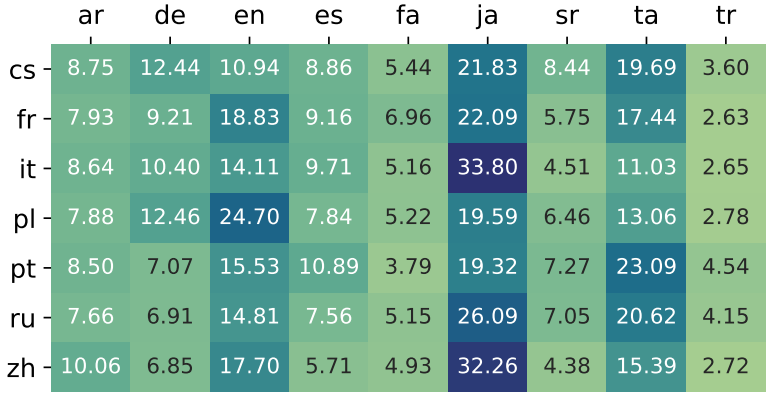
Figure 5.6: Distribution of languages on the top-1 prediction of two mGENRE models on Mewsli-9. Y-axis indicates the source language where X-axis indicates the language of the top-1 prediction. The models trained on those languages.

is more spread across languages since the model is trained to use all of them. Hence the model can exploit connections between an unseen language and all seen languages for the linking process drastically increases accuracy.

Marginalization is effective for this zero-shot setting, but it has a minimal impact in the standard setting (*e.g.*, tables 5.2 and 5.3). When a model has seen the source language at training time it mainly makes use of that to perform a prediction (*i.e.*, the target prediction is in the source language most of the times— >99% see figure 5.6). Instead, when the source language is never seen




(a) Language+Name.




(b) Language+Name$^{M}$.

Figure 5.7: Distribution of languages on the top-1 prediction of two mGENRE models on WIKINEWS-7 (test set). Y-axis indicates the source language (unseen at training time) where X-axis indicates the language (seen at training time) of the first prediction. Note that the models are only trained on `ar`, `de`, `en`, `es`, `fa`, `ja`, `sr`, `ta`, and `tr`.

during training, by marginalising the model can exploit similarities with all seen languages. Indeed, even though marginalization and canonical representation are the top-two systems in the unseen languages setting, they are not on seen languages on the same setting: in table 5.5 we report the results of all these strategies also on the seen languages (Mewsli-9 test set). Complementary to figure 5.7 we also report the probability mass distribution over languages seen for Mewsli-9 in figure 5.6.

**Memory Footprint** As computational cost and memory footprints are important aspects of modeling, we compared the number of parameters used by mGENRE and the best competing mEL system by Botha et al. (2020). Their model has ≈73M parameters and ≈15B entity parameters for a total memory

| Bin | Support | Accuracy |
|---|---|---|
| [0, 1) | 14,741 | 66.7 |
| [1, 10) | 15,279 | 88.1 |
| [10, 100) | 43,169 | 92.0 |
| [100, 1k) | 75,927 | 91.7 |
| [1k, 10k) | 80,329 | 91.5 |
| [10k, 100k) | 47,944 | 93.6 |
| [100k, 1M) | 11,460 | 93.0 |
| [1M, 10M) | 238 | 73.2 |

Table 5.6: Accuracy results on Mewsli-9 dataset by mention frequency in training (full-model, *i.e.,* trained with 'Language+Name' and used with marginalisation and candidates). We also indicate the support of each bin.

usage of ≈61GB where mGENRE has ≈406M model parameters, no entity parameters (*i.e.,* we just have a prefix tree with entity names that occupies ≈2.2GB), for a total of ≈6GB memory usage (*i.e.,* ≈10 times less memory).

**Examples** In table 5.7 we report some examples of correct and wrong predictions of our mGENRE L+N and N+L on selected datapoints from Mewsli-9. Examples are picked to highlight specific behaviors of our models. We show an example of the copying mechanisms (*i.e.,* the N+L model normalizes the mention but L+N fails to do so) as well as an example where the model memorized an acronym (*i.e.,* "MDC" as Movement for Democratic Change) and outputs that correctly in the case of L+N and wrongly for N+L.

## 5.6 Subsequent Work

Most of the relevant subsequent work is shared with chapter 4 and thus discussed in section 4.6. More specific to multilingual entity linking is the recent work Mention Only Linking of Entities with a Mention Annotation Network (MOLEMAN; FitzGerald et al., 2021) that builds upon the Botha et al.'s (2020) Model $F^+$. It is still a bi-encoder type of model that scores mention-entity pairs with a dot-product. However, it overcomes the need for entity descriptions entirely: they only learn a mention-encoder that places similar mentions of the same entity closer in vector space. Then, "all mentions of an entity to serve as prototypes as inference involves retrieving from the full set of labeled entity mentions in the training set". We report MOLEMAN's results in tables 5.2 and 5.3.

| | |
|---|---|
| **Input** | Police in Zimbabwe have stopped opposition leader Morgan Tsvangirai ( **[START] MDC [END]** ) en route to a campaign rally. His convoy was then escorted to a police station in Esigodini. |
| **Correct by L+N** | en » Movement for Democratic Change[Q6926644] |
| **Wrong by N+L** | People's Democratic Party (Zimbabwe) » en[Q48798212] |
| **Input** | Sin embargo, la promoción del **[START] abstencionismo [END]** por parte de los opositores se tradujo en una participación de apenas el 47.32%, alrededor de 9.2 millones de electores. En las municipales de 2013 habían participado 58.92% de los venezolanos con derecho a voto. 61% en los comicios regionales de octubre. |
| **Wrong by L+N** | es » Oposición (política)[Q192852] |
| **Correct by N+L** | Abstención » es[Q345321] |

Table 5.7: Examples of correct and wrong predictions of our mGENRE models Language+Name (L+N) and Name+Language (N+L) on selected samples from Mewsli-9.

## 5.7 Conclusion

In this chapter, we propose an autoregressive formulation to the multilingual entity linking problem. For a mention in a given language, our solution generates entity names left-to-right and token-by-token. The resulting system, maintains entity names in as many languages as possible to exploit language connections and interactions between source mention context and target entity name. The constrained beam search decoding strategy enables fast search within a large set of entity names (*e.g.,* the whole KB in multiple languages) with no need for large-scale dense indices. We additionally design a novel objective that marginalizes over all available languages to perform a prediction. We show that this strategy is really effective in dealing with languages for which no training data is available (*i.e.,* 50% improvements for languages never seen during training). Overall, our experiments show that mGENRE achieves new state-of-the-art performance on three popular multilingual entity linking datasets.

# Chapter 6

# Highly Parallel and Fast Autoregressive Entity Linking

**Chapter Highlights**

In the previous chapters we show that generative approaches are effective for both Entity Disambiguation and Entity Linking (*i.e.,* joint mention detection and disambiguation). However, the previously proposed autoregressive formulation for EL suffers from i) high computational cost due to a complex (deep) decoder, ii) non-parallelizable decoding that scales with the source sequence length, and iii) the need for training on a large amount of data. In this chapter, we propose a very efficient approach that parallelizes autoregressive linking across all potential mentions and relies on a shallow and efficient decoder. Moreover, we augment the generative objective with an extra discriminative component, *i.e.,* a correction term which lets us directly optimize the generator's ranking. When taken together, these techniques tackle all the above issues: our model (ParallelAEL) is $>70$ times faster and more accurate than the previous generative method, outperforming state-of-the-art approaches on the standard English dataset AIDA-CoNLL.[1]

[1] Source code available at https://github.com/nicola-decao/efficient-autoregressive-EL

## 6.1 Introduction

Employing autoregressive language models for both Entity Disambiguation (ED) and Entity Linking (EL) better leverages the implicit knowledge accumulated during pre-training, exploiting a full cross-encoder of entities and their context (see chapters 4 and 5). For ED, autoregressive generation is remarkably good (even in multilingual settings), while for EL, although state-of-the-art on multiple datasets, it suffers from several and critical limitations. The generative model of De Cao et al. (2021a) (see chapter 4) outputs a version of the input document which is markup-annotated with mentions linked to their respective entities. This necessitates using an autoregressive decoder, precluding parallelism across mentions. Generation also has a high computational cost due to relying on a complex and deep Transformer (Vaswani et al., 2017) decoder. Transformers are state-less and their memory footprint scales with sequence length, making them memory-consuming when generating long sequences. Additionally, Transformers-based decoders are notably data-hungry, and their effective training requires large amounts of data. For example, in chapter 4 we had to pre-train their model on WIKIPEDIA abstracts.

In this chapter, we revisit the generative approach to EL and generate mention-entity pairs conditionally independently given the input. This allows for parallelism across mentions, which we exploit by employing a shallow LSTM-based decoder. To optimize more explicitly the generator's ranking, we use a discriminative correction term that pushes the score of the correct predictions to be higher than the rest. Moreover, to enable conditioning on long inputs, we employ an efficient Transformer encoder (Beltagy et al., 2020) designed to support long sequences. Figure 6.1 outlines our model.

**Contributions** We propose a highly parallel model for autoregressive entity linking that retains the advantages of being generative while being $>70$ times faster than a previous generative formulation and as fast as non-generative models. We optimize for the correctness of the decoder's ranking with a discriminative loss to improve autoregressive EL further. The model outperforms state-of-the-art approaches on the standard English AIDA dataset.

## 6.2 Background

Our formulation builds upon the GENRE model by De Cao et al. (2021a). See chapter 4 for a discussion on the GENRE model and relevant related work.

### 6.2.1 Task Definition

Given a document $d$ (*e.g.,* a sentence) an Entity Linking (EL) system $f$ has to return a set $\mathcal{M}$ of mention-entity pairs $\mathcal{M} = f(d) = \{\langle m_i, e_i \rangle\}_{i=1}^{n}$ where each $m_i$

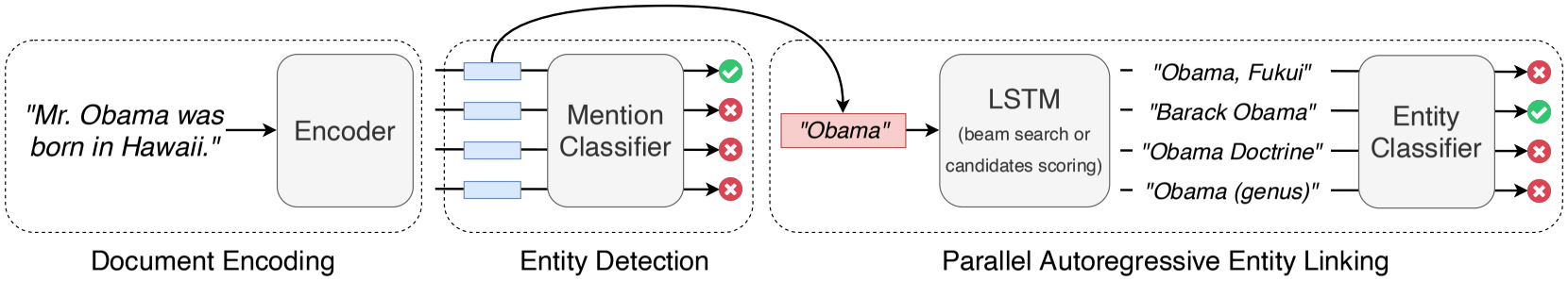



Figure 6.1: Outline of ParallelAEL: a Transformer-based *document encoder* embeds a document into vectors (the encoder is designed to support long text). Then, an *entity detection* module classifies which spans in the document are entity mentions. Conditioning on a mention embedding, an *entity linking* module first uses an LSTM to either generate or score candidates' textual identifiers and then a classifier to re-rank the candidates.

is a entity mentions (Hoffmann et al., 2011). Each mention $m_i$ is a span contained in $d$, *i.e.,* is a pair of start and end positions $\langle m_i^s, m_i^e \rangle$ in $d$. Each mention $m_i$ refers to an entity $e_i \in \mathcal{E}$ in a fixed Knowledge Base (KB)—note that entities can be referred to with multiple ambiguous surface forms (*e.g.,* a mention of "Obama" may refer to the entity "Barack Obama"[2] while in some context it can refer to "Michelle Obama"[3]).

### 6.2.2 Related Work

EL is typically decomposed in Mention Detection (MD; *i.e.,* the task of finding mention spans in text) and Entity Disambiguation (ED; *i.e.,* the task of disambiguating a mention to its respective entity). Many methods (Hoffart et al., 2011; Piccinno & Ferragina, 2014; Steinmetz & Sack, 2013) treat these sub-tasks separately, training different modules. More modern approaches – known as end-to-end EL – instead use a shared (typically neural) architecture. Kolitsas et al. (2018) use a bidirectional LSTM (Hochreiter & Schmidhuber, 1997) as an encoder and then local and global scoring functions to link mentions. They exploit pre-computed entity embeddings by Ganea and Hofmann (2017) and match the embeddings to contextualized mention representations. Martins et al. (2019) also explore joint learning of Named Entity Recognition (NER) and EL showing that the two tasks benefit from joint training, while Li et al. (2020) approach EL specifically for questions.

In this chapter, we focus on monolingual EL in English while there is a line of work that explores cross-lingual entity linking (McNamee et al., 2011; Ji et al., 2015), that is linking from any source language to a standard one (*e.g.,* English), and multilingual entity linking (Botha et al., 2020) that is a generalization of both.

## 6.3 Method

Our method learns by generating observed mention-entity pairs $\mathcal{M}$ given an input document $d$ made of a sequence of tokens $\mathcal{X}$. To create a key opportunity for parallelism, we assume that, given the document tokens $\mathcal{X}$, each mention-entity pair $\langle m_i, e_i \rangle \in \mathcal{M}$ is independent of one another. Moreover, each pair's probability is further factorized as a product of an MD and an ED components:

$$p(\mathcal{M}|\mathcal{X};\theta) \overset{\text{ind.}}{=} \prod_{\langle m_i,e_i\rangle \in \mathcal{M}} p(m_i|\mathcal{X};\theta_{\text{MD}})\; p(e_i|m_i,\mathcal{X};\theta_{\text{ED}}) \;, \tag{6.1}$$

where $\theta = \theta_{\text{MD}} \cup \theta_{\text{ED}}$ is a shared set of parameters (and $\theta_{\text{MD}} \cap \theta_{\text{ED}}$ need not be empty). To provide our models with a rich representation of the document, we

[2] https://www.wikidata.org/wiki/Q76
[3] https://www.wikidata.org/wiki/Q13133

encode it using a Longformer (Beltagy et al., 2020), a Transformer pre-trained with a masked language model objective that is designed to support long sequences.

**Mention Detection** There are different ways to model $p(m_i|\mathcal{X};\theta_{\text{MD}})$ (*i.e.,* the probability that the span $m_i$ in $\mathcal{X}$ contains a mention). One is to score all possible spans which requires a number of evaluations that is quadratic in sequence length. For long documents, that is clearly unfeasible. Thus, for maximizing efficiency, we opt for factorizing the probability of a span as the probability of its start $m_i^s$ times the conditional probability of its end $m_i^e$ given the start:

$$p(m_i|\mathcal{X};\theta_{\text{MD}}) = p(m_i^s|\mathcal{X};\theta_{\text{MD}})\; p(m_i^e|m_i^s,\mathcal{X};\theta_{\text{MD}})\;. \tag{6.2}$$

The first term is the probability that position $m_i^s$ starts a mention, and the second is the probability that the mention has size $m_i^e - m_i^s + 1$, to which we give categorical treatment.[4] Such factorization allows both for fast training and inference. During training, mentions are known. For inference, we consider only the positions for which the probability of starting a mention exceeds a threshold chosen to maximise micro-$F_1$ on the validation set.

**Entity Disambiguation** The disambiguation module learns to generate the unique name of $e_i$ autoregressively (token by token) from left to right:

$$p(e_i|m_i,\mathcal{X};\theta_{\text{ED}}) = p(\mathcal{Y}_e|m_i,\mathcal{X};\theta_{\text{ED}}) = \prod_{i=1}^{n} p(y_i|y_{<i},m_i,\mathcal{X};\theta_{\text{ED}})\;, \tag{6.3}$$

where $\mathcal{Y}_e = \{y_i\}_{i=1}^n$ is the sequence of $n$ tokens in the unique identifier of $e_i \in \mathcal{E}$. To fully exploit our design's potential for parallelism across mentions, we use a small single-layered LSTM (Hochreiter & Schmidhuber, 1997). This language model is not constrained to generating only valid entity names, besides, maximum likelihood training does not directly optimize for the correctness of the generator's ranking. To mitigate those issues, when training the model, we employ an auxiliary loss based on a discriminative classifier that assigns probability

$$p(e_i|m_i,\mathcal{X};\theta_{\text{ED}}) = \frac{\exp(f(m_i,\mathcal{X},\mathcal{Y}_e;\theta_{\text{ED}}))}{\sum_{e'\in\mathcal{E}}\exp(f(m_i,\mathcal{X},\mathcal{Y}_{e'};\theta_{\text{ED}}))}\;, \tag{6.4}$$

where $f$ is an MLP (details in section 6.4.1) and the normalization is over all entities in the KB (*i.e.,* their unique names).

[4] We limit the maximum number of tokens per span to 15 to avoid memory overhead (in the training set there is no mention with more than 12 tokens).

| Split | Documents | Mentions |
|---|---|---|
| Training | 942 | 18,540 |
| Validation | 216 | 4,791 |
| Test | 230 | 4,485 |

Table 6.1: Statistics of the AIDA-CoNLL dataset (standard splits; Hoffart et al., 2011).

**Parameter Estimation** We estimate the parameters of all components jointly as to maximize the model's likelihood given a dataset of observations using stochastic gradient descent (SGD; Robbins & Monro, 1951; Kiefer & Wolfowitz, 1952; Bottou, 2012). For the language model component, we employ length normalization (Sutskever et al., 2011; Sutskever et al., 2014) and label smoothing (Szegedy et al., 2016). All components are further regularized with dropout (Srivastava et al., 2014a). The classification loss is the negative logarithm of equation 6.4, and we approximate the normalization constant via negative sampling, with samples drawn from a candidate set specific to each training instance.

## 6.4 Experimental Setting

We use the standard English AIDA-CoNLL splits (Hoffart et al., 2011) for training, validation (*i.e.,* for doing model selection), and test. See table 6.1 for statistics of this dataset. AIDA provides full supervision for both MD and ED. We only link mentions that have a valid gold KB entity, a setting referred to as *InKB* evaluation (Röder et al., 2018). This is in line with many previous models (Luo et al., 2015; Ganea & Hofmann, 2017; Yamada et al., 2016) and all systems we compare to. As in several previous approaches, for linking we assume the availability of a pre-computed set of candidates instead of considering the whole KB. For that, we use the candidates by Pershina et al. (2015). We also use these candidates to provide negative samples for the discriminative loss during training (see equation 6.4).

### 6.4.1 Architecture details

As the document encoder, we use a Longformer (Beltagy et al., 2020). A Longformer is a RoBERTa (Liu et al., 2019) model with a limited attention window (we use 128 tokens). It has 12 layers, of which we use the first 8 (for faster computation), a hidden size of 768, 12 heads, for a total of 149M parameters. The MD modules (*i.e.,* $p(m_i^s|\mathcal{X};\theta_{\text{MD}})$ and $p(m_i^e|m_i^s,\mathcal{X};\theta_{\text{MD}})$) are both implemented as feed forward neural networks that take as inputs contextualized token embeddings. They have architecture: [LayerNorm, 128, ReLU, LayerNorm, 1]. We applied dropout of 0.1

before linear projections. The autoregressive ED module $p(y_i|y_{<i}, m_i, \mathcal{X}; \theta_{\text{ED}})$ is implemented with an LSTM. Three feed-forward NNs predict the first hidden state, the first context vector, and a vector to append to each decoding step. The predictions are a function of the start and end embeddings of a mention. All 3 FFNNs have architecture [LayerNorm, 768, ReLU, LayerNorm, 768]. The LSTM has an input size of 1536 and a hidden size of 768. The LSTM uses the shared input embedding from the Longformer encoder and an output head initialized from the Longformer. The discriminative classifier $f(m_i, \mathcal{X}, \mathcal{Y}_e; \theta_{\text{ED}})$ is a feed-forward NN that takes as an input a vector representation of a mention and the last context vector of the LSTM. The FFNN has architecture [LayerNorm, 768, ReLU, LayerNorm, 1]. Our whole model has a total of 202M parameters. We manually employ search from using Layer normalization (Ba et al., 2016) or not and the number of the Longformer layers to use.

### 6.4.2 Training details

We optimize ParallelAEL employing Adam (Kingma & Ba, 2015) with weight decay of 1e-2. We use a learning rate of 1e-4 for the Longformer and a learning rate of 1e-3 for all other components. We use a learning rate linear decay schedule for a maximum of 10,000 steps with 500 warm-up steps. We train with a batch size of 32 for a maximum of 100 epochs, and we do model selection on micro-$F_1$ on the validation set. We also optimized the threshold for the MD component with a grid search between -5 and 5 with steps 0.1 measuring micro-$F_1$ on the validation set. Training takes approximately one hour on 4 GPUs Nvidia Titan X 12 GB.

## 6.5 Results

Table 6.2 summarizes the main results of this chapter. Our method reduces the micro-$F_1$ error from the previous state-of-the-art method by 11%. The EL score can be also decomposed in Mention Detection (MD) and Entity Disambiguation (ED) scores. Our method gets an MD micro-$F_1$ score of $\approx$94 and an ED micro-$F_1$ score of $\approx$92 (note that the EL task scores a prediction as correct when both mention detection and disambiguation are done correctly). Unfortunately, most of the baselines we compare to do not report this decomposition, and thus is difficult to systematically investigate where our method stands for MD and ED scores. Nevertheless, Kolitsas et al. (2018) is the second-best system in terms of EL micro-$F_1$, and the authors reported a $\approx$89 ED micro-$F_1$. As a comparison, Broscheit (2019) reported $\approx$88 and van Hulst et al. (2020) $\approx$84 ED micro-$F_1$. This suggests that our improvement mainly comes from improving ED.

**Performance Evaluation** In table 6.3, we compare the speed of our system against the top-2 best baseline models from table 6.2. We run 3 independent

| Method | Micro-$F_1$ |
|---|---|
| Hoffart et al. (2011) | 72.8 |
| Steinmetz and Sack (2013) | 42.3 |
| Daiber et al. (2013) | 57.8 |
| Moro et al. (2014) | 48.5 |
| Piccinno and Ferragina (2014) | 73.0 |
| Kolitsas et al. (2018) | 82.4 |
| Peters et al. (2019) | 73.7 |
| Broscheit (2019) | 79.3 |
| Martins et al. (2019) | 81.9 |
| van Hulst et al. (2020)† | 80.5 |
| Févry et al. (2020a) | 76.7 |
| De Cao et al. (2021a)‡ | <u>83.7</u> |
| Kannan Ravi et al. (2021) | 83.1 |
| Ours | **85.5** |
| Ablations | |
| LM score only | 81.5 |
| Classifier score only | 81.7 |
| Beam Search w/ candidates | 84.9 |
| Beam Search w/o candidates | 49.4* |
| Subsequent work | |
| Mrini et al. (2022) | 85.7 |
| Zhang et al. (2022b) | 85.8 |

Table 6.2: $F_1$ (InKB) on the AIDA test set and some ablation of our ParallelAEL. **Bold** indicates best model and <u>underline</u> indicates previous state-of-the-art. †Results from the Wikipedia 2019 setting as opposed to the 2014 setting (older dump and fewer entities). *Our generative component has only seen a fraction of entities identifiers (≈2k compared to the KB size of ≈ 500k). ‡ discussed in chapter 4.

runs on the validation set and report the number of queries per second on GPU[5] feeding the models with one input at a time (*i.e.,* batch size of 1). For GENRE (De Cao et al., 2021a), we truncate sequences to the maximum supported length. ParallelAEL parallelizes the generation of all entity identifiers and dispenses with generating superfluous text (*i.e.,* the non-mentions) being >70 times faster than GENRE, which has to re-generate the whole source input left-to-right in order to fill in the mention-entity markup sequentially. Notably, our model is also slightly

[5] One Nvidia Titan X 12GB.

| **Method** | **# Queries / Sec** |
|---|---|
| Kolitsas et al. (2018) | $7.39_{\pm\ 5.03}$ |
| De Cao et al. (2021a) | $0.12_{\pm\ 0.08}$ |
| Ours | $\mathbf{8.69}_{\pm\ \mathbf{5.13}}$ |

Table 6.3: Inference speed of ParallelAEL model and the top-2 SOTA model from table 6.2.

faster than Kolitsas et al. (2018) which is a well-established model for EL.

**Discriminative Correction** We train with and without the discriminative correction term of equation 6.4 to appreciate its impact in results. Using only the LM component results in a 4% drop in performance: this is due to not optimizing directly for the correctness of the generator's ranking. Using the classifier alone also leads to a 4% drop in performance. Those ablations indicate that the auxiliary loss helps improve the generator's ranking.

**Beam Search vs Complete Scoring** To compare with previous work, we use pre-computed candidates for ED. This is feasible because the number of candidates to score is relatively small. However, in general, candidates might be too many and thus impractical to score them all. Thus, we test our model using Constrained Beam Search (CBS) as an approximation. When using CBS (with a beam size of 5), performance drops by $<1\%$, and micro-F1 remains higher than that of every other baseline, demonstrating that our formulation is robust even in this setting.

**Ablating Candidates** One of the benefits of the generative formulation is the ability to generate entity names (autoregressively through CBS) without the need for candidates. Thus, we test our model using CBS without candidates (*i.e.,* all entities in the KB are viable candidates). In this setting, ParallelAEL does not excel (42% drop in performance). The drop is not surprising: our generative component has only seen a fraction of entities identifiers ($1,537$ out of $\approx 500,000$ in the KB). Indeed, previous methods (*e.g.,* GENRE from chapter 4 De Cao et al., 2021a) were pre-trained on the whole Wikipedia to mitigate this issue. We do not have the computational budget to do such pre-training so we leave this for follow-up work.

## 6.6 Subsequent Work

All of the relevant subsequent work to this chapter is shared with chapter 4 and thus discussed in section 4.6. We report relevant results from subsequent work in

table 6.2.

## 6.7 Conclusion

In this chapter, we revisit the generative approach to EL seen in chapters 4 and 5 exploiting independence assumptions that enable parallelism across mentions with a shallow LSTM decoder. Despite a simple and scalable design, ParallelAEL sets a new state-of-the-art on English AIDA without a large decoder pre-training.

# Chapter 7

# How do Decisions Emerge across Layers in Neural Models? Interpretation with Differentiable Masking

**Chapter Highlights**

Attribution methods assess the contribution of inputs to the model prediction. One way to do so is *erasure*: a subset of inputs is considered irrelevant if it can be removed without affecting the prediction. Though conceptually simple, erasure's objective is intractable and approximate search remains expensive with modern deep NLP models. Erasure is also susceptible to the *hindsight bias*: the fact that an input can be dropped does not mean that the model 'knows' it can be dropped. The resulting pruning is over-aggressive and does not reflect how the model arrives at the prediction. To deal with these challenges, we introduce Differentiable Masking. DiffMask learns to mask-out subsets of the input while maintaining differentiability. The decision to include or disregard an input token is made with a simple model based on intermediate hidden layers of the analyzed model. First, this makes the approach efficient because we predict rather than search. Second, as with probing classifiers, this reveals what the network 'knows' at the corresponding layers. This lets us not only plot attribution heatmaps but also analyze how decisions are formed across network layers. We use DiffMask to study BERT models on sentiment classification and question answering.[1] In addition, we show how another work from the author of this thesis (but not presented here) (Schlichtkrull et al., 2021) applied this interpretability technique to the EntityGCN model from chapter 3.

[1] Source code available at https://github.com/nicola-decao/diffmask

## 7.1 Introduction

The power of deep neural networks typically comes at the expense of interpretability, which may prevent users from trusting predictions (Kim, 2015; Ribeiro et al., 2016a), makes it hard to detect model or data deficiencies (Gururangan et al., 2018; Kaushik & Lipton, 2018) or verify that a model is fair and does not exhibit harmful biases (Sun et al., 2019a; Holstein et al., 2019).

These challenges have motivated work on interpretability, both in NLP and generally in machine learning; see Belinkov and Glass (2019) and Jacovi and Goldberg (2020) for reviews. In this chapter, we study *post hoc interpretability* where the goal is to explain the prediction of a trained model and to reveal how the model arrives at the decision. This goal is usually approached with attribution methods (Bach et al., 2015; Shrikumar et al., 2017; Sundararajan et al., 2017), which explain the behavior of a model by assigning relevance to inputs.

One way to perform attribution is to use *erasure* where a subset of features (*e.g.,* input tokens) is considered irrelevant if it can be removed without affecting the model prediction (Li et al., 2016; Feng et al., 2018). The advantage of erasure is that it is conceptually simple and optimizes a well-defined objective. This contrasts with most other attribution methods which rely on heuristic rules to define feature salience; for example, attention-based attribution (Rocktäschel et al., 2016; Serrano & Smith, 2019; Vashishth et al., 2019) or back-propagation methods (Bach et al., 2015; Shrikumar et al., 2017; Sundararajan et al., 2017). These approaches received much scrutiny in recent years (Nie et al., 2018; Sixt et al., 2020; Jain & Wallace, 2019), as they cannot guarantee that the network is ignoring low-scored features. They are often motivated as approximations of erasure (Baehrens et al., 2010; Simonyan et al., 2014; Feng et al., 2018) and sometimes evaluated using erasure as ground-truth (Serrano & Smith, 2019; Jain & Wallace, 2019).

Despite its conceptual simplicity, subset erasure is not commonly used in practice. First, it is generally **intractable**, and beam search (Feng et al., 2018) or leave-one-out estimates (Zintgraf et al., 2017) are typically used instead. These approximations may be inaccurate. For example, leave-one-out can underestimate the contribution of features due to saturation (Shrikumar et al., 2017). More importantly, even these approximations remain very expensive with modern deep (*e.g.,* BERT-based; Devlin et al., 2019b) models, as they require multiple computation passes through the model. Second, the method is **susceptible to the hindsight bias**: the fact that a feature *can be* dropped does not mean that the model 'knows' that it can be dropped and that the feature *is not* used by the model when processing the example. This results in over-aggressive pruning that does not reflect what information the model uses to arrive at the decision. The issue is pronounced in NLP tasks (see figure 7.1d and Feng et al., 2018), though it is easier to see on an artificial example (figure 7.2a). A model is asked to predict if there are more 8s than 1s in the sequence. The erasure attributes the prediction

**Question:** Where did the Broncos practice for the Super Bowl ?
**Passage:** The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott . The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott .

(a) Integrated Gradient (Sundararajan et al., 2017).

**Question:** Where did the Broncos practice for the Super Bowl ?
**Passage:** The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott . The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott .

(b) Restricting the Flow (Schulz et al., 2020)

**Question:** Where did the Broncos practice for the Super Bowl ?
**Passage:** The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott . The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott .

(c) NLP explainer (Guan et al., 2019).

**Question:** Where did the Broncos practice for the Super Bowl ?
**Passage:** The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott . The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott .

(d) Erasure exact search optima.

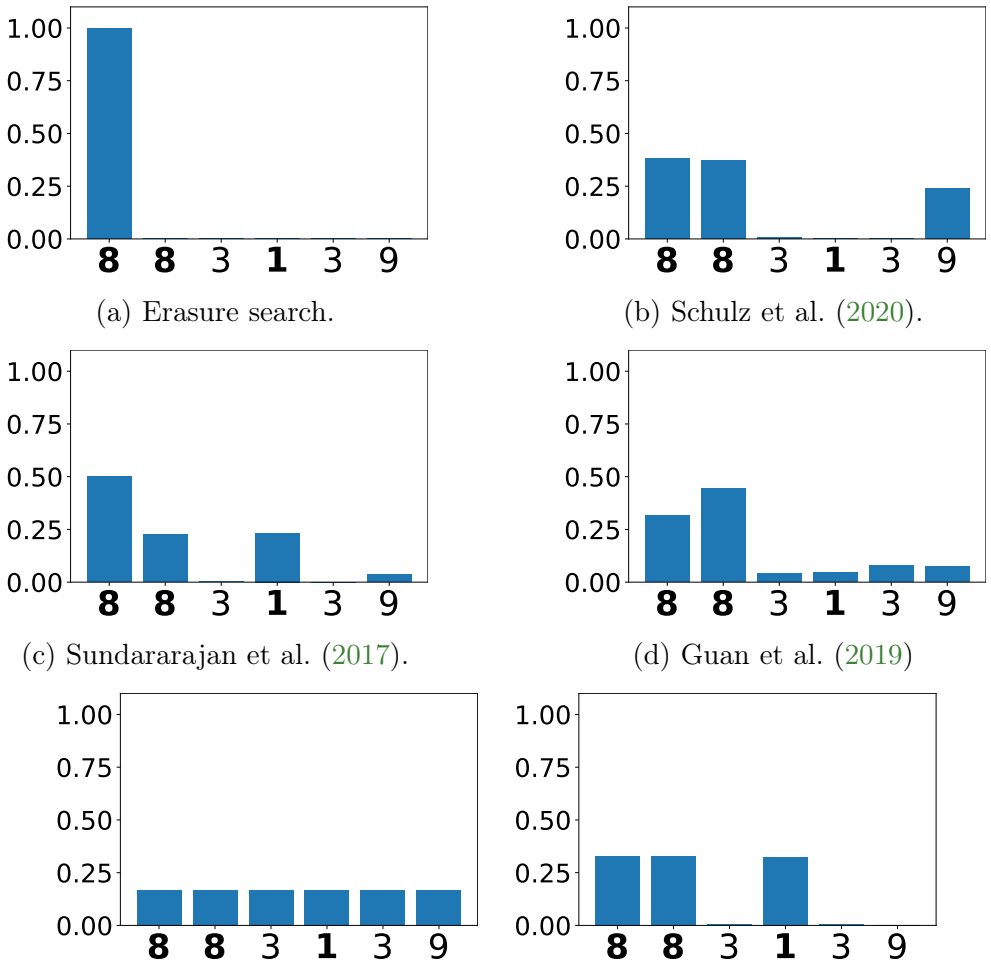
**Question:** Where did the Broncos practice for the Super Bowl ?
**Passage:** The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott . The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott .

(e) Our DiffMask.

**Question:** Where did the Broncos practice for the Super Bowl ?
**Passage:** The Panthers used the San Jose State practice facility and stayed at the San Jose Marriott . The Broncos practiced at Stanford University and stayed at the Santa Clara Marriott .

(f) Our DiffMask non-amortized.

Figure 7.1: Question answering token attribution: **(b)** and **(c)**, are misleading (*i.e.,* not faithful) as they attribute the prediction mostly to the answer span itself (underlined). Our method **(d)** reveals that the model pays attention to other named entities and the predicate 'practice' in both sentences. Predictions of the path-based methods **(a)** are more spread-out. Exact search **(e)** as well as approximate search **(f)** leads to pathological attributions.




(a) Erasure search.

(b) Schulz et al. (2020).

(c) Sundararajan et al. (2017).

(d) Guan et al. (2019)

(e) Our DiffMask conditioned on embedding layer (left) and hidden states (right).

Figure 7.2: Input attributions of several methods on a toy task: Given a sequence $x$ of digits and a query $\langle n, m \rangle$ (8 and 1 in this example) of two digits, determine whether there are more $n$ than $m$ in $x$. Attributions are computed at the vector level and normalized to sum to 1.

to a single 8 digit, as this reduced example yields the same decision as the original one. However, this does not reveal what the model was relying on: it has counted digits 8 and 1 as otherwise, it would not have achieved the perfect score on the test set.

We propose a new method, Differentiable Masking (DiffMask), which overcomes the aforementioned limitations and results in attributions that are more informative and help us understand how the model arrives at the prediction. DiffMask relies on learning sparse stochastic gates (a.k.a., masks), guaranteeing that the information from the masked-out inputs does not get propagated while maintaining end-to-end differentiability without having to resort to REINFORCE

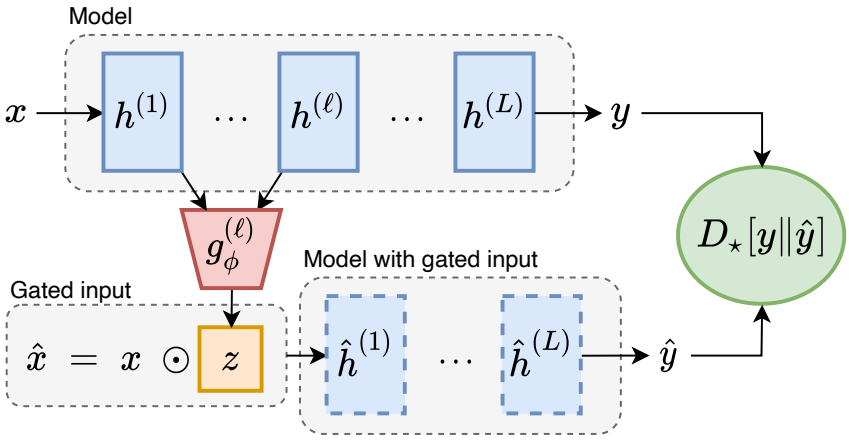



Figure 7.3: DiffMask: hidden states up to layer $\ell$ from a model (top) are fed to a classifier $g$ that predicts a mask $z$. We use this to mask the input and re-compute the forward pass (bottom). The classifier $g$ is trained to mask the input as much as possible without changing the output (minimizing a divergence $D_\star$).

(Williams, 1992). The decision to include or disregard an input token is made with a simple model based on intermediate hidden layers of the analyzed model (see figure 7.3). First, this *amortization* circumvents the need for combinatorial search making the approach efficient at test time. Second, as with probing classifiers (Adi et al., 2017; Belinkov & Glass, 2019), this reveals whether the network 'knows' at the corresponding layer what input tokens can be disregarded. During training inputs are *truly* masked whenever we sample zeros. After training, attribution scores correspond to the expectation of sampling non-zeros.

The amortization lets us not only plot attribution heatmaps, as in figure 7.1e, but also analyze how decisions are formed across network layers. In our artificial example, we see that in the bottom embedding layer the model cannot discard any tokens, as it does not 'know' which digits need to be counted (figure 7.2e, left). In the second layer, it 'knows' that these are 8s and 1s, so the rest gets discarded (figure 7.2e, right). In question answering (see figure 7.12a), where we use a 24-layer model, it takes 13–16 layers for the model to 'realize' that *'Santa Clara Marriott'* is not relevant to the question and discard it. We also adapt our method to measuring the importance of intermediate states rather than inputs. This, as we discuss later, lets us analyze which states in every layer store information crucial for making predictions, giving us insights about the information flow.

**Contributions** In this chapter we introduce DiffMask, a technique addressing limitations of attribution-based methods (especially erasure and its approximations), and demonstrate that it is stable and faithful to the analyzed models. We then use this technique to analyze BERT-based models fined-tuned on sentiment classification and question answering.

## 7.2 Background

Section 7.2.1 includes a summary of the Hard Concrete distribution which is used in this chapter. Such section is quite technical but not that it is not a prerequisite to understand our contributions.

### 7.2.1 The Hard Concrete distribution

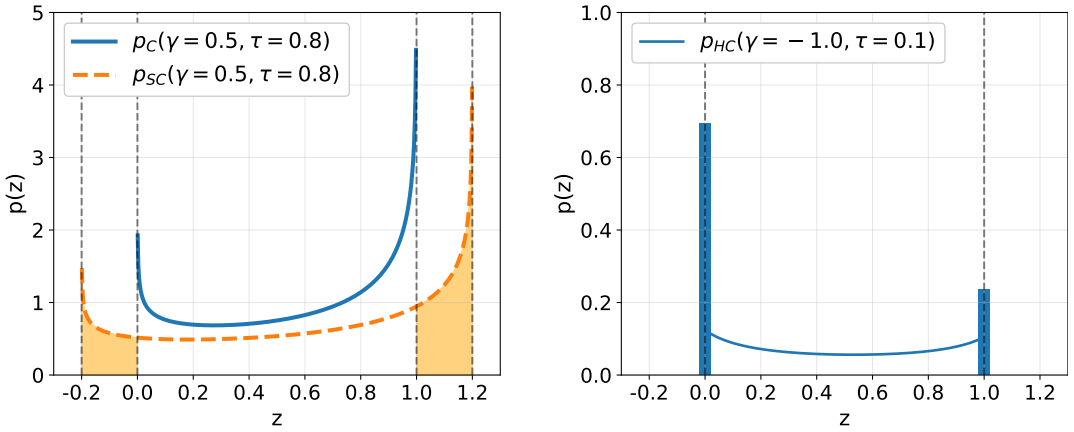
The Hard Concrete distribution, assigns density to continuous outcomes in the open interval $(0, 1)$ and non-zero mass to exactly 0 and exactly 1. A particularly appealing property of this distribution is that sampling can be done via a differentiable reparameterization (Rezende et al., 2014; Kingma & Welling, 2014). A second useful property it that one can compute in close form and in a differentiable way the expectation of sampling zeros or ones from it *i.e.,*

$$\mathbb{E}_{p_{\mathrm{z}}}\left[\mathrm{z} = 0\right] \ , \quad \text{and} \quad \mathbb{E}_{p_{\mathrm{z}}}\left[\mathrm{z} = 1\right] \ , \tag{7.1}$$

where z is a Hard Concrete random variable and $p_{\mathrm{z}}$ a Hard Concrete distribution. The gradients of these expectations can be estimated via Monte Carlo sampling without the need for REINFORCE and without introducing biases. We did modify the original Hard Concrete, though only so slightly, in a way that it gives support to samples in the half-open interval $[0, 1)$, that is, with non-zero mass only at 0. That is because we need only distinguish 0 from non-zero, and the value 1 is not particularly important.[2]

**The distribution** A stretched and rectified Binary Concrete (also known as Hard Concrete) distribution is obtained applying an affine transformation to the Binary Concrete distribution (Maddison et al., 2017; Jang et al., 2017) and rectifying its samples in the interval $[0, 1]$ (see figure 7.4). A Binary Concrete is defined over the open interval $(0, 1)$ ($p_C$ in figure 7.4a) and it is parameterised by a location parameter $\gamma \in \mathbb{R}$ and temperature parameter $\tau \in \mathbb{R}_{>0}$. The location acts as a logit and it controls the probability mass skewing the distribution towards 0 in case of negative location and towards 1 in case of positive location. The temperature parameter controls the concentration of the distribution. The Binary Concrete is then stretched with an affine transformation extending its support to $(l, r)$ with $l \leq 0$ and $r \geq 1$ ($p_{SC}$ in figure 7.4a). Finally, we obtain a Hard Concrete distribution rectifying samples in the interval $[0, 1]$. This corresponds to collapsing the probability mass over the interval $(l, 0]$ to 0, and the mass over the interval $[1, r)$ to 1 ($p_{HC}$ in figure 7.4b). This induces a distribution over the close

[2] Only a true 0 is guaranteed to completely mask an input out, while any non-zero value, however small, may leak some amount of information.




(a) (b)

Figure 7.4: Binary Concrete distributions: (a) a Concrete $p_C$ and its stretched version $p_{SC}$; (b) a rectified and stretched (Hard) Concrete $p_{HC}$.

interval $[0, 1]$ with non-zero mass at 0 and 1. Samples are obtained using

$$s = \sigma\left(\left(\log u - \log(1-u) + \gamma\right)/\tau\right) \ , \tag{7.2}$$

$$z = \min\left(1, \max\left(0, s \cdot (l - r) + r\right)\right) \ , \tag{7.3}$$

where $\sigma$ is the sigmoid function (*i.e.*, $\sigma : x \mapsto (1 + \exp(-x))^{-1}$) and $u \sim \mathcal{U}(0, 1)$. We point to the Appendix B of Louizos et al. (2018) for more information about the density of the resulting distribution and its cumulative density function.

### 7.2.2 Related Work

A large body of literature analyzed BERT and Transformed-based models. For example, Tenney et al. (2019) and van Aken et al. (2019) probed BERT layers for a range of linguistic tasks, while Hao et al. (2019) analyzed the optimization surface. Rogers et al. (2020) provides a comprehensive overview of recent BERT analysis papers.

**Perturbation-based methods** While we motivated our approach through its relation to erasure, an alternative way of looking at our approach is considering it as a *perturbation-based* method. This recently introduced class of attribution methods (Ying et al., 2019; Guan et al., 2019; Schulz et al., 2020; Taghanaki et al., 2019), instead of erasing input, injects noise. Besides back-propagation and attention-based methods discussed in the introduction, another class of interpretation methods (Murdoch & Szlam, 2017; Singh et al., 2019; Jin et al., 2020) builds on prior work in cooperative game theory (*e.g.*, the Shapley value; Shapley, 1953). These methods are not trivial to apply to a new model, as they

are architecture-specific. Their hierarchical versions (*e.g.,* Singh et al., 2019; Jin et al., 2020) also make a strong assumption about the structure of interaction (*e.g.,* forming a tree) which may affect their faithfulness. Also Chen et al. (2018) share some similarities to our work as they also do amortization but use the *Gumbel softmax trick* (Maddison et al., 2017; Jang et al., 2017) to approximate minimal subset selection. They assume that the subset contains exactly $k$ elements where $k$ is a hyperparameter. Moreover, their explainer is a separate model predicting input subsets, rather than a 'probe' on top of the model's hidden layers, and hence cannot be used to reveal how decisions are formed across layers.

**Latent rationales** There is a stream of work on learning interpretable models by means of extracting latent rationales (Lei et al., 2016; Bastings et al., 2019). Some of the techniques underlying DiffMask are related to that line of work, but overall we approach very different problems. Lei et al. (2016) use REINFORCE to minimize a downstream loss computed on masked inputs, where the masks are binary and latent. They employ $L_0$ regularization to solve the task while conditioning only on small subsets of the input regarded as a *rationale* for the prediction. To the same end, Bastings et al. (2019) minimize downstream loss subject to constraints on expected $L_0$ using a variant of the sparse relaxation of Louizos et al. (2018). In sum, they employ stochastic masks to learn an interpretable model which they learn by minimizing a downstream loss subject to constraints on $L_0$, we employ stochastic masks to interpret an existing model and for that we minimize $L_0$ subject to constraints on that model's downstream performance. In another work (Schlichtkrull et al., 2021) by the author not included in this chapter nor this thesis, we also employ stochastic masks and $L_0$ regularization for analyzing graph neural networks. We learn which edges are relevant in multi-hop question answering and graph-based semantic role labeling (Marcheggiani & Titov, 2017; De Cao et al., 2019b).

## 7.3 Method

We aim to understand how a trained model processes an input (*i.e.,* a sequence of embedded tokens) to produce an output (*e.g.,* a vector of class probabilities). Formally, first, for an input made of a set of vectors $\mathcal{X} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{n}$, we obtain the output $y = f(\mathcal{X})$ from the model along with its hidden states $\{\mathcal{H}^{(\ell)}\}_{\ell=0}^{l}$, for all of its $l$ layers where $\mathcal{H}^{(0)} = \mathcal{X}$ (note that each $\mathcal{H}^{(\ell)}$ is a set of hidden states, *e.g.,* one for each input token). We then probe the model using a shallow *interpreter* network which takes hidden states up to a certain layer $\ell$ and outputs a binary mask $\boldsymbol{z} = \{0, 1\}^n$ indicating which input tokens are necessary and which can be disregarded. To assess whether the *masked* input $\hat{\mathcal{X}} = \{\hat{\boldsymbol{x}}^{(i)}\}_{i=1}^{n}$ is sufficient, we re-feed the model with it and compute the output $\hat{y} = f(\hat{\mathcal{X}})$. As long as

$\hat{y}$ approximates the original output $y$ well, we deem the inputs masked by $\boldsymbol{z}$ unnecessary. Masking, however, as in multiplication by *zero*, makes a strong assumption about the geometry of the feature space, in particular, it assumes that the zero vector bears no information. Instead, we replace some of the inputs by a learned baseline vector $\boldsymbol{b} \in \mathbb{R}^{d_x}$, *i.e.,*

$$\hat{\boldsymbol{x}}^{(i)} = \boldsymbol{z}_i \odot \boldsymbol{x}^{(i)} + (1 - \boldsymbol{z}_i) \odot \boldsymbol{b} \quad \forall i \in [\![1, n]\!] \,, \tag{7.4}$$

where $\odot$ is the Hadamard product. See figure 7.3 for an overview of DiffMask.

Our interpreter model consists of $l + 1$ classifiers, the $\ell$-th of which conditions on the stack of hidden states up to $\mathcal{H}^{(\ell)}$ to predict binary 'votes'

$$\boldsymbol{v}^{(\ell)} = g(\mathcal{H}^{(0)}, \ldots, \mathcal{H}^{(\ell)}; \phi_\ell) \quad \text{and} \quad \boldsymbol{v}^{(\ell)} \in \{0, 1\}^n \,, \tag{7.5}$$

towards keeping or masking input tokens. Each classifier is a one-hidden-layer MLP parameterised by $\phi_\ell$, details and hyperparameters will follow. For a given depth $\ell$, the interpreter decides to mask $\boldsymbol{x}^{(i)}$ out as soon as $\boldsymbol{v}_i^{(k)} = \boldsymbol{0}$ for some $k \leq \ell$, *i.e.,* $\boldsymbol{z}_i = \prod_{k=0}^{\ell} \boldsymbol{v}_i^{(k)}$. That is, in order to deem $\boldsymbol{x}^{(i)}$ unnecessary, it is sufficient to do so based on any subset of hidden states up until $\mathcal{H}^{(\ell)}$.

Clearly, there is no direct supervision to estimate the parameters $\phi_\ell$ of the probes and the baseline $\boldsymbol{b}$, thus we borrow erasure's objective: namely, we train the probe to mask-out as many input tokens as possible constrained to keeping $f(\hat{\mathcal{X}}) \approx f(\mathcal{X})$. Since often, the output of $f$ parameterizes a likelihood (*e.g.,* a categorical distribution), we formulate the constraint in terms of a divergence $D_\star$ between the two functions' outputs. We cast this, rather naturally, in the language of constrained optimization.

**Objective** A practical way to minimize the number of non-zeros predicted by $g$ is minimizing the $L_0$ 'norm'.[3] Thus, our $\mathcal{L}_0$ loss for a datapoint $\mathcal{X}$ is defined as the total number of positions that are not masked:

$$\mathcal{L}_0(\phi_0, \ldots, \phi_l, \boldsymbol{b} | \mathcal{X}) = \sum_{i=1}^{n} \mathbf{1}_{\mathbb{R}_{\neq 0}}(\boldsymbol{z}_i) \,, \tag{7.6}$$

where $\mathbf{1}$ is the indicator function.[4] We minimize $\mathcal{L}_0$ for all data-points in a dataset $\mathcal{X} \in \mathcal{D}$ subject to a constraint that predictions from masked inputs have to be

[3] $L_0$, denoted $\|\boldsymbol{z}\|_0$ and defined as $|\{\boldsymbol{z}_i \in \boldsymbol{z} | \boldsymbol{z}_i \neq 0\}| = \sum_{i=1}^{|\boldsymbol{z}|} \mathbf{1}_{\mathbb{R}_{\neq 0}}(\boldsymbol{z}_i)$ where $\mathbf{1}$ is the indicator function[4], is the number of non-zeros entries in a vector. Contrary to $L_1$ or $L_2$, $L_0$ is not a homogeneous function and, thus, not a proper norm. However, contemporary literature refers to it as a norm, and we do so as well to avoid confusion.

[4] $\mathbf{1}_{\mathcal{A}}(x) = 1$ if $x \in \mathcal{A}$ and $\mathbf{1}_{\mathcal{A}}(x) = 0$ if $x \notin \mathcal{A}$.

similar to the original model predictions:

$$\begin{aligned} \min_{\phi_0,\ldots,\phi_l,\boldsymbol{b}} \quad & \mathcal{L}_0(\phi_0,\ldots,\phi_l,\boldsymbol{b}|\mathcal{X}) \\ \text{s.t.} \quad & D_\star[f(\mathcal{X})\|f(\hat{\mathcal{X}})] \leq m \quad \forall x \in \mathcal{D} \ , \end{aligned} \tag{7.7}$$

where margin $m \in \mathbb{R}_{>0}$ is a hyperparameter. Since non-linear constrained optimisation is generally intractable, we employ Lagrangian relaxation (Boyd et al., 2004) optimizing instead

$$\max_{\lambda} \min_{\phi_0,\ldots,\phi_l,\boldsymbol{b}} \mathcal{L}_0(\phi_0,\ldots,\phi_l,\boldsymbol{b}|\mathcal{X}) + \lambda(D_\star[f(\mathcal{X})\|f(\hat{\mathcal{X}})] - m) \ , \tag{7.8}$$

where $\lambda \in \mathbb{R}_{\geq 0}$ is the Lagrangian multiplier.

**Stochastic masks** Our objective poses two challenges: i) $L_0$ is discontinuous and has zero derivative almost everywhere, and ii) to output binary masks, $g$ needs a discontinuous output activation such as the step function. A strategy to overcome both problems is to make the binary variables stochastic and treat the objective in expectation, in which case one option is to resort to REINFORCE (Williams, 1992), another is to use a sparse relaxation to binary variables (Louizos et al., 2018; Bastings et al., 2019). As we shall see (we compare the two aforementioned options in table 7.3 and discuss them in Section 7.4.2), the latter proved more effective. Thus we opt to use the Hard Concrete distribution, a mixed discrete-continuous distribution on the closed interval $[0, 1]$. This distribution assigns a non-zero probability to exactly zero while it also admits continuous outcomes in the unit interval via the *reparameterization trick* (Kingma & Welling, 2014). We refer to Louizos et al. (2018) for details, but also provide a brief summary in section 7.2.1. With stochastic masks, the objective is computed in expectation, which addresses both sources of non-differentiability. Note that during training inputs are *truly* masked-out whenever we sample exact zeros. After training, attribution scores correspond to the expectation of sampling non-zero masks since any non-zero value corresponds to a leak of information.

**Masking hidden states** To reveal which hidden states store information necessary for realizing the prediction, we modify the probe slightly. For a given depth $\ell$, we use a mask $\boldsymbol{z}^{(\ell)} = g(\mathcal{H}^{(\ell)}; \phi_\ell)$ to replace some of the *states* in $\mathcal{H}^{(\ell)} = \{\boldsymbol{h}^{(\ell,i)}\}_{i=1}^m$ by a layer-specific $d_\ell$-dimensional baseline $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_\ell}$, *i.e.,*

$$\hat{\boldsymbol{h}}^{(\ell,i)} = \boldsymbol{z}_i^{(\ell)} \odot \boldsymbol{h}^{(\ell,i)} + (1 - \boldsymbol{z}_i^{(\ell)}) \odot \boldsymbol{b}^{(\ell)} \quad \forall i \in [\![1, m]\!] \ . \tag{7.9}$$

The resulting state $\hat{\mathcal{H}}^{(\ell)} = \{\boldsymbol{h}^{(\ell,i)}\}_{i=1}^m$ is used to replace $\mathcal{H}^{(\ell)}$ and thus to recompute subsequent states, $\hat{\mathcal{H}}^{(\ell+1)}, \ldots, \hat{\mathcal{H}}^{(l)}$, as well as the output $\hat{y}$. Here we do not aggregate 'votes' with a product because for this probe we want to discover whether hidden states are predictive of their own *usefulness.* see figure 7.5 for an overview of this variant of DiffMask.

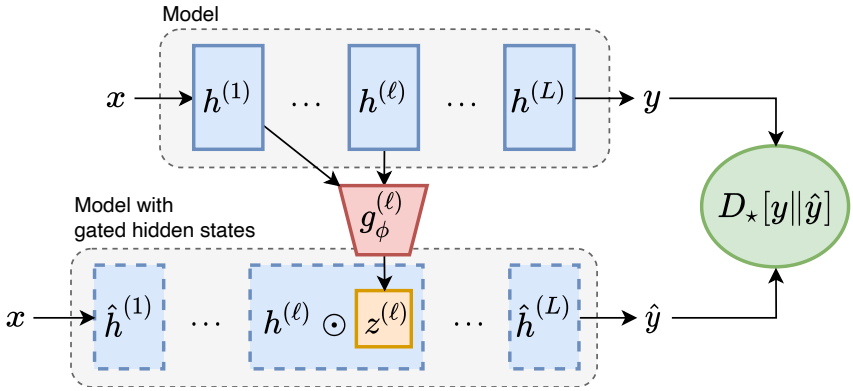



Figure 7.5: DIFFMASK for hidden states: states up to layer $\ell$ from a model (top) are fed to a classifier $g$ that predicts a mask $\boldsymbol{z}^{(\ell)}$. We use this to mask the $\ell$ih hidden state and re-compute the forward pass from that point on (bottom). The classifier $g$ is trained to mask the hidden state as much as possible without changing the output (minimizing a divergence $D_\star$).

**Probe parameterization** We parameterized the probe functions with a single layer MLP. Note that the architecture of this probe is chosen to be simple but different model choices are also possible and will not affect our general framework.[5] When masking input tokens, 'votes' $\boldsymbol{v}_i^{(\ell)}$ are computed as

$$\boldsymbol{v}_i^{(\ell)} = g(\boldsymbol{h}^{(\ell,i)}; \phi_\ell) \tag{7.10}$$

$$\boldsymbol{\gamma}_i^{(\ell)} = \xi \cdot \tanh\left(\text{NN}\left([\boldsymbol{x}^{(i)}; \boldsymbol{h}^{(\ell,i)}]; \phi_\ell\right)\right) + b^{(\ell)} , \tag{7.11}$$

$$\boldsymbol{v}_i^{(\ell)} \sim \text{HardConcrete}(\mathbf{v}_i^{(\ell)}; \boldsymbol{\gamma}_i^{(\ell)}, \tau, l, r) , \tag{7.12}$$

where $[\cdot;\cdot]$ denotes concatenation, $\xi = 10, \tau = 0.2, l = -0.2, r = 1.0$ are fixed hyperparameters. See section 7.2.1 for details about the Hard Concrete distribution including its parameterization. NN is a feed-forward neural network with architecture $[d_h/4, \tanh, 1]$ where $d_h$ is the BERT hidden size, $b^{(\ell)} \in \mathbb{R}$ are learnable biases. We use the same functional form to compute $\boldsymbol{z}^{(\ell)}$ for masking hidden states but in that case we omitted $\boldsymbol{x}^{(i)}$ from the input of the feed-forward NN. For the input probe the output of the last projection (but not the bias) is constrained to be $\in (-\xi, \xi)$ for numerical stability. We initialized the bias of the last FFNN layer to 5 to start with high probability of keeping states (fundamental for good convergence as the initialized DIFFMASK has not learned what to mask yet).

[5] In our open source implementation, we also used different architectures. Final results did not change much.

## 7.4 Experimental setting

The goal of this work is to uncover a *faithful* interpretation of an existing model, *i.e.,* revealing, as accurately as possible, the process by which the model arrives at the prediction. Human-provided labels, such as human rationales (Camburu et al., 2018; DeYoung et al., 2020), will not help us in demonstrating this, as humans cannot judge if an interpretation is faithful (Jacovi & Goldberg, 2020). More precisely, human-provided labels do not show how the model behaves – *e.g.,* annotations of what parts of the input are relevant for solving a particular task do *not* constitute a guarantee that a model relies on those parts more than others when making a prediction. When we evaluate an attribution method by comparing its outputs with human annotations, we are not measuring whether it provides faithful attributions but only if they are *plausible* according to humans. This goes against our goals as we aim to use the interpretation method to detect model deficiencies, which are usually cases where the model does not behave like humans. The ground-truth explanations of how a model makes certain predictions depend not only on the data but also on the model, and, unfortunately, are generally not known for real tasks and with complex models. This makes the evaluation and comparison of attribution methods non-trivial.

Our strategy is to i) show the effectiveness of DiffMask in a controlled setting (*i.e.,* a toy task) where ground-truth is available; ii) test the effectiveness of our relaxation for learning discrete masks (on a real model for sentiment classification); and iii) demonstrate that the method is stable and models behave the same when masking is applied. Once we have established that DiffMask can be trusted, we use it to analyze BERT-based models (Devlin et al., 2019b) fine-tuned on sentiment classification, and on question answering.

### 7.4.1 Toy task

Our toy task is defined as: given a sequence $\boldsymbol{x}$ of digits (*i.e.,* $\boldsymbol{x}_i \in [\![0, 9]\!]$), and a query $\langle n, m \rangle$ of two digits (*i.e.,* $n, m \in [\![0, 9]\!]$), determine whether $|\{\boldsymbol{x}_i \in \boldsymbol{x} : \boldsymbol{x}_i = n\}| > |\{\boldsymbol{x}_i \in \boldsymbol{x} : \boldsymbol{x}_i = m\}|$ that is whether the number of occurrences of $n$ in $\boldsymbol{x}$ are grater than the number of occurrences of $m$.

**Data** We generate sequences of varying length (up to 10 digits long) sampling each element independently: with 50% probability, we draw uniformly $n$ or $m$ and, with 50% probability, we draw uniformly from the remaining digits. We generate 10k data-points, keeping 10% of them for validation. The space of input sequences is $>10^{10}$. Thus, a model that solves the task cannot simply memorize the training set.

**Model** The query and input are embedded, concatenated, and then fed to a single-layer feed-forward NN, followed by a single-layer unidirectional GRU (Cho

et al., 2014).[6] The classification is done by a linear layer that acts on the last hidden state of the GRU. Unsurprisingly, the model solves the task almost perfectly (accuracy on test is >99%). The precise model formulation is the following: given a query $q = \langle n, m \rangle$ and an input $\boldsymbol{x} \in \{0, 9\}^k$, they are embedded as

$$\begin{aligned} \hat{\boldsymbol{n}} &= \text{Emb}_q(n) \ , \\ \hat{\boldsymbol{m}} &= \text{Emb}_q(m) \ , \\ \hat{\boldsymbol{x}}_i &= \text{Emb}_x(\boldsymbol{x}_i) \quad \forall i \in [\![1, k]\!] \ , \end{aligned} \tag{7.13}$$

where $\text{Emb}_q$ and $\text{Emb}_x$ are embedding layers of dimensionality 64. The prediction is computed as

$$\begin{aligned} \boldsymbol{h}^{(1,i)} &= \text{FFNN}([\hat{\boldsymbol{n}}; \hat{\boldsymbol{m}}; \hat{\boldsymbol{x}}^{(i)}]) \quad \forall i \in [\![1, k]\!] \ , \\ \boldsymbol{h}^{(2,0)} &= \mathbf{0}^\top \ , \\ \boldsymbol{h}^{(2,i)} &= \text{GRU}(\boldsymbol{h}^{(1,i)}, \boldsymbol{h}^{(2,i-1)}) \quad \forall i \in [\![1, k]\!] \ , \\ y &= \boldsymbol{w}^\top \boldsymbol{h}^{(2,t)} + b \ , \end{aligned} \tag{7.14}$$

where $[\cdot;\cdot]$ denotes concatenation, FFNN is a feed-forward neural network with architecture $[64 \times 3, \tanh, 2]$, GRU is a Gated Recurrent Network (Cho et al., 2014) with hidden size of 64, and $\boldsymbol{w} \in \mathbb{R}^{64}$, $b \in \mathbb{R}$ are the weight and bias parameter of the final classifier respectively.

**Attribution methods** Integrated gradient attribution (Sundararajan et al., 2017) is computed with 500 steps. Attribution of Schulz et al. (2020) is computed at token level with $\beta = 10/k$ where $k$ is the token embedding size. We optimized using the RMSprop (Tieleman & Hinton, 2012) with learning rate 1e-1 for 500 steps. Attribution of Guan et al. (2019) is computed at token level with $\lambda$ =1e-4 using RMSprop with learning rate 1e-1 for 500 steps. Our DiffMask is optimized for 100 epochs using Lookahead RMSprop (Tieleman & Hinton, 2012; Zhang et al., 2019a) with learning rate 1e-2 for $\phi, b$ and 1e-1 for $\alpha$. For these attribution methods we used our own re-implementation.

**Ground-truth for hidden-state attribution** We plot the distribution of hidden states (we use dimensionality 2, with the purpose of having a bottleneck and to support clear visualization) in figure 7.6 and observe a linear separation between states of digits present in the query and states not in the query. This means that the role of the feed-forward layer is to decide which digits to keep. Since the model solves the task, the role of the GRU must then be to count which digit occurred the most. The prediction must be attributed uniformly to *all* the hidden states corresponding to either $n$ or $m$.

[6] We use a feed-forward NN to incorporate the query information, rather than another GRU layer, to ensure that counting cannot happen in the first layer. This helps us define the

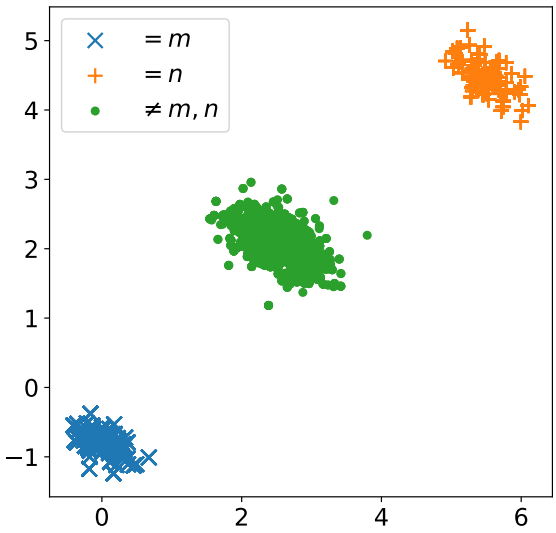



Figure 7.6: Hidden state values for the two-neuron toy task. Clusters of whether the input digit is equal to the first or second position in the query ($= n$ or $= m$ respectively) or not at all ($\neq n, m$) are completely linear separable.

### 7.4.2 Sentiment Classification

**Data** We used the Stanford Sentiment Treebank (SST; Socher et al., 2013) available here[7]. We pre-processed the data as in Bastings et al. (2019). Training and validation sets contain $8,544$ and $1,101$ sentences respectively.

**Model** For the sentiment classification experiment we downloaded[8] a pre-trained model from the Huggingface implementation[9] of Wolf et al. (2020), and we fined-tuned on the SST dataset. We report hyperparameters used for training the model and our DiffMask in table 7.1.

### 7.4.3 Question Answering

**Data** We used the Stanford Question Answering Dataset (SQuAD v1.1; Rajpurkar et al., 2016b) available here[10]. Pre-processing excluded QA pairs with more than 384 BPE tokens to avoid memory issues. After this we end up having $86,706$ training instances and $10,387$ validation instances.

ground-truth for the method.

[7] https://nlp.stanford.edu/sentiment/trainDevTestTrees_PTB.zip

[8] https://huggingface.co/transformers/pretrained_models.html

[9] https://github.com/huggingface/transformers

[10] https://rajpurkar.github.io/SQuAD-explorer

| **Original model** | **Sentiment classification** | **Question answering** |
|---|---|---|
| Type | $\text{BERT}_{\text{BASE}}$ (uncased) | $\text{BERT}_{\text{LARGE}}$ (uncased) |
| Layers | 12 | 24 |
| Hidden units | 768 | 1024 |
| Pre-trained masking | standard | whole-word |
| Optimizer | Adam* | Adam* |
| Learning rate | 3e-5 | 3e-5 |
| Train epochs | 50 | 2 |
| Batch size | 64 | 24 |
| **DiffMask model** | **Sentiment classification** | **Question answering** |
| Optimizer | Lookahead RMSprop** | Lookahead RMSprop** |
| Learning rate $\phi, b$ | 3e-4 | 3e-4 |
| Learning rate $\lambda$ | 1e-1 | 1e-1 |
| Train epochs (inputs) | 100 | 1 (per layer) |
| Train epochs (hidden) | 100 | 4 |
| Batch size | 64 | 8 |
| Constrain | $D_{KL}[y\|\hat{y}] < 0.5$ | $D_{KL}[y\|\hat{y}] < 1$ |

Table 7.1: Hyperparameters for the sentiment classification and question answering experiments. Optimizers: * Kingma and Ba (2015), ** Tieleman and Hinton (2012), Zhang et al. (2019a).

**Model** For the question answering experiment we downloaded [8] an already fine-tuned model from the Huggingface implementation[9] of Wolf et al. (2020) We report hyperparameters used by them for training the original model and the ones used for our DiffMask in table 7.1.

## 7.5 Results

### 7.5.1 Toy task

We start with an example of *input attributions*, see figure 7.2, which illustrates how DiffMask goes beyond input attribution as typically known.[11] The attribution provided by erasure (figure 7.2a) is not informative: for each datapoint the search always finds a single digit that is sufficient to maintain the original prediction and discards all the other inputs. The perturbation methods by Schulz et al. (2020) and Guan et al. (2019) (figure 7.2b and 7.2d) are also over-aggressive in pruning. They assign low attribution to some items in the query even though

[11] To enable comparison across methods, the attributions in this Section are normalized between 0 and 1.

| **Methods** | $D_{JS} \downarrow$ |
|---|---|
| Exact erasure | 0.27 |
| Integrated Gradient | 0.27 |
| Schulz et al. (2020) | 0.18 |
| Guan et al. (2019) | 0.24 |
| DiffMask | **0.00** |

Table 7.2: Toy task: attribution to hidden states, average divergence in nats between the *ground-truth* attributions and those by different methods.

those had to be considered when making the prediction. Differently from other methods, DiffMask reveals input attributions conditioned on different levels of depth. Figure 7.2e shows both input attributions according to the input itself and according to the hidden layer. It reveals that at the embedding layer there is no information regarding what part of the input can be erased: attribution is uniform over the input sequence. After the model has observed the query, hidden states predict that masking input digits other than $n$ and $m$ will not affect the final prediction: attribution is uniform over digits in the query. This reveals the role of the feed-forward layer as a filter for positions relevant to the query. Other methods do not allow for this type of inspection. These observations are consistent across the entire test set.

For *attribution to hidden states* (*i.e.,* the output of the feed-forward layer) we can compare methods in terms of how much their attributions resemble the ground-truth across the test set. Table 7.2 shows how the different approaches deviate from the gold-truth in terms of Kullback-Leibler ($D_{KL}$) and Jensen–Shannon ($D_{JS}$) divergences.[12]

### 7.5.2 Sentiment Classification

We turn now to a real task and analyze models fine-tuned for sentiment classification on the Stanford Sentiment Treebank (SST; Socher et al., 2013).

**Erasure search as learning masks** Before diving into an analysis of a BERT sentiment model, we would like to demonstrate that we can approximate the result of erasure well through our differentiable relaxations. For that, we train a single-layer GRU sentiment classifier and compare the analyses by DiffMask to solutions provided by erasure (exact search). To isolate the impact of our objective, we disable amortization, thus estimating Hard Concrete parameters for each example independently. We compare DiffMask to REINFORCE (Williams,

[12]We use $D_{KL}[p\|q]$ and $D_{JS}[p\|q]$ where $p$ is the *ground-truth* distribution and $q$ is the predicted attribution distribution.

| **Metric** | REINFORCE+ | DIFFMASK |
|---|---|---|
| Precision | 74.69 | **81.26** |
| Recall | 80.82 | **85.89** |
| $F_1$ | 73.57 | **80.75** |
| Optimality | 8.83 | **32.67** |
| $L_0$ | 33.13 | **30.58** |

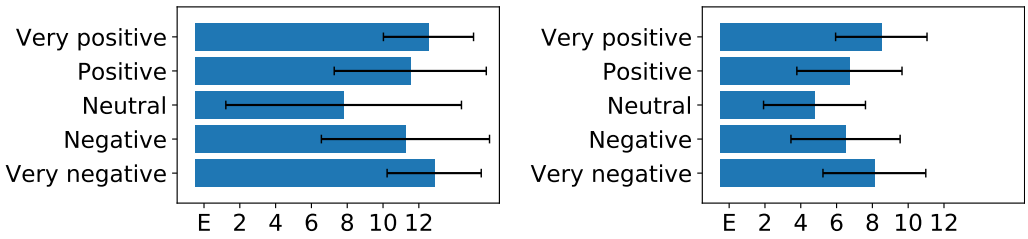
Table 7.3: Sentiment classification: optimization with DIFFMASK and REINFORCE (not amortised – with a moving average baseline for variance reduction) vs. erasure with exact search. All metrics are computed at token level; optimality is measured at sentence level.

1992) with a moving average baseline for variance reduction. Since erasure is prohibitive for long sentences, we limit our evaluation to sentences up to 25 words (54% of the data). Table 7.3 shows that DIFFMASK and REINFORCE achieve comparable levels of sparsity, but our method reaches an optimal solution much more often (33% of the times vs 9%) and is, on average, closer to an optimal solution (81% $F_1$ vs 75% $F_1$).

**Faithfulness and Plausibility** Now, we get back to the fully-amortized DIFFMASK approach applied to a 12-layers $\text{BERT}_{\text{BASE}}$ model and verify that there is no performance degradation when applying masking. The $F_1$ score of the model on the validation set moved from 37.9% to 38.3% while masking 46.3% input tokens, and to 38.9% while masking 67.6% hidden states. The explanations provided by DIFFMASK are also stable. Across 5 runs with different seeds, the standard deviation of input attributions are 0.05 and 0.03 for inputs and hidden states, respectively.

While we cannot use human labels to evaluate faithfulness of our method, comparing them and DIFFMASK attribution will tell us whether the sentiment model relies on the same cues as humans. Specifically, we compare to SST token level annotation of sentiment. in figure 7.7a, we show after how many layers on average an input token is dropped, depending on its sentiment label. This suggests that the model relies more heavily on strongly positive or negative words and, thus, is generally consistent with human judgments (*i.e., plausible*).

**Analysis** We used DIFFMASK to analyse the behavior of our BERT model. in figure 7.8, we report the average number of layers that input tokens or hidden states are kept for (or, equivalently, after how many layers they are dropped on average), aggregating by part-of-speech tags (PoS). It turns out that determinants, punctuation, and pronouns can be completely discarded from the input across all validation set, while adjectives and nouns should be kept. Also the `[CLS]`




(a) Input. (b) Hidden states.

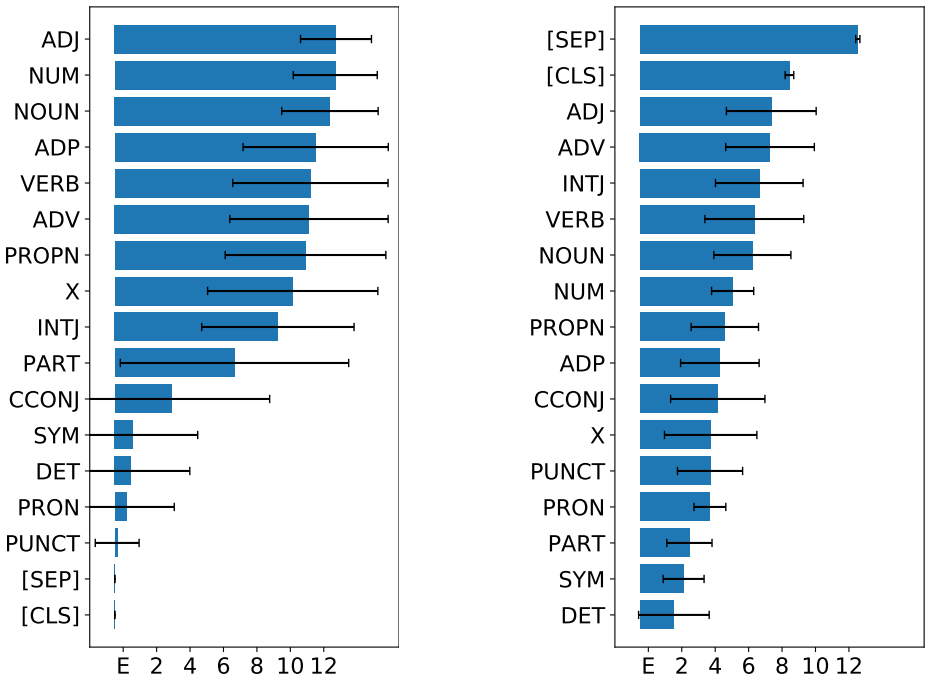
Figure 7.7: Sentiment classification: average number of layers that predict to keep input tokens or hidden states aggregated by token level sentiment annotations.

and [SEP] tokens can be ignored indicating that the model does not need such markers. Examining the POS tags distribution for hidden states leads to further conclusions. Here, the [CLS] and [SEP] tokens are the most important ones. This is not surprising as the classifier on top of BERT uses the [CLS] hidden state which gets progressively updated through all layers. Both these special tokens are not important as inputs because BERT can infer these markers in other layers, however, they are heavily used in the computation.

Figure 7.9e we show a visual example of that. We see that the model, even in the bottom layers, knows that the punctuation and both separators can be dropped from the input. This contrasts with hidden states attribution (figure 7.9f) which indicates that the separator states (especially [SEP]) are very important. By putting this information together, we can hypothesize that the separator is used to aggregate information from the sentence, relying on self-attention. In fact, this aggregation is still happening in layer 12; at the very top layers, states corresponding to almost all non-separator tokens can be dropped.

**Comparison to other methods** in figure 7.9, we visually compare different techniques on one example form validation set. While previous techniques (*e.g.,* integrated gradient) do not let us test what a model 'knows' in a given layer (*i.e.,* attribution to input conditioned on a layer), they can be used to perform attribution to hidden layers. All methods except attention correctly highlight the last hidden state of the [CLS] token as important. Its importance is due to the top-level classifier using the [CLS] hidden state. Although for DiffMask we show the expectation of keeping states, it assigns much sharper attributions. For instance, on the validation set, it assigns to the last hidden state of the [CLS] the biggest attribution 99% of the times where Schulz et al. (2020) only 71%. Raw attention (figure 7.9a) does not seem to highlight any significant patterns in that example except that start and end of sentence tokens ([CLS] and [SEP], respectively) receive more attention than the rest.[13] Attributions by Schulz et al.

[13]Voita et al. (2019a) and Michel et al. (2019) pointed out that many Transformer heads play no or minor role.

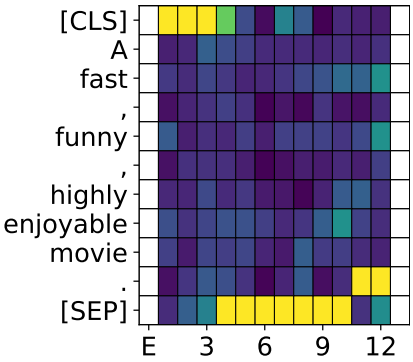



(a) Inputs. (b) Hidden states.

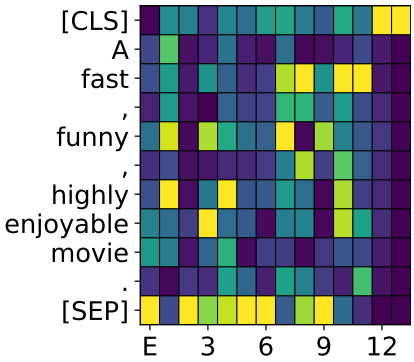
Figure 7.8: Sentiment classification: average number of layers that predict to keep input tokens (a) or hidden states (b) aggregating by part-of-speech tags (POS) and `[CLS]`, `[SEP]` tokens on validation set.

(2020) and Guan et al. (2019) assign slightly higher importance to hidden states corresponding to 'highly' and 'enjoyable', whereas it is hard to see any informative patterns provided by integrated gradient. Notice that for DiffMask, a near-zero attribution has a very clear interpretation: such a state is not used for prediction since in expectation it is dropped (not gated).

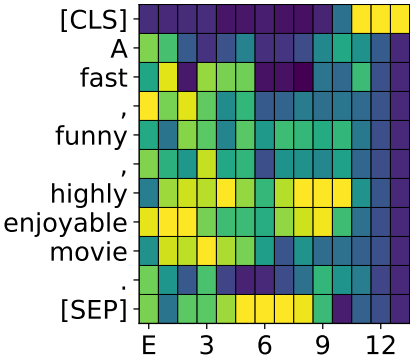
**Ablation** As argued in the introduction and shown on the toy task, many popular methods (*e.g.,* erasure and its approximations) are over-aggressive in discarding inputs and hidden units. Amortization is a fundamental component of DiffMask and is aimed at addressing this issue. in figure 7.10 we show how our method behaves when ablating amortization and thus optimizing on a single example instead. Noticeable, our method converges to masking out all hidden states at any layer (figure 7.10b). This happens as it learns an *ad hoc* baseline just for that example. When we ablate both amortization and baseline learning (figure 7.10c), the method struggles to uncover any meaningful patterns. This highlights how both core components of our method are needed in combination with each other.



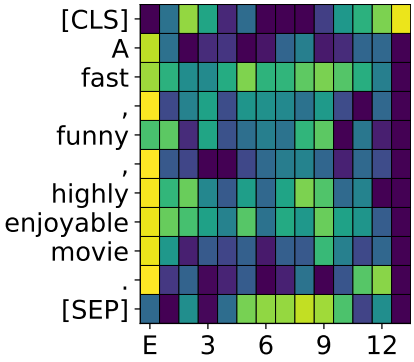

(a) Attention.



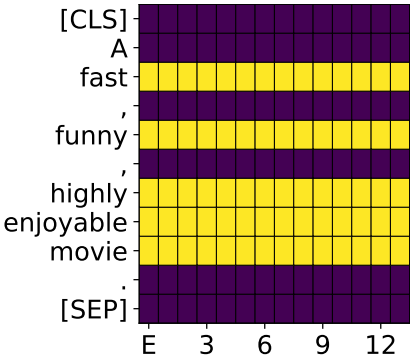

(b) Sundararajan et al. (2017).



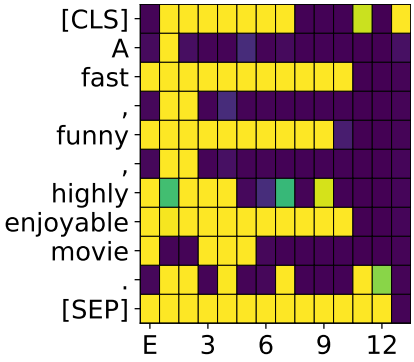

(c) Schulz et al. (2020).



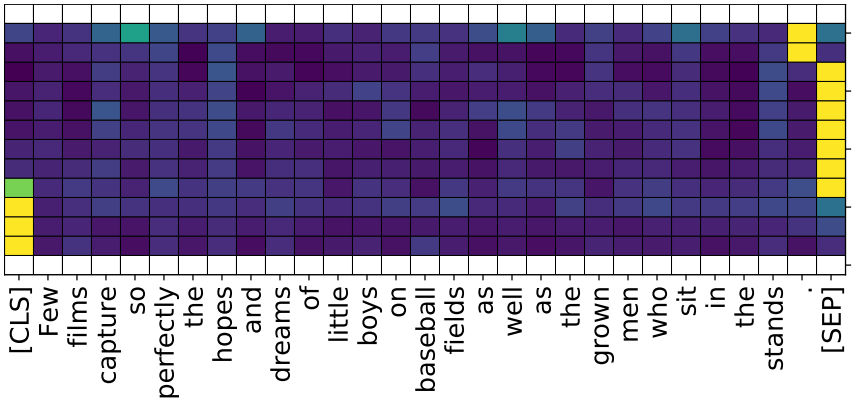

(d) Guan et al. (2019).



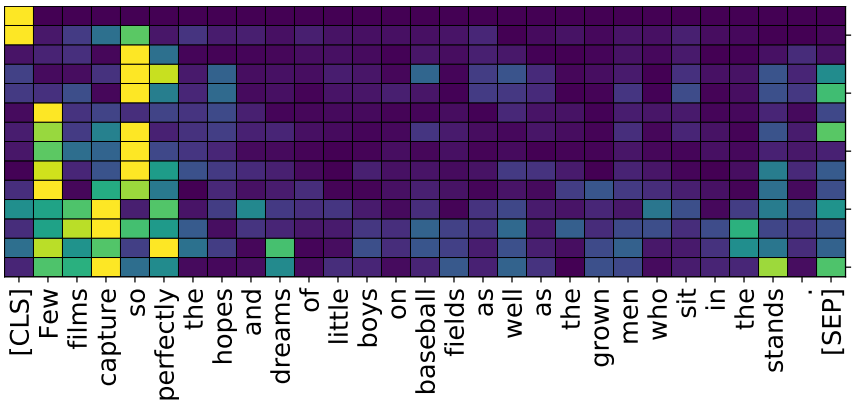

(e) DiffMask on input.



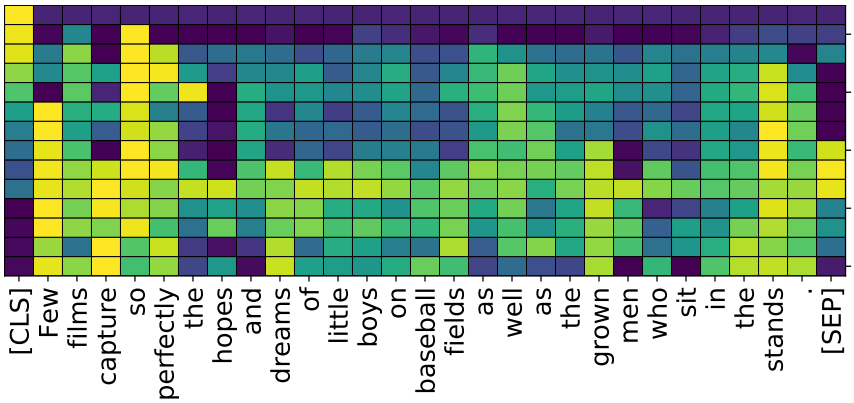

(f) DiffMask on hidden.



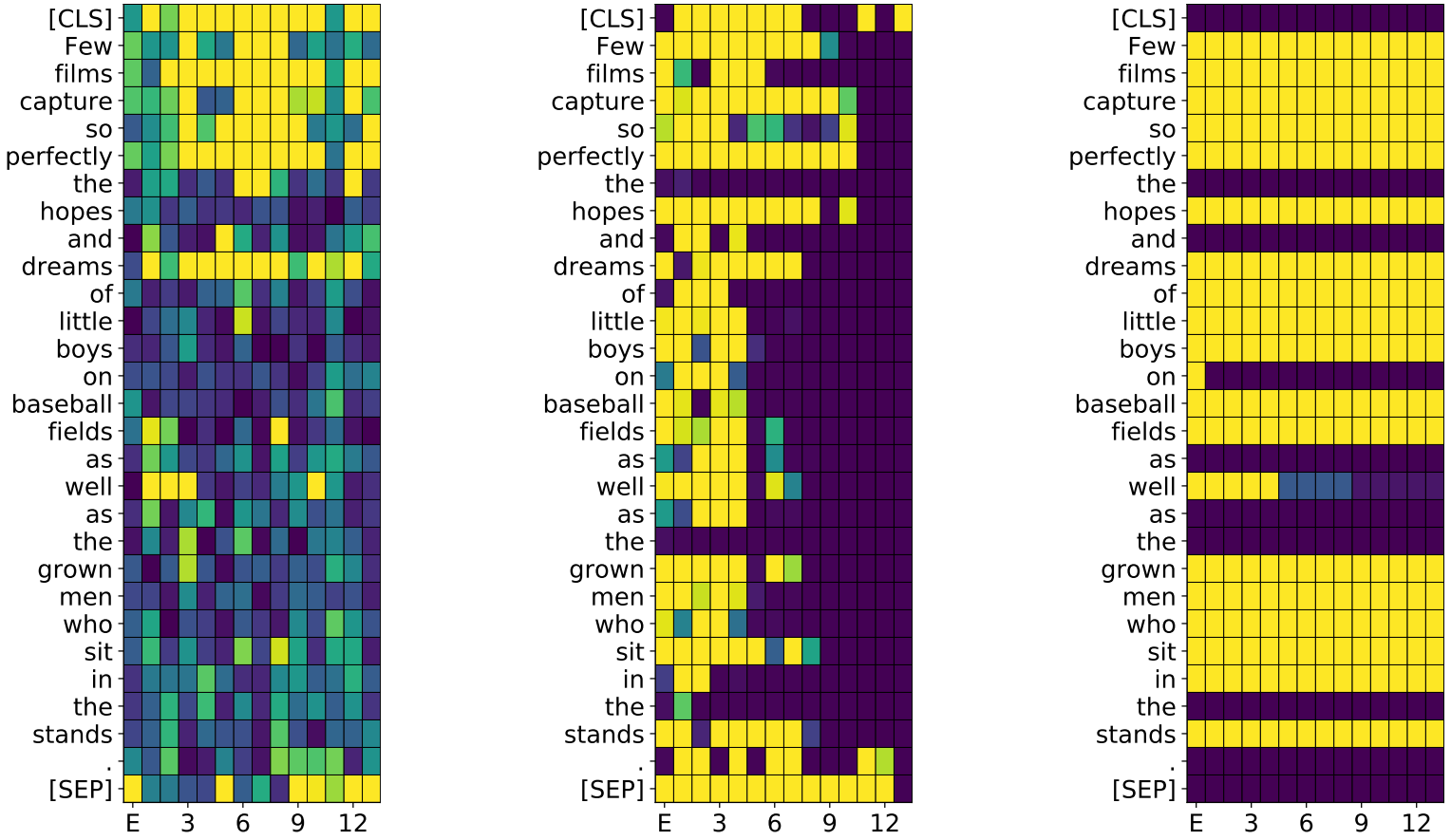

(g) Attention.




(h) Sundararajan et al. (2017).

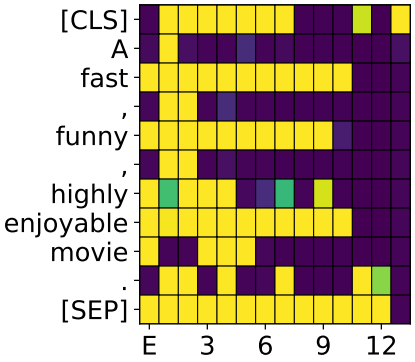



(i) Schulz et al. (2020).

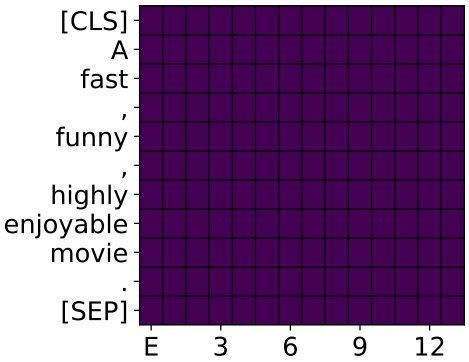



(j) Guan et al. (2019). (k) DiffMask. (l) DiffMask on input.

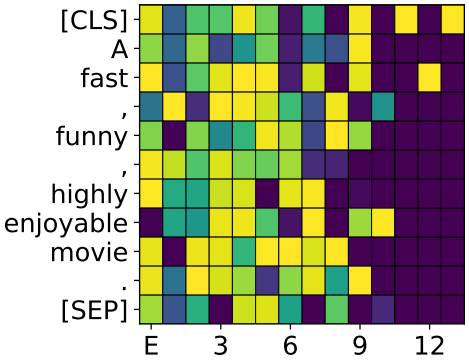
Figure 7.9: Sentiment classification: comparison between attribution method for hidden layers w.r.t. the predicted label. All plots are normalized per-layer by the largest attribution. Attention heatmap is obtained max pooling over heads and averaging across positions.




(a) Masking hidden states with amortization.

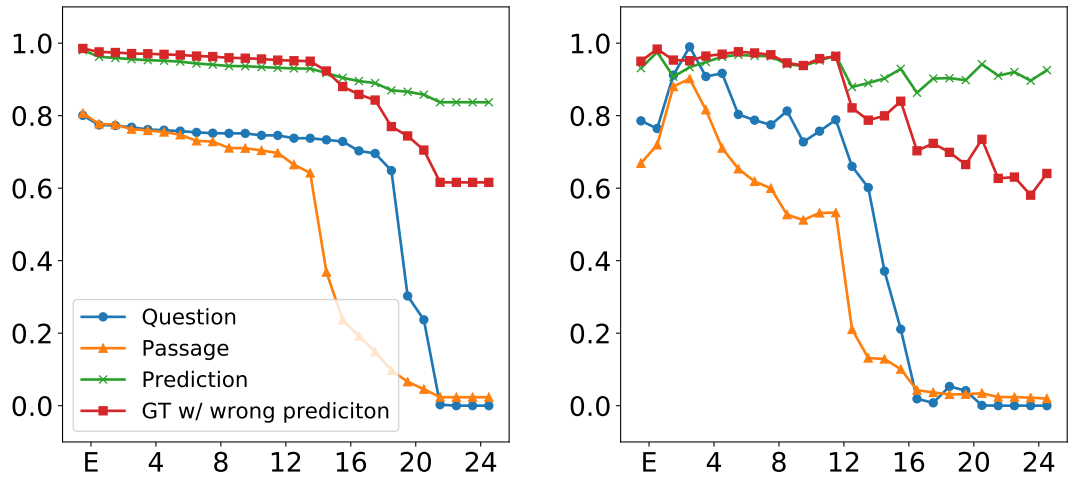



(b) Masking hidden states without amortization.




(c) Masking hidden states without amortization and without baseline.

Figure 7.10: Sentiment classification: ablation study on amortization and baseline.




(a) Input. (b) Hidden states.

Figure 7.11: Questions answering: average expectation of keeping input **(a)** and hidden states **(b)** from different layers.

### 7.5.3 Question Answering

We turn now to QA where we analyse a fine-tuned $\text{BERT}_{\text{LARGE}}$ model on the Stanford Question Answering Dataset (SQuAD v1.1; Rajpurkar et al., 2016b).

**Analysis** We start by asking DiffMask **which tokens does the model keep?** We do a similar analysis as for sentiment classification of POS tags over the entire validation set. We summarize the results in figure 7.13. It turns out that conjunctions and adpositions are dropped by the embedding and first layer, respectively, on average. On the contrary, proper nouns and punctuation are usually predicted to be dropped only after the 14-th layer. We argue that due to the pre-training objective, BERT could infer well missing parts of the input, especially if they are trivial to infer (*e.g.,* as often the case for prepositions). On the contrary, nouns and proper nouns are important as they count for 84% of the answers on SQuAD. For example, in figure 7.12a, we can see that it takes 13–16 layers for the model to 'realize' that *'Santa Clara Marriot'* is not relevant to the question and discard it.

Unlike in sentiment classification, separator tokens as well as punctuation assume a central role as inputs (*i.e.,* punctuation is considered the most important POS tag as for both questions and passages is usually dropped after the 17-th layer). Punctuation serves to demarcate sentence boundaries, useful for QA but not for sentiment classification.

Tokens from questions are generally masked by higher layers than tokens from passages as we show in figure 7.11a, which suggests that they are more important. We highlight that even in higher layers when DiffMask masks $> 95\%$ of the

tokens, the original model prediction is almost always kept $>90\%$. Noticeably, when the original BERT makes wrong predictions, the tokens annotated as the ground truth answer are kept $\approx 60\%$ of the time. This may suggest that when this happens the model still considers other options (*e.g.,* valid options such as the ground truth) as plausible, thus DiffMask detects them as important.

Now, we inspect hidden states attributions to answer **where is the information stored?** in figure 7.11b we can see a similar trend as for masking input, *i.e.,* question's hidden states are kept more on average and deeper in the computation. States on layers 2–3 are dropped less than from the embedding and first layer. This is consistent with findings of Voita et al. (2019b) which show that frequent tokens, such as determiners, accumulate contextual information. However, they are not important as inputs as we show in an example in figure 7.12b.

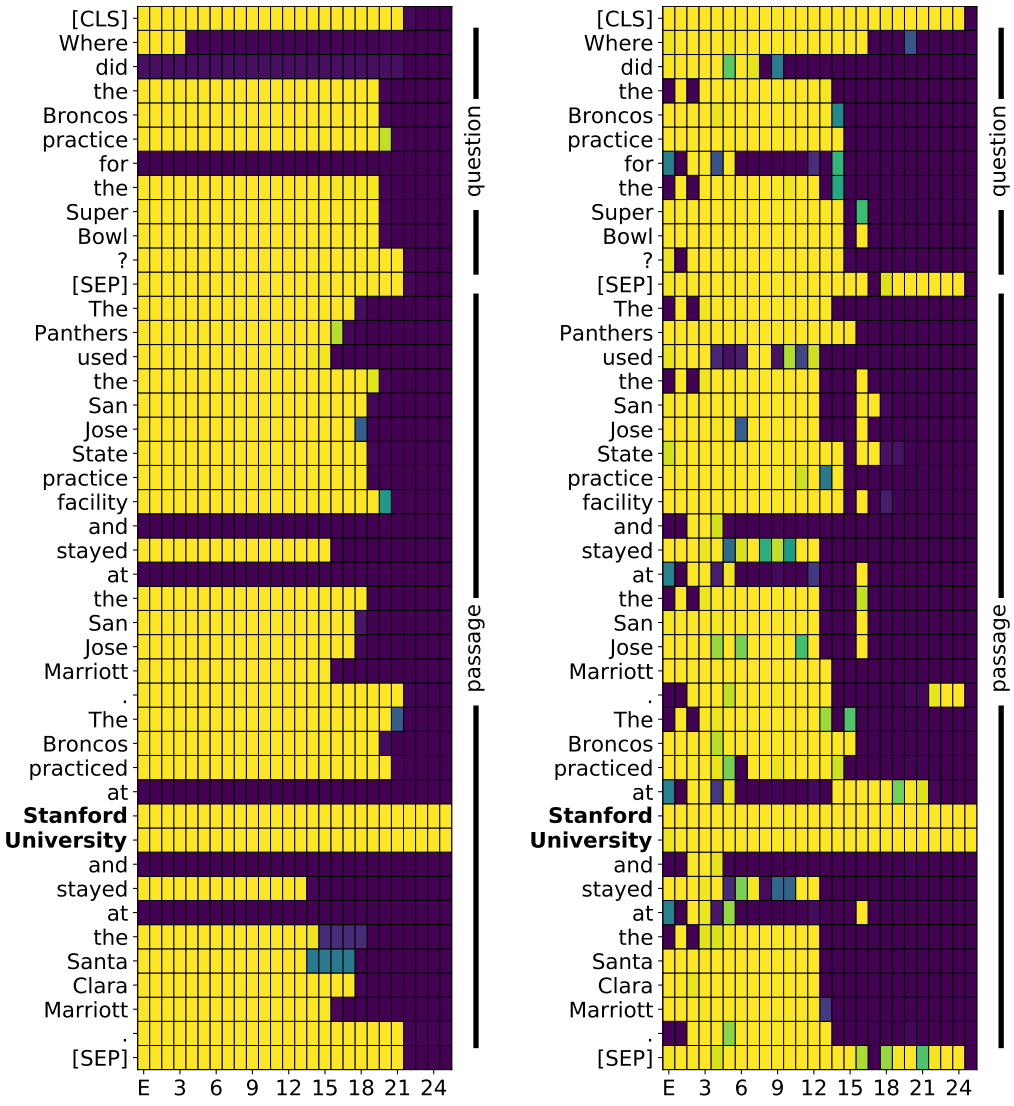
The hidden states corresponding to separator tokens are always kept across all layers except the last one across the validation set. Notice that, this token is also used as a delimiter between the question and the passage, and hence indicates where questions as well as passages end.

The level of hidden states pruning is quite incremental (after layer 3) and gets strong, after layer 9 more than 50% of them can be masked out. A steep increase in superfluous states 13–14 (visible on both parts of figure 7.11) may indicate that some states, at that point in computation, contain enough information needed for the classification while all the others can indeed be removed without affecting the model prediction. Our observation that higher layers are more predictive is in line with findings of Kovaleva et al. (2019). They pointed out that the final layers of BERT change most and are more task-specific. Again, the fact that states corresponding to the ground truth answer are still active on top layers when the model makes a wrong prediction indicates that the model is still considering different span options across top layers as well.

Finally, in figure 7.13 we report statistics on the average number of layers that predict to keep input tokens aggregating by POS tag.

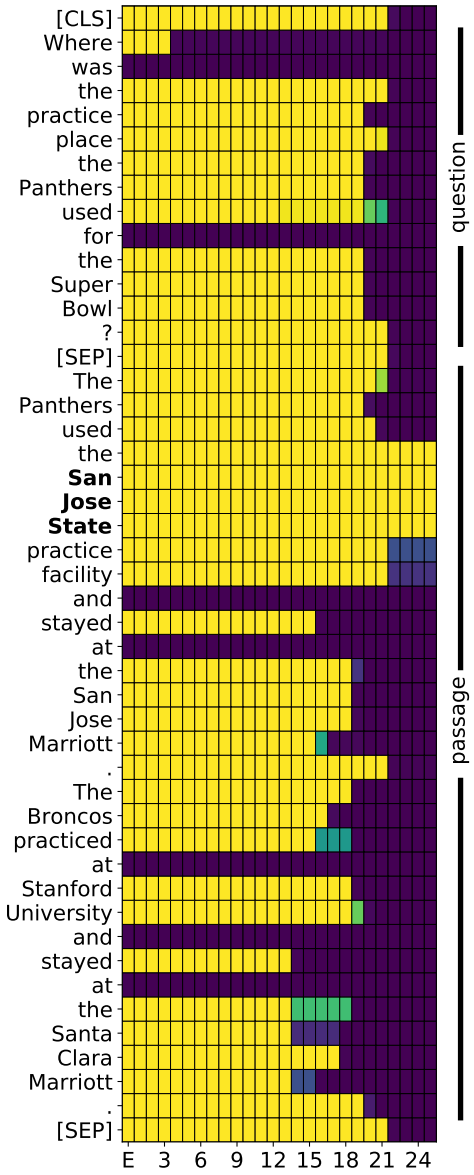
**Comparison to other methods** As we do not have access to the ground-truth, we start by contrasting DiffMask qualitatively to other attribution methods on a few examples. We highlight some common pitfalls that afflict other methods (such as the hindsight bias) and how DiffMask overcomes those. This helps demonstrate our method's faithfulness to the original model.

Figure 7.1 shows input attributions by different methods on an example from the validation set. Erasure (figure 7.1d), as expected, does not provide useful insights, it essentially singles out the answer discarding everything else including the question. This cannot be faithful and is a simple consequence of erasure's hindsight bias: when only the span that contains the answer is presented as input, the model predicts that very span as the answer, but this does not imply that the model ignores everything else when presented with the complete document as input.

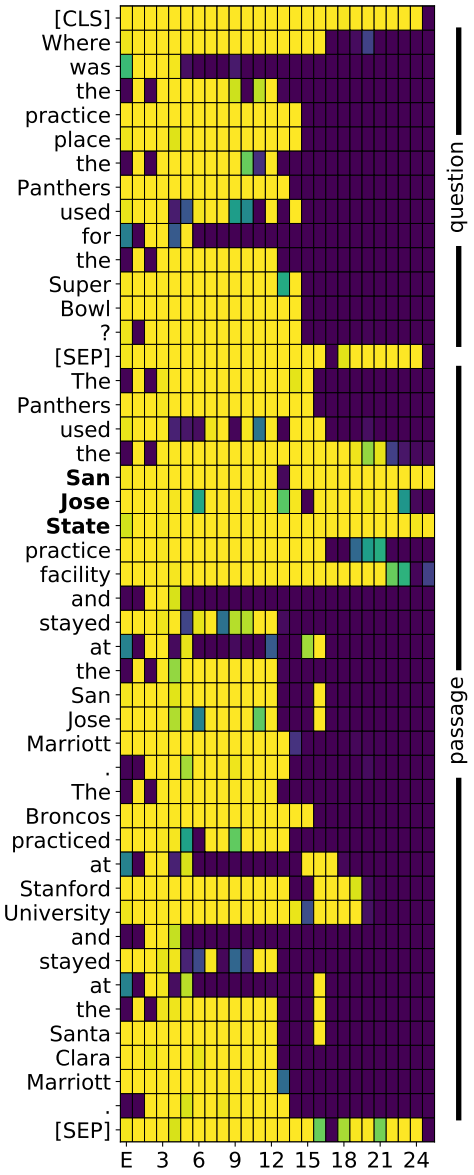



(a) Gating the input.

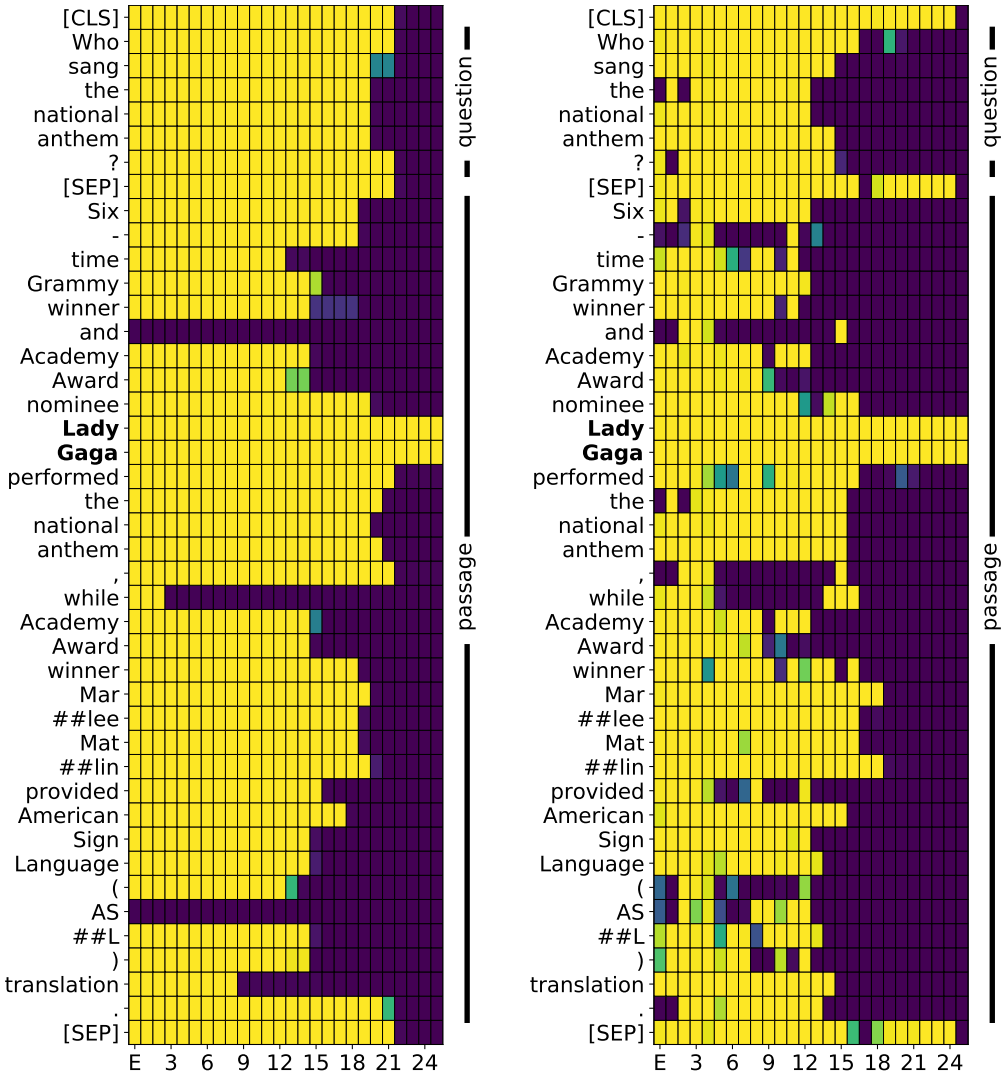
(b) Gating hidden states.




(c) Gating the input.

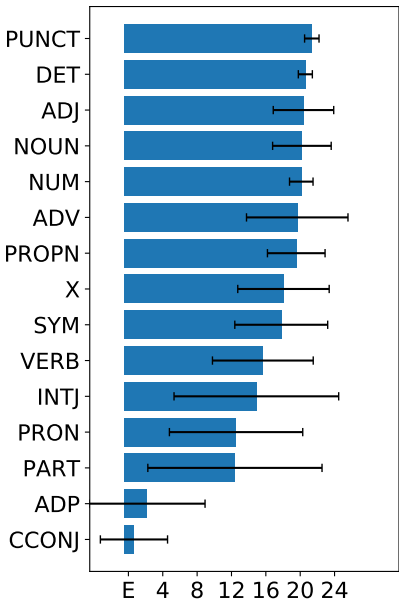



(d) Gating hidden states.

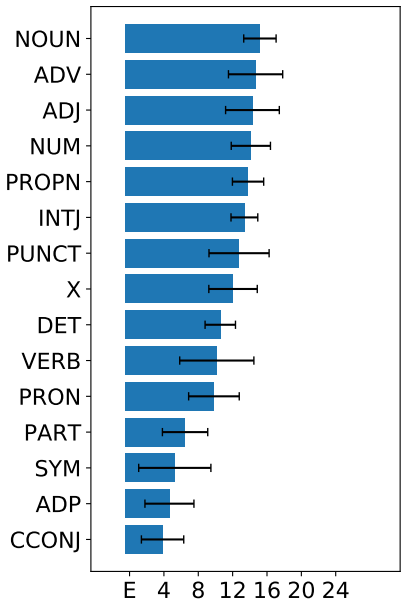



(e) Gating the input. (f) Gating hidden states.

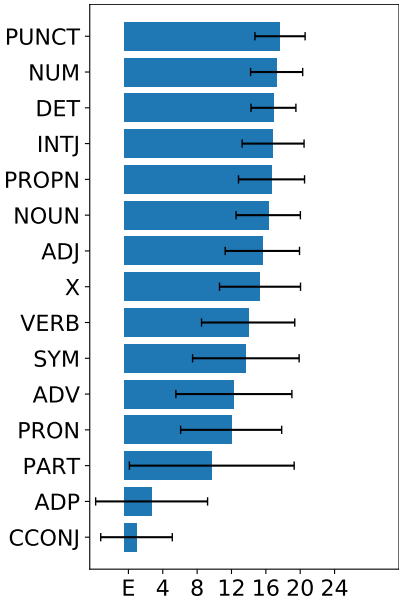
Figure 7.12: Expectation predicted by DiffMask to keep the inputs in (a) (c) and hidden states in (b) (d) on two different questions answering pairs. The correct answers is highlighted in bold.



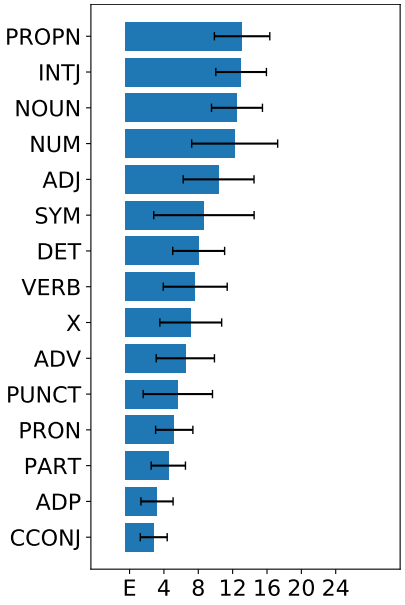

(a) Question inputs.



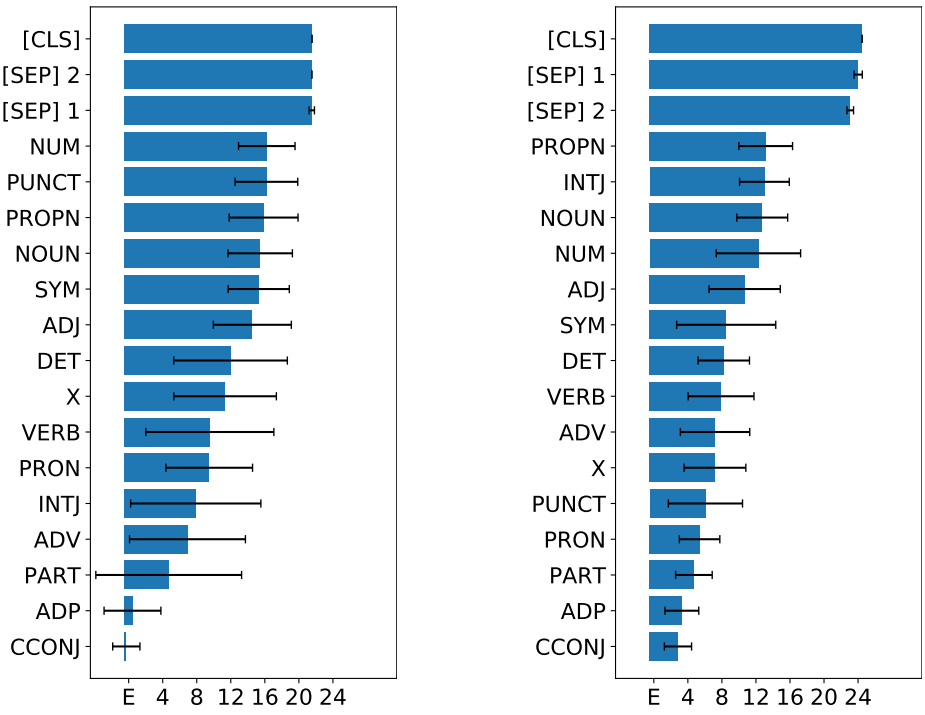

(b) Question hidden states.




(c) Context inputs.




(d) Context hidden states.




(e) All inputs. (f) All hidden states.

Figure 7.13: Question answering: average number of layers that predict to keep input tokens (a), (c) and (e) or hidden states (b), (d) and (f) aggregating by part-of-speech tag (POS) on validation set.

The methods of Schulz et al. (2020) and Guan et al. (2019) optimize attributions on single examples and thus also converge to assigning high importance mostly to words that support the current prediction and that indicate the question type. For this experiment we used Per-Sample Bottleneck attribution from Schulz et al. (2020). The authors also proposed a Readout Bottleneck where they train a second neural network to predict the mask. But differently from our formulation, they condition on subsequent layers and thus attributions are prone to the hindsight bias.

Integrated gradient does not seem to highlight any discernible pattern, which we speculate is mainly because a zero baseline is not suitable for word embeddings. Choosing a more adequate baseline is not straightforward and remains an important open issue (Sturmfels et al., 2020). Note that, DiffMask without amortization (figure 7.1f) resembles erasure (as shown in section 7.5.2 for SST).

Differently from all other methods, our DiffMask probes the network to understand what it 'knows' about the input-output mapping in different layers. in figure 7.1e we show the expectation of keeping input tokens conditioned on any one of the layers in the model to make such predictions (see figure 7.12a for a per-layer visualization). Our input attributions highlight that the model,

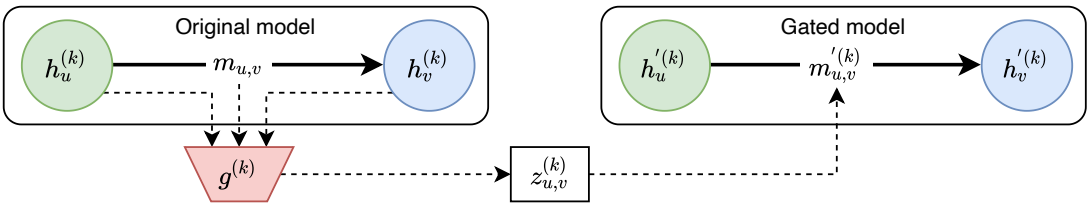



Figure 7.14: GraphMask uses vertex hidden states and messages at layer $k$ as input to a classifier $g$ that predicts a mask $z^{(\ell)}$ **(left)**. We use this to mask the messages of the $k$th layer and re-compute the forward pass with modified node states **(right)**. The classifier $g$ is trained to mask as many hidden states as possible without changing the output of the gated model. Image and caption taken with permission from the authors of Schlichtkrull et al. (2021).

in expectation across layers, *wants* to keep words in the question, the predicate 'practice' in both sentences as well as all potential candidate answers (*i.e.,* named entities). But eventually, the most important spans are in the question and the answer itself.

## 7.6 Subsequent Work

In a follow-up work, which the author of this thesis contributed to, we proposes a variation of DiffMask for interpreting graph neural networks (GraphMask; Schlichtkrull et al., 2021).[14] GraphMask interprets the predictions of GNNs identifying unnecessary edges. With a similar mechanism of DiffMask, given a trained GNN model, GraphMask learns a classifier that, for every edge in every layer, predicts if that edge can be dropped. This model also exploits the Hard Concrete distribution. Thus it is trained in a fully differentiable fashion, employing stochastic gates and encouraging sparsity through the expected $L_0$ norm. Figure 7.14 shows an outline of GraphMask.

**Using GraphMask to analyze EntityGCN**[15] In table 7.4, we investigate which edge types are used across the three layers of the EntityGCN model presented in chapter 3. EntityGCN's ablation test suggested that `COREF` edges provide marginal benefit to the model; our analysis does not entirely agree. Investigating further, we see that only 2.3% of the retained `COREF` edges overlap with `MATCH` edges (compared to 32.4% for the entire dataset). In other words, the system relies on `COREF` edges only in harder cases not handled by the surface

[14] The author of this thesis is not the main author of Schlichtkrull et al. (2021). Thus, the material presented in Schlichtkrull et al. (2021) does not list as contributions of this thesis.

[15] This whole paragraph, figures, tables, and captions are taken and adapted with permission from the authors of Schlichtkrull et al. (2021). This paragraph is not a contribution of this thesis nor an original work from the author.

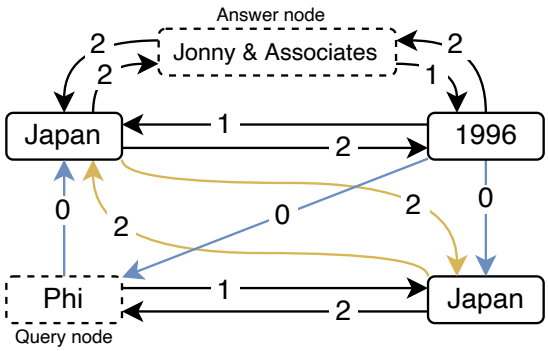



Figure 7.15: GraphMask example of subgraph of retained edges (21% of the original) for the query *"record_ label Phi"*. → is `DOC-BASED`, → is `COMPLEMENT`, and → is `MATCH` where edge labels indicate in which layer GraphMask retains such edge. GraphMask removed 21% of edges from the original graph. Best viewed in color. Image and caption taken with permission from the authors of Schlichtkrull et al. (2021).

| **Edge Type** | **k = 0** | **k = 1** | **k = 2** |
|---|---|---|---|
| `MATCH` (8.1%) | 9.4% | 11.1% | 8.9% |
| `DOC-BASED` (13.2%) | 5.9% | 17.7% | 10.7% |
| `COREF` (4.2%) | 4.4% | 0% | 0% |
| `COMPLEMENT` (73.5%) | 31.9% | 0% | 0% |
| Total (100%) | 51.6% | 28.8% | 19.6% |

Table 7.4: GraphMask retained edges for EntityGCN by layer ($k$) and type on WikiHop development set. Table and caption taken with permission from the authors of Schlichtkrull et al. (2021).

`MATCH` heuristic. The role `COMPLEMENT` edges play is interesting as well: this class represents the majority of non-superfluous edges in the bottom layer but is always superfluous in subsequent layers. The model relies on an initial propagation step across these edges, perhaps for an initial pooling of context. The EntityGCN model concatenates a representation of the query to every node in the graph before running GNN. As such, one might expect edges connecting mentions of the query entity to the rest of the graph to be superfluous. This, however, is not the case – at least one such edge is retained in 92.7% of all cases, and in 84.1% of cases in the bottom layer. We hypothesize that the model relies on GNN to see whether other mentions share a surface form or co-occur with mentions of the query entity, and, if not, how they otherwise connect to those. To investigate this, we measure the percentage of retained edges at each layer that occur on paths originating from query entities. We find that the proportion of edges that occur on paths from mentions of the query increases drastically by layer, from

11.8% at layer 0, to 42.7% at layer 1, and culminating in 73.8% in the top layer. A mention corresponding to the predicted answer is for 99.7% of examples the target of *some* retained edge. However, the chance that the predicted entity is connected to the query (72.1%) is near-identical to that of the average candidate entity (69.2%). As such, the GNN is responsible not only for propagating evidence for the predicted answer through the graph but also for propagating evidence to alternate candidates. The majority of paths take one of two forms – a `COMPLEMENT` edge followed by either a `MATCH` or a `DOC-BASED` edge (22%), or a `COMPLEMENT` edge followed by two `MATCH` or `DOC-BASED` edges (52%). `MATCH` and `DOC-BASED` edges in the bottom layer tend to represent one-hop paths rather than being the first edge on a longer path. Relations used by EntityGCN are symmetric (*e.g.,* a coreference works in both directions). A distinct feature of the subgraphs retained by GraphMask is that pairs of an edge and its inverse are *both* judged to be either superfluous or non-superfluous (individually in each layer). In figure 7.15 we report an example of edge pruning of a datapoint from WikiHop validation set. The aforementioned behavior can indeed be seen for the `DOC-BASED` edges in layer 2 between *Japan* and *Johnny & Associates*. Indeed, 49%, 98%, and 79% of retained edges in, respectively, layers 0, 1, and 2 have their inverses also retained. In other words, an 'undirected' message exchange between mentions, resulting in enriched mention representations, appears crucial.

A similar technique from this chapter is used in De Cao et al. (2021d).[16] Inspired by causal mediation analysis, they propose a method that discovers within a neural LM a small subset of neurons responsible for a particular linguistic phenomenon, *i.e.,* subsets causing a change in the corresponding token emission probabilities. They employ a Hard Concrete distribution paired with $L_0$ regularization to ensure that the search converges to discrete and sparse solutions. They analyze subject-verb number agreement and gender bias detection in LSTMs, confirming that each of these phenomena is mediated through a small subset of neurons that do not play any other discernible role.

Similar to DiffMask, Chen and Ji (2020) also proposed using word masks to interpret neural text classifiers. They build upon the information bottleneck (IB) principle (Tishby & Zaslavsky, 2015) backed by the efficient variational lower bound of the IB objective function (Alemi et al., 2017) to mask as many words as possible such that the network keeps enough information for predicting the desired label. They present an extensive evaluation of their method with three neural text classifiers (CNN, LSTM, and BERT -based) on seven classification datasets. Similarly, Chen et al. (2021) explored word-group masking for the interpretability of predictions on sentence pairs. Differently from our work, they use a biased gradient estimator for the masks (the Gumbel-softmax trick; Maddison et al., 2017; Jang et al., 2017) to address the discreteness of sampling from categorical distributions in backpropagation. They evaluate their method on tasks that

[16] Another work by the author but not presented in this thesis.

require sentence pairs as inputs, such as SNLI and paraphrase identification.

Vision DiffMask (Nalmpantis et al., 2023) is also an example of direct application of our DiffMask to the vision domain with some modification. During the forward pass, they transform each of the model's hidden states into a patch-level mask. The masks are then aggregated over the hidden layers and result in their final mask.

Finally, we point the reader to a comprehensive review interpreting deep learning models in natural language processing (Sun et al., 2021a) for further information on the current state of research in the field.

## 7.7 Conclusion

We have introduced a new *post hoc* interpretation method which learns to completely remove subsets of inputs or hidden states through masking. We circumvent an intractable search by learning an end-to-end differentiable prediction model. To overcome the hindsight bias problem, we probe the model's hidden states at different depths and amortize predictions over the training set. Faithfulness is validated in a controlled experiment pointing more clearly to some flaws of other attribution methods. We used our method to study BERT-based models on sentiment classification and question answering. DiffMask sheds light on what different layers 'know' about the input and where information about the prediction is stored in different layers.

# Chapter 8

# Editing Factual Knowledge in Language Models

**Chapter Highlights**

The factual knowledge about entities acquired during pre-training and stored in the parameters of Language Models (LMs) can be useful in downstream tasks (*e.g.,* question answering, textual inference or entity linking shown in previous chapters). However, some facts can be incorrectly induced or become obsolete over time. We present KnowledgeEditor, a method which can be used to *edit* this knowledge and, thus, fix 'bugs' or unexpected predictions without the need for expensive re-training or fine-tuning. Besides being computationally efficient, KnowledgeEditor does not require any modifications in LM pre-training (*e.g.,* the use of meta-learning). In our approach, we train a hyper-network with constrained optimization to modify a fact without affecting the rest of the knowledge; the trained hyper-network is then used to predict the weight update at test time. We show KnowledgeEditor's efficacy with two popular architectures and knowledge-intensive tasks: i) a BERT model fine-tuned for fact-checking, and ii) a sequence-to-sequence BART model for question answering. With our method, changing a prediction on the specific wording of a query tends to result in a consistent change in predictions also for its paraphrases. We show that this can be further encouraged by exploiting (*e.g.,* automatically-generated) paraphrases during training. Interestingly, our hyper-network can be regarded as a 'probe' revealing which components need to be changed to manipulate factual knowledge; our analysis shows that the updates tend to be concentrated on a small subset of components.[1]

[1] Source code available at https://github.com/nicola-decao/KnowledgeEditor

## 8.1 Introduction

Using pre-trained transformer-based Language Models (LMs; Vaswani et al., 2017; Devlin et al., 2019a; Radford et al., 2019; Lewis et al., 2020a; Raffel et al., 2020; Brown et al., 2020) has recently become a standard practice in NLP. Factual knowledge induced during pre-training can help in downstream tasks, but it can also be incorrect or become obsolete over time (*e.g.,* not reflecting changes of heads of states or country populations). Developing reliable and computationally efficient methods for bug-fixing models without the need for expensive re-training would be beneficial. See figure 8.2 for an example of revising the memory of a model that initially misremembered Namibia's capital.

Unlike conventional Knowledge Bases (KBs) that explicitly store factual knowledge in form of attributes and relations, neural models implicitly memorize world facts about entities in their parameters. One cannot easily access and interpret their computation and memories (Ribeiro et al., 2016b; Belinkov & Glass, 2019; Voita et al., 2019c; De Cao et al., 2020), thus, modifying their knowledge is a challenging problem. Motivated by practical considerations, we formulate the following desiderata for a method aimed at tackling this problem (see section 8.2.1 for a more formal treatment):

- **Generality:** be able to modify a model that was not specifically trained to be editable (*i.e.,* no need for special pre-training of LMs, such as using meta-learning) and without re-training it;
- **Reliability:** be able to successfully update a specific fact without affecting the rest of the acquired knowledge;
- **Consistency:** the changes should be consistent across equivalent formulations of a fact (*e.g.,* when asked to update an answer for one question, answers to its paraphrases should change accordingly).

The problem has been previously tackled in Zhu et al. (2020) and Sinitsin et al. (2020), as discussed in detail in section 8.2.3. However, both do not ensure that the edited model will be 'reliable', *i.e.,* that the rest of the knowledge would not be badly affected, and that the changes are 'consistent' across equivalent inputs. Additionally, Sinitsin et al. (2020) method requires expensive specialized training of the original network. While re-training the original network was feasible in their applications (*e.g.,* in machine translation), it is problematic when the network is a pre-trained LM. We propose a novel method that overcomes these limitations.

We treat editing the memories of a neural model as a *learning-to-update* problem. We use an efficient parameterization of a hyper-network that is trained to update the LM parameters when provided with a single fact that needs to be modified. We do not require meta-learning, re-training or fine-tuning of the original network. We employ constrained optimization in training: we constrain

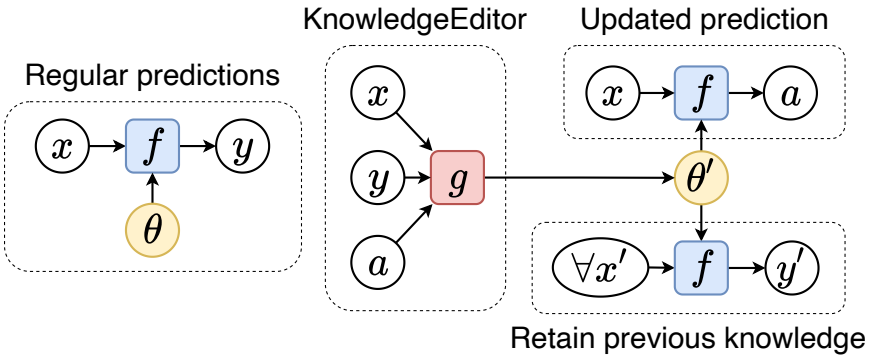


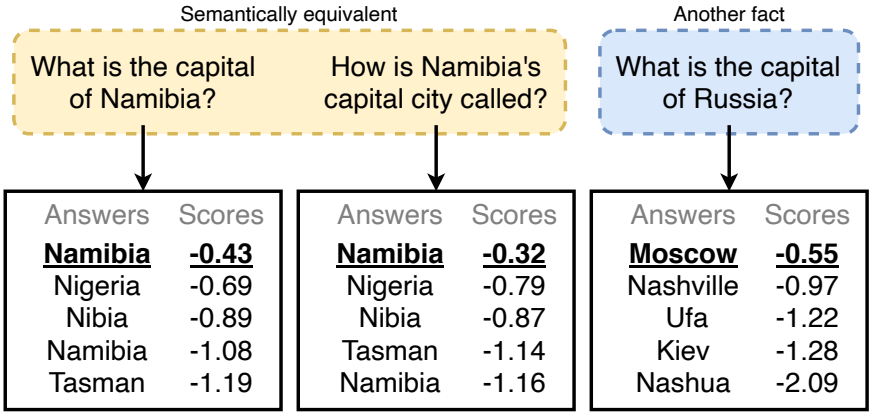

Figure 8.1: **Left:** a model $f$ with parameters $\theta$ prefers a prediction $y$ for input $x$ (*e.g.*, $y$ is the mode/argmax of a discrete distribution parameterized by $f(x;\theta)$). **Right:** our method uses a hyper-network $g$ to update the parameters of $f$ to $\theta'$ such that $f(x;\theta')$ prefers an alternative prediction $a$ without affecting the prediction $y'$ of *any* other input $x' \neq x$. Our model *edits the knowledge* about $x$ stored in the parameters of $f$.

the edited model to retain the same predictions as the original one regardless of the distance between the original and updated models in the parameter space. We show how this framework can be extended to incorporate (*e.g.,* automatically-generated) paraphrases in training, further improving consistency. figure 8.1 shows an outline of our method.

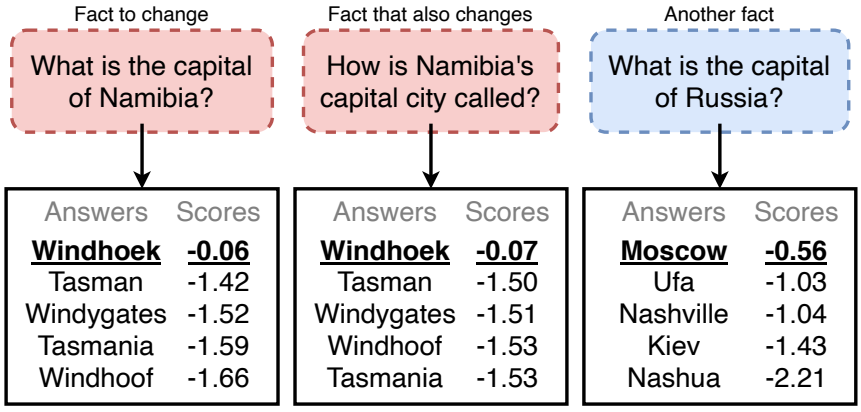
Differently from both previous methods, we do not have to select a subset of parameters to update as we let our model learn that by itself. In fact, our hyper-network can be regarded as a 'probe' revealing which components of the network need to be changed to manipulate factual knowledge, *i.e.,* revealing the 'causal mediation mechanisms' (Vig et al., 2020). We observe that the updates end up being concentrated in a restricted set of model components, even though we do not encourage any kind of sparsity. Interestingly, the most-updated components are different from the groups of parameters receiving large gradients (see figure 8.5).

**Contributions** Our contributions are as follows:

- we define the task of factual knowledge editing and propose a set of evaluation metrics;
- we propose KnowledgeEditor, a model that *learns* to modify LMs memories efficiently and reliably while maintaining consistent predictions for semantically equivalent inputs;
- we verify that our proposed method largely meets our desiderata—while other baselines based on fine-tuning fail—testing it with different LM architectures




(a) Model predictions before the update.




(b) Model predictions with edited parameters.

Figure 8.2: Predictions from a pre-trained language BART model fine-tuned for closed-book question answering. **Left:** model top-k predictions from Beam Search. **Right:** top-k after using our method conditioning on changing '*What is the capital of Namibia?*' from '*Namibia*' (wrong) to '*Windhoek*' (correct prediction). Changing one fact also changes a semantically equivalent question and keeps the predictions from other facts the same.

on knowledge-intensive tasks such as fact-checking and open-domain question answering;

- we analyze the updates for KnowledgeEditor and the alternatives.

## 8.2 Background

We want to edit the memory of a neural language model such that when, presented with an input, its output reflects a revised collection of facts. Unfortunately, the knowledge of a language model is typically opaque to us, being stored non-locally across a large number of parameters and architectural components. Thus, concretely, to operationalize the task, we seek a change in the model's parameters that affects predictions from the model only for a specific input. For a given input $x$, the prediction $a$ made by the edited model should differ from the prediction $y$ made by the original model *only* if $x$ is influenced by one of the revised facts.

### 8.2.1 Task Definition

More formally, we have a model $x \mapsto f(x;\theta)$ with trained parameters $\theta$, and a dataset of revisions $\langle x, y, a\rangle \in \mathcal{D}$, *i.e.,* $x$ is an input, $y$ is the prediction preferred by $f(x;\theta)$, and $a$ is an alternative prediction which we would like an edited version of the model to prefer. Concretely, we keep the model architecture $f$ fixed, and seek alternative parameters $\theta'$ such that for $x$, $f(x;\theta')$ would prefer the prediction $a$ instead of $y$ while keeping all other predictions unchanged. In practice, we approximate the set of 'all other predictions' using a finite data set $\mathcal{O}^x$ where $x \notin \mathcal{O}^x$ (which during validation is $\mathcal{D}$ without the triples containing $x$). Moreover, predictions need not be continuous nor differentiable outputs from the model; instead, they may result from an arbitrary decision rule based on $f(x;\theta)$. For example, when $f(x;\theta)$ parameterizes a discrete distribution $p_{\mathrm{y|x}}$ over the output space, the most standard decision rule is to output the mode of the distribution:

$$y* = \arg\max_{c \in \mathcal{Y}} \; p_{\mathrm{y|x}}(c|x;\theta) \;, \tag{8.1}$$

where $\mathcal{Y}$ is the sample space (*i.e.,* the set of all the possible outcomes of y). In text classification solving this is straightforward (for $\mathcal{Y}$ is small), in sequence-to-sequence we resort to beam search to approximate the mode (for $\mathcal{Y}$ is too large or unbounded). Note that, within this task formulation, the random variables x and y and the elements in the triples in the dataset of revisions $\mathcal{D}$ do not necessary need to be text (*i.e.,* sequence of tokens). Our task definition is general enough to comprehend any type of input/output (*e.g.,* text, images, audio/video, etc.).

**Semantically equivalent inputs** Optionally, for some revision $\langle x, y, a\rangle \in \mathcal{D}$, we may also have a set $\mathcal{P}^x$ of inputs semantically equivalent to $x$ (*e.g.,*

automatically-generated paraphrases). Such a set can be used in at least two ways: i) to obtain explicit supervision for changes that should be realized in tandem with $\langle x, y, a \rangle$; and, independently of that, ii) to evaluate whether an edited model makes consistent predictions on semantically equivalent inputs. Note that in this work we never use paraphrases at test time, only for training and evaluation of our approach; generating them at test time, while potentially helpful, would have compromised efficiency.

### 8.2.2 Evaluation

To test if a method $g$, producing edited parameters $\theta' = \theta + g(x, y, a; \phi)$, meets our desiderata, we measure:

i) *success rate*: how much $g$ successfully updates the knowledge in $\theta'$, measured as accuracy of revised predictions for inputs in $\mathcal{D}$ (*i.e.,* how many times their predictions change to the desired alternative):

$$\text{success rate} := \frac{1}{|\mathcal{D}|} \sum_{\langle x,y,a \rangle \in \mathcal{D}} \mathbf{1}_{\{a\}}(f(x; \theta')) \ , \tag{8.2}$$

where $\mathbf{1}$ is the indicator function.[2]

ii) *retain accuracy*: how well $\theta'$ retains the original predictions of $f$ (*i.e.,* how many times their predictions are equal), measured as accuracy with respect to input-output pairs in sets $\mathcal{O}^x$ (which during validation is $\mathcal{D}$ without the triples containing $x$):

$$\text{retain accuracy} := \frac{1}{|\mathcal{D}|} \sum_{\langle x,y,a \rangle \in \mathcal{D}} \frac{1}{|\mathcal{O}^x|} \sum_{x' \in \mathcal{O}^x} \mathbf{1}_{\{f(x';\theta)\}}(f(x'; \theta')) \ ; \tag{8.3}$$

iii) *equivalence accuracy*: how consistent the predictions of the revised model $\theta'$ are for semantically equivalent inputs, measured as accuracy of the revised predictions for all $\mathcal{P}^x$:

$$\text{equivalence accuracy} := \frac{1}{|\mathcal{D}|} \sum_{\langle x,y,a \rangle \in \mathcal{D}} \frac{1}{|\mathcal{P}^x|} \sum_{\hat{x} \in \mathcal{P}^x} \mathbf{1}_{\{a\}}(f(\hat{x}; \theta')) \ ; \tag{8.4}$$

iv) *performance deterioration*: how much test performance of the updated model deteriorates which assesses whether the updated model preserves the original model's overall performance (*e.g.,* performance can be accuracy, $\mathrm{F}_1$ or any other measure):

$$\text{performance deterioration} := 1 - \frac{\text{performance of } f(\cdot; \theta')}{\text{performance of } f(\cdot; \theta)} \ . \tag{8.5}$$

[2] $\mathbf{1}_{\mathcal{A}}(x) = 1$ if $x \in \mathcal{A}$ and $\mathbf{1}_{\mathcal{A}}(x) = 0$ if $x \notin \mathcal{A}$.

These values are obtained by comparing predictions of $f(\cdot;\theta)$ and $f(\cdot;\theta')$ for different subsets of inputs (*e.g.,* $\mathcal{D}$, $\mathcal{O}^x$, $\mathcal{P}^x$) and against different targets (*e.g.,* gold-standard, original predictions, or alternative predictions). While these metrics are straightforward to compute in principle, some can be computationally demanding. For example, retain accuracy depends on predictions for *all* inputs we have access to, which is potentially the entirety of the downstream task's validation/test data.[3]

Previous work has evaluated similar versions of this task differently. Sinitsin et al. (2020) measure performance deterioration and success rate but do not measure *retain accuracy* nor *equivalence accuracy*. A small performance deterioration does not guarantee high equivalence accuracy as the former is sensitive to changes in cases where the original model makes wrong decisions. Assessing accuracy against old or revised facts, which Zhu et al. (2020) also do, does not help to measure the retain accuracy. We argue that preserving model predictions for inputs in $\mathcal{O}^x$ (which during validation is $\mathcal{D}$ without the triples containing $x$) is critical in production settings, where model predictions might have been extensively analyzed and tested. For $x' \in \mathcal{O}^x$, we aim to maintain all original predictions as well as the model scores $f(x';\theta')$ itself, effectively avoiding the need to re-calibrate the models (for example, in applications where probability estimates are used downstream).

### 8.2.3 Related Work

**Modifying transformers** The most straightforward strategy to edit the knowledge of a model would be to re-train it on a new dataset with additional, modified, or removed facts. This is often unfeasible as LMs require large-scale expensive training that can hardly be reproduced by the most. Sinitsin et al. (2020) propose a meta-learning approach (Finn et al., 2017) for model modification that learns parameters that are easily *editable* at test time (*e.g.,* updating the knowledge of the model requires only a few SGD steps from these learned parameters). To have a reliable method, they employ a regularized objective forcing the updated model not to deviate from the original one. This technique suffers from three main limitations: i) it requires expensive and specialized pre-training, ii) it is sensitive to many hyper-parameters (*e.g.,* the weights of the regularizers and the subset of parameters to update), and iii) their multitask objective does not *guarantee* reliability (*i.e.,* the model is penalized for diverging from the original, rather than constrained not to).

Instead of penalizing an updated model for deviating from the original one, Zhu et al. (2020) use constrained optimization. They use a less computationally expensive procedure as they *re*-fine-tune on a specific downstream task (with altered data). Their method employs either an $L_2$ or $L_\infty$ constraint between the original model's parameters and the edited ones. However, a norm-based constraint

[3] During training of $g$, however, we can use sub-sampling (*i.e.,* mini batches) to approximate the metric.

on parameters ignores the highly non-linear nature of LMs and how parameters determine the outputs of the model. Indeed, a minimal change in parameter space may produce a completely different output for many datapoints leading to a potentially unreliable method. Additionally, they show the need to select a subset of parameters to be updated, which requires extra development effort. Zhu et al. (2020) method is similar to Elastic Weight Consolidation (Kirkpatrick et al., 2017), a technique developed for preventing catastrophic forgetting in neural network models.

**Knowledge in Language Models** Petroni et al. (2019) show that pre-trained language models recall factual knowledge without fine-tuning, which they do by feeding specific prompts to LMs. Hand-crafted prompts have been found not to be the best option to extract knowledge from LMs, and various solutions have been proposed to understand what LMs 'know' (Jiang et al., 2020; Shin et al., 2020; Liu et al., 2021). Additionally, Roberts et al. (2020) show that large models can be fine-tuned to access their internal memories to answer questions in natural language without any additional context and with surprisingly high accuracy—a setting they referred to as closed-book question answering. Although performing quite well, these models cannot reach the prediction quality of alternatives that retrieve and use context. Approaches that incentivize memorization of factual knowledge show to be beneficial for many downstream tasks suggesting that research on methods that effectively edit the memory of a model is indeed important (Zhang et al., 2019b; Sun et al., 2019b; Sun et al., 2020b). Some recent hybrid approaches that use both *implicit* and *explicit memory* show some benefits for question answering (Févry et al., 2020b; Verga et al., 2020). Notably, language models that *only* rely on internal *implicit* memory are state-of-the-art for (multilingual-) Entity Linking (De Cao et al., 2021a; De Cao et al., 2022). An effective mechanism for editing LM's implicit memory may be applicable in all these settings.

**Causal Interventions** Identification of minimal changes to neural networks needed to achieve a certain behaviour has been studied in the context of research in interpreting neural networks (Lakretz et al., 2019; Vig et al., 2020; Elazar et al., 2021; Csordás et al., 2021). The components which need to be updated can be interpreted as controlling or encoding the corresponding phenomena (*e.g.,* subject-verb agreement). Much of this research focused on modifying neuron activations rather than weights and on sparse interventions (*e.g.,* modifying one or a handful of neurons). While far from our goals, there are interesting connections with our work. For example, our analysis of updates in section 8.5.4, though very limited, may shed some light on how factual knowledge is encoded in the parameters of a model.

## 8.3 Method

We propose to treat the task of editing the memory of a neural model as a learning problem. Instead of defining a handcrafted algorithm to compute the new parameters $\theta'$, we learn a KnowledgeEditor: a model that predicts $\theta'$ conditioned on an atomic fact that we want to modify. Concretely, KnowledgeEditor is a hyper-network (Ha et al., 2017)—*i.e.,* a neural network that predicts the parameters of another network. Since the task requires every other prediction to stay the same—except the one we desire to change—we cast the learning task as a constrained optimization problem.

**Optimization** For an input $x$, changing the prediction of a model $f(\cdot;\theta)$ to $a$ corresponds to minimizing the loss $\mathcal{L}(\theta; x, a)$ incurred when $a$ is the target. Preserving the rest of the knowledge corresponds to constraining the updated parameter $\theta'$ such that model outputs $f(\cdot;\theta')$ do not change for any $x' \in \mathcal{O}^x$. Our editor $g$ is a neural network parameterized by $\phi$ which we choose by optimising the following objective for each data-point $\langle x, y, a\rangle \in \mathcal{D}$:

$$\begin{aligned} \min_{\phi} \quad & \sum_{\hat{x}\in\mathcal{P}^x} \mathcal{L}(\theta'; \hat{x}, a) \\ \text{s.t.} \quad & \mathcal{C}(\theta; \theta', f; \mathcal{O}^x) \leq m \ , \end{aligned} \tag{8.6}$$

where $\mathcal{P}^x$ is the set of semantically equivalent inputs to $x$ (for convenience we assume it contains at least $x$), $\theta' = \theta + g(x, y, a; \phi)$, $\mathcal{C}$ is a constraint on the update, and the margin $m \in \mathbb{R}_{>0}$ is a hyperparameter. The constraint is used to express our desire to preserve model outputs unchanged for $x' \neq x$. Note that only $x$, but not the rest of $\mathcal{P}^x$, are provided as input to the editor, as these will not be available at test time. In our models, $f(x;\theta)$ parameterizes a discrete distribution $p_{\text{y|x}}$ over the output sample space $\mathcal{Y}$, hence we choose to constrain updates in terms of sums of Kullback-Leibler (KL) divergences from the updated model to the original one:

$$\mathcal{C}_{KL}(\theta; \theta', f; \mathcal{O}^x) = \sum_{x'\in\mathcal{O}^x} \sum_{c\in\mathcal{Y}} p_{\text{y|x}}(c|x';\theta) \log \frac{p_{\text{y|x}}(c|x';\theta)}{p_{\text{y|x}}(c|x';\theta')} \ . \tag{8.7}$$

The constraint pushes the updated model to predict output distributions identical to the original one for all $x' \neq x$. An alternative constraint we could employ is an $L_p$ norm over the parameter updates such that $g$ is optimized to make a minimal update to the original model parameter:

$$\mathcal{C}_{L_p}(\theta; \theta', f; \mathcal{O}^x) = \left( \sum_i |\theta_i - \theta'_i|^p \right)^{1/p} \ . \tag{8.8}$$

This constraint was previously used by Zhu et al. (2020). However, such a constraint, expressed purely in parameter space and without regards to the model

architecture $f$, does not directly encourage model outputs to be close to original ones in *function space* (*i.e.,* the two functions to be similar). Neural models are highly non-linear functions, so we do not expect this type of constraint to be effective. This will be empirically demonstrated in section 8.5.

**Tractable approximations** Non-linear constrained optimization is generally intractable, thus we employ Lagrangian relaxation (Boyd et al., 2004) instead using a multiplier $\alpha \in \mathbb{R}$ and be approximated by

$$\min_{\phi} \max_{\alpha} \sum_{\hat{x} \in \mathcal{P}^x} \mathcal{L}(\theta'; \hat{x}, a) + \exp(\alpha) \cdot \big(\mathcal{C}(\theta; \theta', f; \mathcal{O}^x) - m\big) \ . \tag{8.9}$$

which can be evaluated with automatic differentiation and optimized via gradient descent. The constraint itself poses a computational challenge, as it requires assessing KL for all datapoints in the dataset at each training step. For tractability, we evaluate the constraint approximately via Monte Carlo (MC) sampling. Finally, in sequence-to-sequence models, assessing KL is intractable even for a single data point, as the sample space $\mathcal{Y}$ is unbounded. In such cases we approximate the computation on a subset of the sample space obtained via beam search.

**Architecture** Instead of predicting $\theta'$ directly, our hyper-network predicts a shift $\Delta\theta$ such that $\theta' = \theta + \Delta\theta$. A *naive* hyper-network implementation might be over-parameterized, as it requires a quadratic number of parameters with respect to the size of the target network. Thus, we apply a trick similar to Krueger et al. (2017) to make $g$ tractably predict edits for modern large deep neural networks (*e.g.,* BERT). Namely, $g$ makes use of the gradient information $\nabla_\theta \mathcal{L}(\theta; x, a)$ as it carries rich information about how $f$ accesses the knowledge stored in $\theta$ (*i.e.,* which parameters to update to increase the model likelihood given $a$).[4]

We first encode $\langle x, y, a \rangle$, concatenating the text with special separator and feeding it to a bidirectional-LSTM (Hochreiter & Schmidhuber, 1997). Then, we feed the last LSTM hidden states to a FFNN that outputs a single vector $\boldsymbol{h}$ that conditions the further computations. To predict the shift for a weight matrix $\boldsymbol{W} \in \mathbb{R}^{n \times m}$, we use five FFNNs conditioned on $h$ that predict vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^m, \boldsymbol{\gamma}, \boldsymbol{\delta} \in \mathbb{R}^n$ and a scalar $\eta \in \mathbb{R}$. Then

$$\begin{aligned} \Delta \boldsymbol{W} &= \sigma(\eta) \cdot (\boldsymbol{A} \odot \nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}; x, a) + \boldsymbol{B}) \ , \\ \text{with} \quad \boldsymbol{A} &= \text{softmax}(\boldsymbol{\alpha}) \otimes \boldsymbol{\gamma} \quad \text{and} \quad \boldsymbol{B} = \text{softmax}(\boldsymbol{\beta}) \otimes \boldsymbol{\delta} \ , \end{aligned} \tag{8.10}$$

where $\sigma$ is the sigmoid function (*i.e.,* $\sigma : x \mapsto (1+\exp(-x))^{-1}$), $\odot$ is the Hadamard product, $\otimes$ is the outer product, and softmax indicates the softmax function (*i.e.,* $\text{softmax} : \boldsymbol{x} \mapsto \exp(\boldsymbol{x}) / \sum_i \exp(\boldsymbol{x}_i)$). Note that we $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times m}$. With this

[4] A version of our hyper-network that does not use gradient information converges far too slowly.

formulation, the parameters for the hyper-network $\phi$ scale linearly with the size of $\theta$. An interpretation of equation 8.10 is that an update $\Delta \boldsymbol{W}$ is a gated sum of a scaled gradient of the objective and a bias term. The scale for the gradient and the bias are generated via an outer vector product as it allows for efficient parameterization of a matrix with just three vectors. The gate lets the model keep some parameters unchanged.

**Margin annealing** The margin $m$ is a hyperparameter and therefore fixed. However, i) it is hard to choose since it is task-dependent, and ii) it should be as small as possible. If the margin is too small, however, we risk having a small feasible set, and the model may never converge. To address both issues, we pick some initial value for the margin and anneal it during training conditioned on validation performance: when the model successfully changes $> 90\%$ of the predictions, we multiply the margin by 0.8. We stop decreasing the margin once it reaches a desirable small value. The annealing procedure prevents the model from diverging while increasingly tightening the constraint.

## 8.4 Experimental Setting

We aim to evaluate the effectiveness of KnowledgeEditor comparing to baselines on knowledge-intensive tasks where the importance of modifying the memory of a large LM has a broad impact. We then test our method on closed-book fact-checking and closed-book question answering with the metrics proposed in section 8.2.2.

### 8.4.1 Baselines

We compare against two baselines: i) fine-tuning and ii) the method proposed by Zhu et al. (2020). Fine-tuning corresponds to using standard gradient descent, minimizing the loss for the fact/prediction we want to revise. For this, we follow Sinitsin et al. (2020) and employ RMSProp (Tieleman & Hinton, 2012).[5] We set the learning rate to 1e-5 and stop upon successfully changing the output of the model or having reached a maximum of 100 gradient steps. (Zhu et al., 2020) method extends fine-tuning with an $L_\infty$ constraint on parameters.[6] Following both Sinitsin et al. (2020) and Zhu et al. (2020) we report these baselines fine-tuning all parameters or just a subset of them. We limit the search to selecting entire layers and base our decision on performance on a subset of the validation set. Note that selecting a subset of parameters for update requires an extensive search, which KnowledgeEditor dispenses with by automatically learning it.

[5] We tried alternatives like Adam but eventually RMSProp was the most effective.

[6] We search the hyper-parameter for the penalty $m \in \{$1e-3, 5e-4, 1e-4, 5e-5, 1e-5$\}$ selecting the best based on the sum of success rate and retain accuracy.

### 8.4.2 Fact-checking

We evaluate on closed-book fact-checking (FC) using the binary FEVER dataset (Thorne et al., 2018) from KILT (Petroni et al., 2021). FEVER has 104,966 training and 10,444 validation instances respectively. For every input claim $x$, the model predicts the probability $f(x;\theta)$ that it may be true. This is done *without* retrieving any evidence from a corpus, instead, just by relying on the knowledge accumulated during pre-training and encoded in its own parameters—this is similar to Lee et al. (2020) that investigate closed-book and zero-shot FC using masked-LMs. Concretely, we ask the LM to perform binary classification. We fine-tune a BERT base model (Devlin et al., 2019a) with an additional linear layer on top that maps the hidden state corresponding to the BOS (beginning of a sentence) token to the probability of the positive label. Given the available supervision, we train the architecture to maximize the model likelihood penalized by entropy regularization and weight decay. The final model has an accuracy of 77.1%.[7]

### 8.4.3 Question answering

We also evaluate on a task with a more complex sample space: closed-book question answering (QA). Here QA is treated as a sequence-to-sequence problem from question to answer without retrieving nor providing any evidence Roberts et al., 2020. This, as in FC, emphasises the role of the knowledge acquired in pre-training and encoded in the parameters of the model. For this task, we used the Zero-Shot Relation Extraction (zsRE) dataset by Levy et al. (2017). We prefer zsRE to other popular QA datasets such as SQuAD (Rajpurkar et al., 2016a), Natural Questions (Kwiatkowski et al., 2019) or TriviaQA (Joshi et al., 2017) because it is annotated with human-generated question paraphrases that we can use to evaluate our model's robustness to semantically equivalent inputs. zsRE is specifically constructed not to have relation overlaps between training and test (*i.e.,* it is zero-shot). We re-split the dataset to have the same distribution in training and test splits—we are not interested in zero-shot specifically, so we avoid the additional complexity it entails. The original zsRE dataset has 147,909 training and 3,724 validation instances respectively. After re-splitting and employing all paraphrases, we have 244,173 training and 27,644 validation instances respectively. For this task, we fine-tune a BART base model (Lewis et al., 2020a) with a standard seq2seq objective, *i.e.,* maximizing the model likelihood given the observed output sequence (Sutskever et al., 2011; Sutskever et al., 2014) and regularized with dropout (Srivastava et al., 2014a) and label smoothing (Szegedy et al., 2016). The final model has an accuracy (exact match

[7] This is comparable with what reported by Petroni et al. (2021) for a larger BART model.

between model prediction and gold standard) of 22.1%.[8]

### 8.4.4 Generating alternative predictions

Generation of alternative predictions is task-dependent as it requires producing a plausible substitute target for a given input—*e.g.,* if we need to edit the knowledge about a head of a state, a plausible substitute label should be a person, not a random (even if well-formed) string. Fact-Checking is straightforward: we simply flip the label, as it is binary classification. For QA, we exploit high-probability outcomes under the model distribution as a proxy to plausible revisions. In particular, we pick all hypotheses enumerated via beam search except the top-1.[9]

### 8.4.5 Semantically equivalent inputs

We would like the updated model to be consistent for semantically equivalent inputs (see $\mathcal{P}^x$ in section 8.2.1 and 8.3) as opposed to *just* learning a new specific and isolated datapoint. This consistency is indicative of an effective editing mechanism that taps into the knowledge stored in the model. However, not all datasets come with paraphrases of its inputs (*e.g.,* in our case FEVER does not come with paraphrases and zsRE only has paraphrases for 30% for the dataset). To this end, we generate semantically equivalent inputs using round-trip translation (Sennrich et al., 2016; Wieting & Gimpel, 2018). We employ English-to-German and German-to-English Transformer models from Marian Neural Machine Translation (MarianNMT; Junczys-Dowmunt et al., 2018) provided by Huggingface Transformers (Wolf et al., 2020). We use beam search with beam size 5 to obtain 25 paraphrases. From this set, we exclude any candidate paraphrase $\hat{x} \in \mathcal{P}^x$ of $x$ for which the prediction $\hat{y}$ supported by $f(\hat{x}; \theta)$ does not match the prediction $y$ supported by $f(x; \theta)$. This filtering ensures that, according to the current model, all paraphrases have the exact same prediction.

### 8.4.6 Architecture details

The original models we want to modify are a BERT base model (Devlin et al., 2019a) and a BART base model (Lewis et al., 2020a) for fact-checking and question answering respectively. They are both Transformer based models with 12 layers each and hidden size of 768. BERT has 12 heads, where BART has 16. They have 110M and 139M parameters respectively. BERT has a vocabulary size of $30,522$ where BART has $50,265$.

[8] This is more than reported by Petroni et al. (2021) on the original split of zsRE. That is because the original split aims at zero-shot evaluation, while we have an overlap of relation types between training and validation sets.

[9] This does not always guarantee that the alternative predictions have the same semantic type as the original one, but it is likely since the model assigns high probability to them.

KnowledgeEditor has a small single-layered bidirectional-LSTM with input size 768 and hidden size of 128. The FFNN that condenses the LSTM states follows a [256, tanh, 1024] architecture where the 5 FFNN have all a [1024, tanh, $d$] architecture where $d$ depends on the weight to modify. In our experiments, we do not use our model to modify biases, layer norms, word and positional embeddings of LMs. Overall, KnowledgeEditor has 54M and 67M parameters for BERT and BART respectively.

### 8.4.7 Training details

The original models which we want to modify are trained with a batch size of 256 using Adam (Kingma & Ba, 2015) with learning rate of 3e-5, weight decay of 1e-2, and a linear schedule with warm-up (50k total number of updates and 500 warm-up updates). We trained for a maximum of 20 epochs and employ model selection using accuracy on the validation set.[10]

KnowledgeEditor models are trained with a batch size of 1024 for FC and 256 for QA using Adam (learning rate of 3e-4 for the parameters and 1e-1 for the Lagrangian multiplier) with weight decay (1e-2) and a linear schedule with a warm-up (200k total number of updates and 1k warm-up updates). We trained for a maximum of 200 epochs and employ model selection using overall accuracy (success rate and retain accuracy) on the validation set (approximated using mini-batches).[11] The margin for the $\mathcal{C}_{KL}$ is annealed between 1e-1 and 1e-3 for the fact-checking model, and between 1e-3 and 1e-5 for the BART question answering model. For the sequence-to-sequence loss, we employ a cross-entropy loss with label smoothing of 0.1.

## 8.5 Results

Table 8.1 reports the main results for fact-checking and question answering. Overall, KnowledgeEditor achieves high performance in all metrics. Some other methods also achieve high accuracy in some metrics but always sacrificing others (*i.e.,* never meeting all our desiderata at once).

We compare methods along different metrics (as opposed to a single one), as there is no way to precisely determine the importance of each of these metrics. To gather more insight, we compute their stochastic convex combination with coefficients sampled from a Dirichlet distribution (with $\alpha = 1$ to ensure a very diverse set of combinations) and report in figure 8.4 an estimate of the probability that a system outperforms another across $1,000$ such combinations. The probability of our full method to outperform all baselines is very high for both FC and QA ($\approx$97% and $\approx$88%, respectively). In figure 8.3, we show the distributions of the

[10] On 4 Nvidia Titian X 12GB which take approximately 10 minutes for FC and 3 hours for QA.
[11] On 4 Nvidia Titian X 12GB which take approximately 1 day for FC and 3 days for QA.

combined scores (*i.e.,* the raw data for the approximation reported in figure 8.4). We then analyze different aspects of our method and the baselines.

### 8.5.1 Success rate

Every method achieves an almost perfect success rate on fact-checking. All methods but ours apply updates in a loop, stopping either when the new model is successfully updated or after reaching a maximum number of iterations. The success rate for KnowledgeEditor is not 100% because we do not apply more than one update even in case of failure. To this end, we also show an experiment with our method with multiple updates within a *loop* employing the same stopping criteria as the baselines. Note that we apply this only at test time (*i.e.,* we do not train for multiple updates). When applying multiple updates also our method reaches a 100% success rate on fact-checking and almost perfect accuracy ($>99\%$) for QA.[12]

Closed-book QA is a more challenging task since the output space is text and not just a binary label. In this setting, KnowledgeEditor achieves high accuracy ($\approx 95\%$ or $>99\%$ with the *loop*). Among all methods, KnowledgeEditor gets the best success rate while also obtaining the best retain accuracy. In QA, Zhu et al. (2020) method does not reach a good success rate ($\approx 80\%$). We searched hyperparameters for their method also to have high retain accuracy, and indeed that is higher than regular fine-tuning. However, unlike fact-checking, regular fine-tuning for QA gets almost perfect scores but at the expense of the retain accuracy. Sequence-to-sequence models are more sensitive to a slight parameter shift. This happens because minor changes may completely alter the top-k prediction from beam search (in the case of QA). Differently, in a binary classifier (in the case of FC) the probability of a prediction can change substantially without crossing the decision boundary (usually set at 0.5 when not calibrated).

### 8.5.2 Retaining previous knowledge

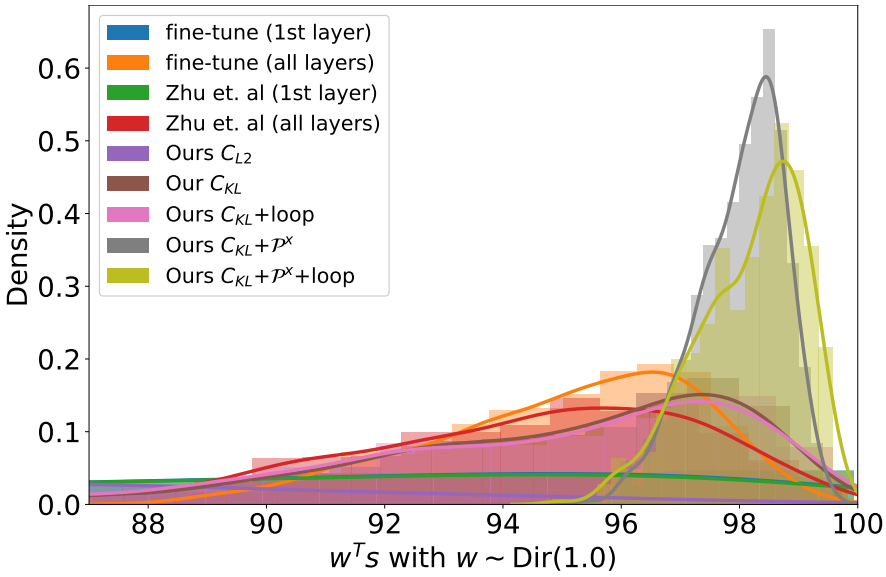
KnowledgeEditor maintains the predictions in the validation set almost perfectly (retain accuracy is $\approx 98\%$ for both FC and QA). Conversely, as expected, our method with $\mathcal{C}_{L_2}$ has very low retain accuracy (always $< 50\%$). $\mathcal{C}_{L_2}$ suffers from catastrophic forgetting because it does not enforce the updated model to be close to the original one in function space (*i.e.,* the two functions to be similar) but just in parameter space.

Fine-tuning all layers is successful but it affects the previously acquired knowledge negatively: retain accuracy is $\approx 87\%$ and $\approx 68\%$ for FC and QA, respectively, while performance deterioration in $\approx 2\%$ and $\approx 4\%$. Fine-tuning a single layer is

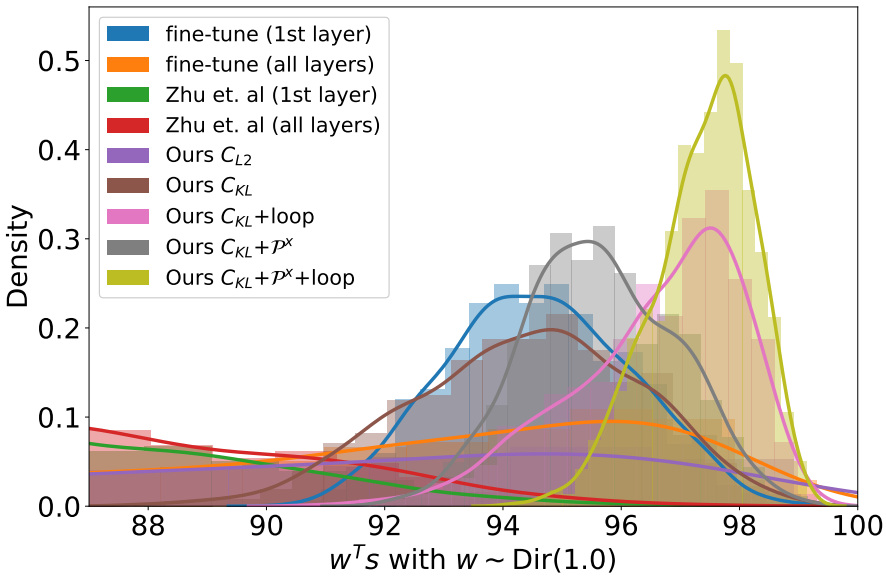
[12] Even if we do not train for multiple subsequent updates, its success opens the possibility to add this at training time. We leave the exploration of this technique to future work.

| Method | Fact-Checking | | | | Question Answering | | | |
|---|---|---|---|---|---|---|---|---|
| | **Success rate ↑** | **Retain acc. ↑** | **Equiv. acc. ↑** | **Perform. det. ↓** | **Success rate ↑** | **Retain acc. ↑** | **Equiv. acc. ↑*** | **Perform. det. ↓** |
| Fine-tune (1st layer) | 100.0 | 99.44 | 42.24 | 0.00 | 98.68 | 91.43 | 89.86 / 93.59 | 0.41 |
| Fine-tune (all layers) | 100.0 | 86.95 | 95.58 | 2.25 | 100.0 | 67.55 | 97.77 / 98.84 | 4.50 |
| Zhu et al. (2020) (1st layer) | 100.0 | 99.44 | 40.30 | 0.00 | 81.44 | 92.86 | 72.63 / 78.21 | 0.32 |
| Zhu et al. (2020) (all layers) | 100.0 | 94.07 | 83.30 | 0.10 | 80.65 | 95.56 | 76.41 / 79.38 | 0.35 |
| Ours $\mathcal{C}_{L_2}$ | 99.10 | 45.10 | 99.01 | 35.29 | 99.10 | 46.66 | 97.16 / 99.24 | 9.22 |
| KnowledgeEditor | 98.80 | 98.14 | 82.69 | 0.10 | 94.65 | 98.73 | 86.50 / 92.06 | 0.11 |
| + loop$^{\dagger}$ | 100.0 | 97.78 | 81.57 | 0.59 | 99.23 | 97.79 | 89.51 / 96.81 | 0.50 |
| + $\mathcal{P}^x$ $^{\ddagger}$ | 98.50 | 98.55 | 95.25 | 0.24 | 94.12 | 98.56 | 91.20 / 94.53 | 0.17 |
| + $\mathcal{P}^x$ + loop$^{\ddagger}$ | 100.0 | 98.46 | 94.65 | 0.47 | 99.55 | 97.68 | 93.46 / 97.10 | 0.95 |

Table 8.1: Accuracy scores on fact-checking and question answering for the metrics presented in section 8.2.2. *We report both the accuracy on the set of generated paraphrases (left) and human-annotated (right).$^{\dagger}$Apply updates in a loop, stopping when the update is a success or when reaching a maximum number of iterations (only at test time). $^{\ddagger}$Using paraphrases (semantically equivalent inputs) as additional supervision (only at training time).

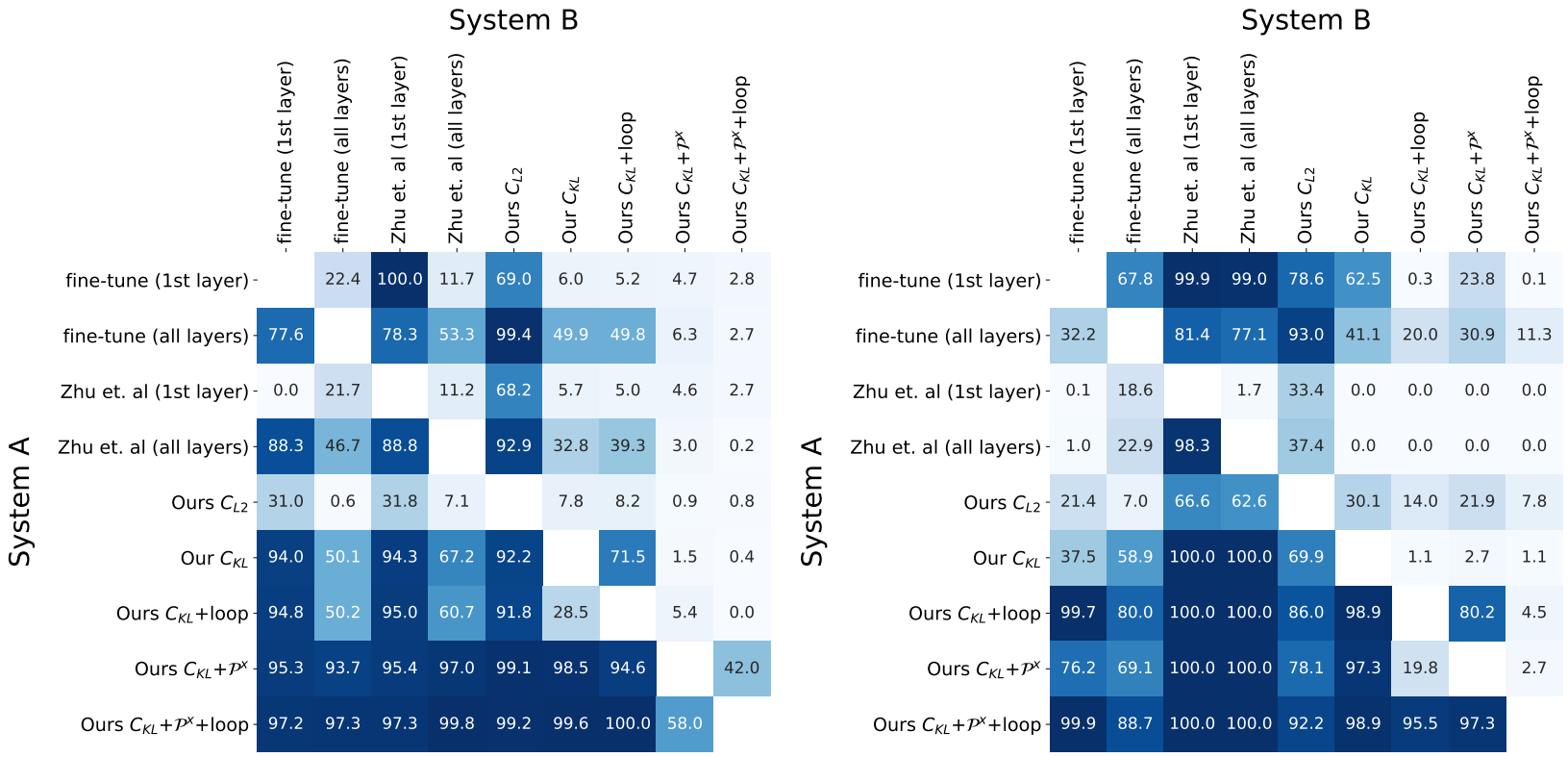



(a) Fact-checking.




(b) Question answering.

Figure 8.3: Probability distributions of weighted sum of metrics according to 1k random assignments sampled from a Dirichlet distribution (with $\alpha = 1$—see all values in table 8.1). Sampling weights allows to interpret the score in a probabilistic way. KnowledgeEditor (with different variants) presents distributions that are more skewed towards a high score (100) indicating that it is highly likely that when assigning some weights to the metrics, the weighted sum will be in favour of our method. *Better view with colors.*




(a) Fact-checking.

(b) Question answering.

Figure 8.4: Probability that *system A* is better than *system B* according to a weighted sum of metrics (see individual values in table 8.1) sampling mixing coefficients 1,000 times from a Dirichlet distribution (with $\alpha = 1$ to cover a diverse spectrum of metric combinations). The probability that KnowledgeEditor (with $\mathcal{C}_{KL} + \mathcal{P}^x$ + loop) is better than competing systems is high (>97% for FC and >88% for QA) indicating that it is highly likely that when assigning some weights to the metrics, the weighted sum will be in favour of our method. *Better view with colors.*

more effective as it prevents over-fitting (the best model updates the 1st layer in both FC and QA). However, in FC the updated model does not generalize on semantic equivalent inputs: the accuracy on paraphrases is much lower even than versions of our methods which do not use paraphrases in training (42% vs. >81%), and even more so when compared to those which use them (>94%).

Fine-tuning with Zhu et al. (2020) method does not affect performance for FC much, which is not surprising since standard fine-tuning already gets almost perfect scores. Differently, in the QA setting, using their constrained optimization boosts the retain accuracy (up to +4% to normal fine-tuning) but at the cost of a low success rate (≈80% where fine-tuning gets the perfect score).

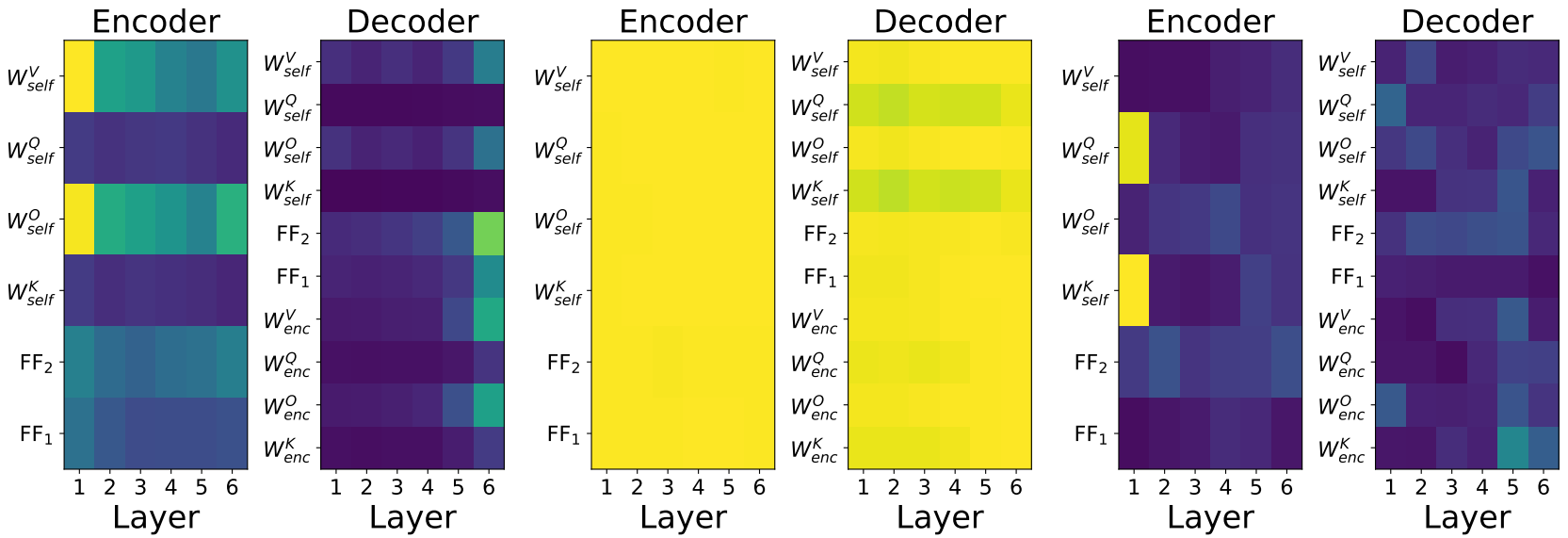
### 8.5.3 Accuracy on paraphrases

We evaluate our method both with and without the additional supervision of paraphrases to improve generalization—that corresponds to have $\mathcal{P}^x$ as the set of paraphrases of $x$ or $\mathcal{P}^x = \{x\}$ in equation 8.6, respectively. Without this additional supervision, KnowledgeEditor is already competitive in equivalence accuracy. However, employing this additional supervision is clearly beneficial on both tasks: we get the same success rate and re-train accuracy but equivalence accuracy improves by >70% on FC and >30% on QA, respectively (for generated paraphrases). In FC, although fine-tuning of a single layer proved to be optimal in terms of success rate and retain accuracy, it performs poorly for paraphrases. That is the model successfully updates the prediction of a particular datapoint, but does not update predictions of paraphrases. This indicates that fine-tuning to edit the knowledge of a model does not generalize well, and it *overfits* to specific inputs. On QA, also Zhu et al. (2020) performs poorly compared to our or other methods.

When other methods perform on par or better than ours on paraphrases, they do not have good retain accuracy (*e.g.,* see QA fine-tuning on table 8.1). Fine-tuning on QA seems to generalize better than on FC, but does not preserve previous knowledge. In table 8.1 we also report both the accuracy on the set of generated and human-generated paraphrases. Surprisingly, the scores on human-generated paraphrases are higher. We speculate that this happens because automatic paraphrases are sometimes not semantically equivalent or fluent.

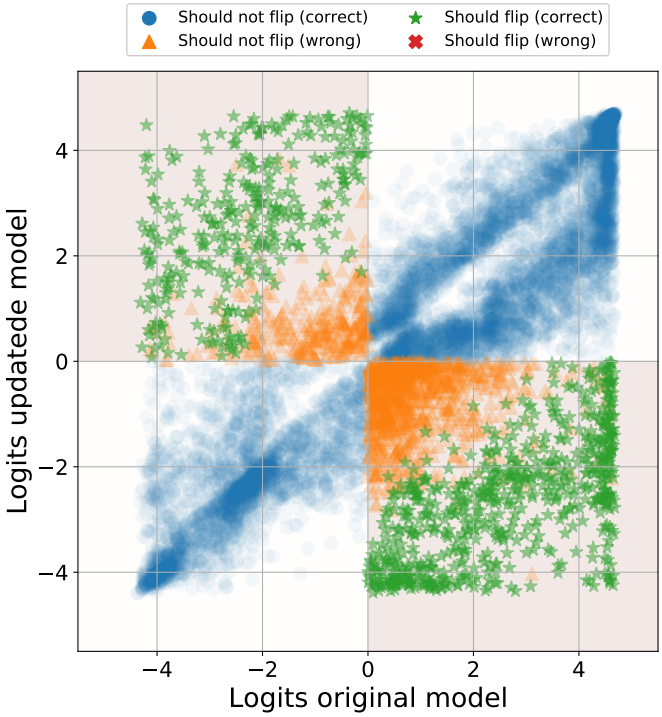
### 8.5.4 Analysis of model updates

In figure 8.6 we plot the distribution of logits of the original and updated model on FC for different methods. With an ideal method, all logits before and after an update have to stay the same (except the ones we want to change). From that figure, we can see distributions of different types of errors such as datapoints whose predictions were mistakenly flipped (from true to false or the other way around). These errors are mostly concentrated around the origin, where small

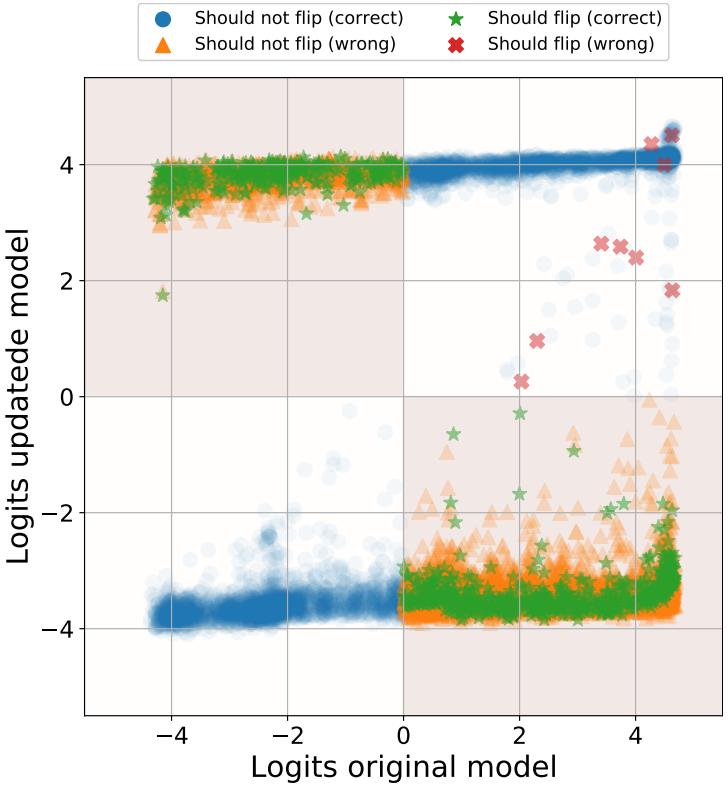



(a) Gradients. (b) Fine-tune (all layers). (c) KnowledgeEditor + $\mathcal{P}^x$.

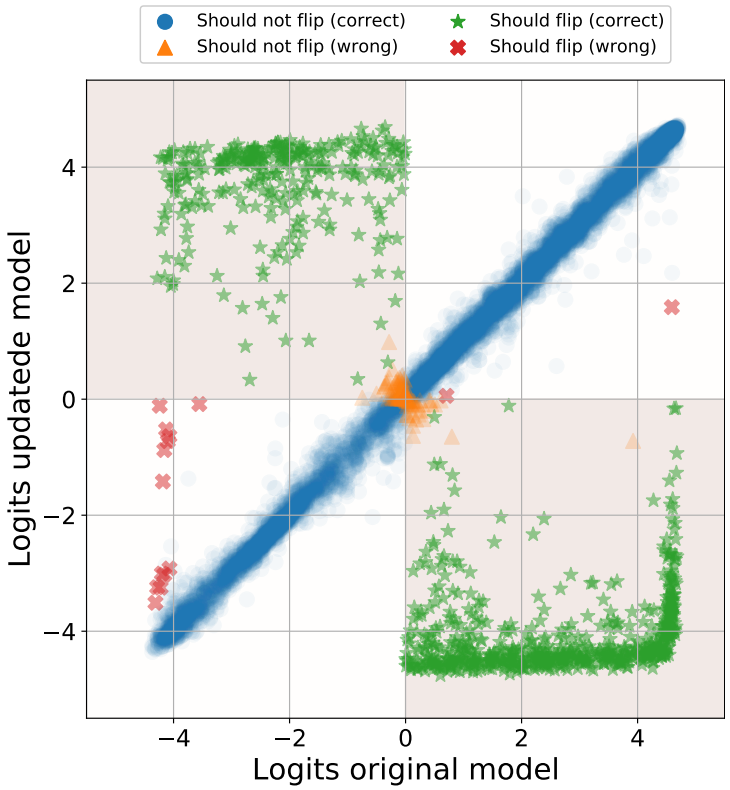
Figure 8.5: Average normalized magnitude of updates on weight matrices across layers for the QA experiment. Fine-tuning updates all layers uniformly while our updates are more sparse.




(a) Fine-tune (all layers).




(b) $\mathcal{C}_{L_2}$.




(c) Ours $\mathcal{C}_{KL}$ with $\mathcal{P}^x$.

Figure 8.6: Distribution of logits of the original model and updated model on FEVER. Fine-tuning all layers **(a)** leads to many errors, and the probability of the predictions does not stay the same even when they do not cross the decision boundary. $\mathcal{C}_{L_2}$ **(b)** successfully flips labels, but it does not force the predictions to stay the same. For our full method, $\mathcal{C}_{KL}$ with $\mathcal{P}^x$ **(c)**, errors are mainly concentrated around the origin where the model is uncertain, and small perturbations make logits to cross the decision boundary. *Better view with colors.*

perturbations make logits cross the decision boundary. When fine-tuning all layers, we can see a clear impact on logits, they undergo a lot of change (*i.e.,* points do not concentrate around the diagonal). Indeed, fine-tuning makes many datapoints cross the decision boundary and their probabilities to change from the original ones. The failure of $\mathcal{C}_{L_2}$ is visible in figure 8.6b as this method preserves almost none of the previous predictions. Instead KnowledgeEditor preserves almost all of the predicted labels as well as their probabilities (most datapoints in figure 8.6c stay on the diagonal).

We also report visualizations of the average weight updates for the QA experiment in figure 8.5. We report the setting with additional supervision from paraphrases (but the heatmaps are similar without them). There are three main observations from this plot. First, gradients are mostly concentrated on the first encoder layer and the last decoder layer. Gradients explain why the best subset of parameters to update is the first layer. Secondly, fine-tuning does not preserve gradient magnitudes and updates the whole model almost uniformly. That happens because of the optimizer's adaptive learning rate that initially erases the gradient direction. The gradient direction plays a role only after a couple of gradient steps, but most of the time, the method only needs one step to modify its knowledge. Lastly, our updates are sparser and are not consistent with the gradient for changing the predictions. That indicates that our method learns to use the gradient in a meaningful way (*i.e.,* ignoring some directions or manipulating its magnitude). It is surprising that the knowledge manipulation seems to be achieved by primarily modifying parameters affecting the shape of the attention distribution ($\boldsymbol{W}^{K}_{self}$ and $\boldsymbol{W}^{Q}_{self}$) rather than, *e.g.,* values ($\boldsymbol{W}^{V}_{self}$). As we discussed, the hyper-network may be regarded as a probe providing insights about the mechanism used by the model to encode the knowledge (Vig et al., 2020). For example, the focus on the bottom layer is already intriguing, as it contrasts with claims that memorization happens in top layers of image classification models (Stephenson et al., 2021), hinting at substantial differences in the underlying memorization mechanisms in NLP and vision. Proper investigation is however outside of the scope of this study.

**Gradient Analysis** During preliminary experiments, we studied a version of our hyper-network that did not exploit gradient information (see equation 8.10). Without gradient information, on FC the models converged $\approx 10$ times slower to reach the same accuracy and did not converge for QA (*i.e.,* the model was not able to get $> 75\%$ success rate and $> 50\%$ retain accuracy). That suggest that the gradients are helpful and actually used by our hyper-network but should not used directly, without a modification. To better show this, in table 8.2 we report correlations between different update methods and the gradient in terms of cosine similarities between updates. Naturally, fine-tuning and the gradient are highly correlated, but our method (with and without additional paraphrases

| | $\nabla_\theta \mathcal{L}$ | Fine-tune | $\mathcal{C}_{KL}$ | $\mathcal{C}_{KL} + \mathcal{P}^x$ |
|---|---|---|---|---|
| $\nabla_\theta \mathcal{L}$ | 1.000 | 0.451 | -0.018 | -0.025 |
| Fine-tune | 0.451 | 1.000 | -0.010 | -0.011 |
| $\mathcal{C}_{KL}$ | -0.017 | -0.010 | 1.000 | 0.183 |
| $\mathcal{C}_{KL} + \mathcal{P}^x$ | -0.021 | -0.011 | 0.183 | 1.000 |

Table 8.2: Average cosine similarities between different update methods and the gradient for the update as well. Fine-tuning is applied to all layers.

supervision), poorly correlates with the others. Low cosine similarity can be due to two factors i) the model indeed projects the gradient to a different and more 'knowledge preserving' direction, or ii) the parameter space is so large that cosine similarity gets to zero very quickly, not revealing the genuine underlying similarity.

## 8.6 Subsequent Work

The work in this chapter inspired recent research on updating language models. Model Editor Networks with Gradient Decomposition (MEND; Mitchell et al., 2022a) also proposes post-hoc editing via small auxiliary editing networks. Unlike our KnowledgeEditor, MEND only learns to transform the gradient obtained by standard fine-tuning for changing the predictions simplifying our formulation. To enable ease at scale, MEND uses a low-rank decomposition of the gradient to make the parameterization of their transformation tractable. The authors showed comparable if not superior performance to our formulation by applying their technique to larger models (*e.g.,* a T5-XXL with 11B parameters). Eventually, since gradients of updates can be computed for multiple inputs (*i.e.,* a batch), MEND can modify multiple facts simultaneously (as opposed to KnowledgeEditor where the architecture or method should be modified to achieve that). Although advantageous, multiple edits seem to work well only for small batches (up to 5) and further research is needed in this direction.

Semi-Parametric Editing with a Retrieval-Augmented Counterfactual Model (SERAC; Mitchell et al., 2022b) explored the opportunity to store edits in explicit external memory. SERAC then learns to reason over them to modulate the original model's predictions as needed. The authors show that methods like Editable Neural Networks (ENN; Sinitsin et al., 2020), KnowledgeEditor, and MEND completely fail for a batch of 75 edits while SERAC does not and achieves almost perfect accuracy.

Dhingra et al. (2022) studied time-aware language models proposing a new dataset (TempLAMA) with the objective of probing LMs for factual knowledge that changes over time. The authors also propose a simple technique for jointly

modeling text with temporal information. Concretely, they condition the prediction of an LM to time, *i.e.,* modeling $p(y|x, t; \theta)$ for a token $y$, context $x$ and time $t \in \mathbb{R}_{>0}$. This improves the memorization of seen facts from the training but allows post-training calibration on predictions about unseen facts from the future.

## 8.7 Conclusions

In this chapter, we explore the task of editing the factual knowledge implicitly stored in the parameters of Language Models. For this task, we formally define desiderata, the objective, and a set of metrics to measure the efficacy of different methods. We concretely evaluate that on two benchmarks based on closed-book fact-checking and question answering. We propose KnowledgeEditor, a method based on a hyper-network that *learns* to modify implicit knowledge stored within LM parameters efficiently and reliably. We provide comprehensive evaluations for our models against different variants of fine-tuning demonstrating the advantage of our approach. The magnitude of the updates predicted by our method may unfold the mechanisms used by the LMs to encode factual knowledge; we leave such investigation for future work.

Technology built upon pre-trained LMs inherits some or all of their potential harms (Bender et al., 2021). Our technology for editing the knowledge of LMs does not exacerbate their potential harms and can, in fact, be used to mitigate harms, as models can be corrected once problems are discovered. However, we note that malicious uses of our knowledge editor are possible. For example, malicious agents may use the techniques presented in this work to inject incorrect knowledge into LMs.

# Chapter 9

# Conclusions

In this thesis, we investigated the use of entities in various aspects of Natural Language understanding. In this chapter, we will briefly restate the main contributions of this thesis and discuss some limitations, future work, and what could have been done differently.

**Reasoning across Documents with Graph Convolutional Networks** With the intent to build an automatic question-answering system that reasons using several documents, we initially conducted experiments to have evidence that this class of models can be built (chapter 3). We designed a graph neural network that operates over a compact graph representation of a set of documents where nodes are mentions to entities and edges signal relations such as within and cross-document coreference. The model learns to answer questions by gathering evidence from different documents via a differentiable message-passing algorithm that updates node representations based on their neighborhood. We leveraged a pre-trained language model to compute initial node representations that we fixed to make the GNN model much faster to train and make inference with. We identify two main limitations of our study. The first is that we assumed that, given a query from a user, our system gets the gold documents to answer the query. The second is that we provide the list of all entities present in the evidence documents. These two assumptions were necessary to reduce complexity and focus on testing our model's reasoning ability. However, any realistic system must incorporate a retrieval and linking component to address the task fully. Additionally, we did not consider any storage requirement. Because the initial node representations are taken from a pre-trained language model, we need to store such vectors in storage. Even for the considered dataset (*i.e.,* WikiHop) the saved files occupied a considerable amount of storage (10-20 times the text

size). This might be inconvenient or even unfeasible for real-world systems. In our case, we traded computation for storage, making our system quite fast but putting no hard limit on the pre-processed data we save. Modern solutions for multi-document deading (for question answering, summarization, or other tasks) rely on efficient transformers instead of graph representation. They overcome the memory limitation of input documents with sub-quadratic attention mechanisms which allow to input the concatenation of several thousand-word documents (Tay et al., 2022a). With these solutions in mind, we hypothesize that the future of multi-document reasoning techniques will mostly rely on these efficient transformers. However, the use of entities to select which documents to read might still be part of these systems.

**Entity Retrieval with Language Models** In the subsequent chapters (chapters 4, 5, and 6), we then investigated how to leverage pre-trained language models to retrieve entities from text to meet the requirements set by the question answering system developed before. Although we were motivated by our previous research, the development of entity detection and linking systems is useful across a large number of NLU tasks, and thus the utility of such progress goes way beyond adding up to our previous multi-document reasoning solutions. We defined a class of models that detect and classify entities using an autoregressive formulation. This class of models generates the target label name left-to-right token-by-token with constrained decoding. Additionally, they significantly reduce the memory footprint of current systems (up to 15 times). Our formulation scales linearly in parameter space with respect to only vocabulary size which is generally fixed rather than entity size which can be in the order of millions for large Knowledge Bases. Subsequently, we successfully extended this approach, initially tested in English, to more than 100 languages. We identified two main limitations of our approach: (i) it is not fast as beam search is used at inference time to do classification, and (ii) it cannot learn end-to-end the representation of the labels since they are fixed as their names. We addressed the first limitation in chapter 6 modifying the transformed encoder-decoder architecture into a transformer encoder and LSTM decoder. The RNN-based decoder substantially speeds-up inference time (up to 70 times) making the approach much more suitable for real-world applications. However, our study was limited to both training on a relatively small infrastructure and testing on small datasets. Conversely, the end-to-end learning of label representations is still an open problem. Recent works (DSI and SEAL; Tay et al., 2022b; Bevilacqua et al., 2022) proposed a different solution for indexing paragraphs (*i.e.,* labels for document retrieval). Differential Search Index (DSI) uses unsupervised learning to first assign hierarchical unique labels to documents while Search Engines with Autoregressive LMs (SEAL) directly employs constrained beam search to get paragraphs. We think that it is worth investigating alternative solutions to learning both the label identifiers

and the retriever at the same time. For now, such methods may be inefficiently implemented with reinforcement learning but we advocate for investigating ad-hoc alternatives.

**Post-hoc Interpretability** In chapter 7, we have introduced a new *post hoc* interpretation method which learns to completely remove subsets of inputs or hidden states through masking. We circumvented an intractable search by learning an end-to-end differentiable prediction model. To overcome the hindsight bias problem, we probed the model's hidden states at different depths and amortize predictions over the training set. Faithfulness was validated in a controlled experiment pointing more clearly to some flaws of other attribution methods. In practice, back-propagating through the expected $L_0$ loss is quite challenging since the training is unstable. The instability comes for two main reasons: (i) we are optimizing a constrained objective so the optimal values for the parameter always stands at a saddle point which is not ideal for SGD, and (ii) the Hard Concrete distribution has zero gradient when sampling exact zeros but non-zero $L_0$ loss which creates unbalance feedback for how much sampling zeros helps the overall objective. In addition to the training problems, the faithfulness of our method was only measured with a simple experiment while left undiscussed for a real-world task. We did not discuss that simply because it is not possible to fully know what a deep neural model does which makes it impossible to fully demonstrate the faithfulness of any post-hoc interpretability method. Certainly we can approximate faithfulness evaluation (*e.g.,* Bastings et al., 2022) but how accurate or safe that it is still unclear. While the ability to analyze models with post-hoc interpretability methods seems desirable, the impracticality of evaluating the faithfulness of these methods imposes limitations on that research direction. Although opinionated, we want to advise future research in interpretability to better consider the development of *pre-hoc* systems rather than *post-hoc*. We think building models that are inherently interpretable will produce much more robust systems rather than attempting to provide attribution to black-box models.

**Model Editing** In chapter 8, we proposed a method that can be used to edit this factual knowledge about entities and, thus, fix 'bugs' or unexpected predictions without the need for expensive re-training or fine-tuning. We trained a hyper-network with constrained optimization to modify a fact without affecting the rest of the knowledge; the trained hyper-network is then used to predict the weight update at test time. We showed the efficacy with two popular architectures and knowledge-intensive tasks: (i) a BERT model fine-tuned for fact-checking, and (ii) a sequence-to-sequence BART model for question answering. With our method, changing a prediction on the specific wording of a query tends to result in a consistent change in predictions also for its paraphrases. We showed that this can be further encouraged by exploiting (*e.g.,* automatically generated) paraphrases

during training. One of the main limitations of this study was that we tested a fact-single model update. However, real-world applications definitely require a large number of following updates without retraining. Subsequent studies addressed that problem but not in a definitive way (Mitchell et al., 2022a; Mitchell et al., 2022b). We encourage investigation in that direction. Similar to what is expressed above, another consideration for future work is that we think building models that are inherently editable should be much more efficient than trying to edit arbitrary models *a posteriori*. For instance, language models with external and controllable memory would allow easy editing with much less burden.

# Appendix A

# Appendix

# A.1 GENRE examples

```
ID: '87d95287-707e-4bd9-9633-ca0c611a4a3a_World_Without_Superma:8'
inputs: '[..] When Superman leaves Earth for New Krypton , he appoints , newly freed from the Phantom Zone , to take his place as guardian of [START_ENT] Metropolis [END_ENT] . Mon-El assumes the secret identity of Johnathan Kent as a tribute to Clark \'s adoptive father , posing as Clark \'s cousin . [..]'
gold_output: 'Metropolis (comics)'
predicted_outputs: [
    ('Metropolis_(comics)', -0.09),
    ('Themyscira_(DC_Comics)', -1.09),
    ('Metropolis_(disambiguation)', -1.27),
    ('Superman_(comic_book)', -1.51),
    ('Superman_(Earth-Two)', -1.52)
]
```

Figure A.1: Example of a GENRE prediction for named entity disambiguation on KILT WNED. The input is plain text where a mention is flagged with two special start and end tokens `[START_ENT]` and `[END_ENT]`. The output is a ranked list of entity (where we report the log-likelihood as well).

```
ID: 'sfq_18245'
inputs: "Which Florentine painter 1535-1607 used the name Bronzino after the death of his 'uncle'?"
gold_output: 'Bronzino'
predicted_outputs: [
    ('Florence', -0.37),
    ('Bronzino', -0.62),
    ('Niccolo_Machiavelli', -0.64),
    ('Giorgio_de_Chirico', -0.71),
    ('Vitruvian_Man', -0.73)
]
```

(a) TriviaQA (open domain question answering).

```
ID: '4713'
inputs: 'Tool has won three Oscars.'
gold_output: 'Tool (band)'
predicted_outputs: [
    ('Tool_(band)', -0.08),
    ('Tool_(disambiguation)', -1.59),
    ('Machine_Head_(band)', -1.73),
    ('Language_Arts_(album)', -1.97),
    ('Machine_Gun_(band)', -2.12)
]
```

(b) FEVER (fact checking).

Figure A.2: Example of GENRE predictions for the document retrieval task on KILT. The input is a query and the output is a ranked list of Wikipedia article titles (we also report the log-likelihood of the solutions).

```
ID: '1106testa_SOCCER'
inputs: 'SOCCER - RESULT IN SPANISH FIRST DIVISION. MADRID 1996-08-31 Result of game played in the Spanish first division on Saturday: Deportivo Coruna 1 Real Madrid 1.'
gold_output: 'SOCCER - RESULT IN [SPANISH](Spain) FIRST DIVISION . [MADRID](Madrid) 1996-08-31 Result of game played in the [Spanish](Spain) first division on Saturday : Deportivo Coruna 1 [Real Madrid](Real Madrid C.F.) 1.'
predicted_output: 'SOCCER - RESULT IN [SPANISH](Spain) FIRST DIVISION . [MADRID](Madrid) 1996-08-31 Result of game played in the [Spanish](Spain) first division on Saturday : [Deportivo](Deportivo de La Coruna) Coruna 1 [Real Madrid](Real Madrid C.F.) 1.'
gold_spans: [
    [19, 7, 'Spain'],
    [44, 6, 'Madrid'],
    [91, 7, 'Spain'],
    [147, 11, 'Real_Madrid_C.F.']
]
predicted_spans: [
    [19, 7, 'Spain'],
    [44, 6, 'Madrid'],
    [91, 7, 'Spain'],
    [128, 9, 'Deportivo_de_La_Coruna'],
    [147, 11, 'Real_Madrid_C.F.']
]

Micro-precision:    0.80
Micro-recall:       1.00
Micro-F1:           0.88
```

Figure A.3: Example of a GENRE prediction for end-to-end entity linking on AIDA. The input is plain text and the output is a *Markup* string where the links are WIKIPEDIA titles. Spans are in the format $\langle s_i, l_i, t_i \rangle$: *start of the mention*, *length of the mention*, and *title* respectively.

| **Language** | **Pages** | **Names** | **Hyperlinks** |
|---|---|---|---|
| Afrikaans (af) | 85,456 | 110,705 | 1,089,581 |
| Albanian (sq) | 86,234 | 112,112 | 978,394 |
| Amharic (am) | 15,280 | 19,905 | 75,575 |
| Arabic (ar) | 971,861 | 1,883,080 | 10,308,074 |
| Armenian (hy) | 260,395 | 582,941 | 3,082,000 |
| Assamese (as) | 6,119 | 19,041 | 61,209 |
| Azerbaijani (az) | 152,033 | 189,793 | 1,562,968 |
| Bambara (bm) | 747 | 916 | 2,191 |
| Basque (eu) | 337,916 | 430,456 | 4,305,648 |
| Belarusian (be) | 181,030 | 415,152 | 2,459,794 |
| Bengali (bn) | 76,121 | 257,730 | 960,484 |
| Bosnian (bs) | 82,164 | 184,148 | 1,916,515 |
| Breton (br) | 67,388 | 88,284 | 1,255,295 |
| Bulgarian (bg) | 257,962 | 376,934 | 4,655,641 |
| Burmese (my) | 48,683 | 55,700 | 98,992 |
| Catalan (ca) | 630,340 | 1,024,519 | 14,790,419 |
| Chinese (zh) | 1,085,180 | 1,951,612 | 17,262,417 |
| Croatian (hr) | 193,705 | 250,008 | 4,223,179 |
| Czech (cs) | 439,249 | 719,643 | 12,173,376 |
| Danish (da) | 255,957 | 405,745 | 5,621,483 |
| Dutch (nl) | 1,986,801 | 2,714,649 | 25,002,389 |
| English (en) | 6,071,492 | 14,751,661 | 134,477,329 |
| Esperanto (eo) | 270,871 | 447,159 | 5,570,306 |
| Estonian (et) | 201,505 | 342,215 | 4,700,888 |
| Finnish (fi) | 470,896 | 737,165 | 8,390,037 |
| French (fr) | 2,160,840 | 3,718,185 | 59,006,932 |
| Frysk (fy) | 42,893 | 72,490 | 1,206,432 |
| Fulah (ff) | 306 | 421 | 912 |
| Gaelic, (gd) | 15,126 | 23,631 | 180,186 |
| Galician (gl) | 159,849 | 229,561 | 4,709,070 |
| Ganda (lg) | 2,376 | 2,668 | 2,476 |
| Georgian (ka) | 135,040 | 138,267 | 1,369,094 |
| German (de) | 2,356,465 | 3,877,850 | 60,638,345 |
| Greek (el) | 170,541 | 251,692 | 3,310,875 |
| Guarani (gn) | 3,755 | 5,589 | 89,593 |
| Gujarati (gu) | 29,091 | 32,526 | 402,483 |
| Haitian (ht) | 59,350 | 63,279 | 677,064 |
| Hausa (ha) | 4,143 | 5,025 | 19,929 |
| Hebrew (he) | 253,861 | 444,127 | 9,947,354 |
| Hindi (hi) | 138,378 | 192,652 | 1,040,288 |
| Hungarian (hu) | 459,261 | 663,995 | 10,138,904 |

| **Language** | **Pages** | **Names** | **Hyperlinks** |
|---|---|---|---|
| Icelandic (is) | 48,563 | 75,963 | 772,213 |
| Igbo (ig) | 1,521 | 3,000 | 4,702 |
| Indonesian (id) | 516,196 | 1,015,784 | 7,882,254 |
| Irish (ga) | 51,824 | 61,336 | 435,135 |
| Italian (it) | 1,571,189 | 2,450,009 | 39,382,886 |
| Japanese (ja) | 1,173,978 | 1,877,660 | 45,957,053 |
| Javanese (jv) | 57,422 | 75,792 | 718,589 |
| Kannada (kn) | 25,986 | 33,880 | 227,731 |
| Kazakh (kk) | 229,165 | 271,260 | 1,564,344 |
| Khmer (km) | 9,838 | 12,349 | 73,950 |
| Kongo (kg) | 1,247 | 1,440 | 3,733 |
| Korean (ko) | 475,605 | 1,061,961 | 8,309,492 |
| Kurdish (ku) | 26,963 | 42,134 | 244,779 |
| Kyrgyz (ky) | 80,985 | 89,486 | 271,335 |
| Lao (lo) | 4,414 | 5,761 | 16,173 |
| Latin (la) | 132,410 | 186,829 | 1,986,307 |
| Latvian (lv) | 99,062 | 226,570 | 1,522,814 |
| Lingala (ln) | 3,262 | 4,134 | 15,518 |
| Lithuanian (lt) | 197,215 | 282,077 | 3,512,764 |
| Macedonian (mk) | 103,960 | 152,384 | 2,035,348 |
| Malagasy (mg) | 92,500 | 142,156 | 857,000 |
| Malay (ms) | 331,403 | 388,110 | 3,190,700 |
| Malayalam (ml) | 67,475 | 152,809 | 712,869 |
| Marathi (mr) | 55,601 | 100,904 | 355,536 |
| Mongolian (mn) | 21,772 | 28,455 | 208,847 |
| Nepali (ne) | 34,107 | 39,904 | 151,958 |
| Norwegian (no) | 521,665 | 816,772 | 10,234,086 |
| Oriya (or) | 15,532 | 30,431 | 79,261 |
| Oromo (om) | 1,063 | 1,317 | 7,153 |
| Panjabi (pa) | 33,934 | 46,720 | 145,204 |
| Pashto (ps) | 11,773 | 16,878 | 46,987 |
| Persian (fa) | 716,604 | 2,139,255 | 5,567,774 |
| Polish (pl) | 1,370,672 | 1,812,412 | 25,817,929 |
| Portuguese (pt) | 1,053,673 | 1,858,821 | 20,625,904 |
| Quechua (qu) | 21,670 | 41,230 | 247,508 |
| Romanian (ro) | 403,517 | 979,524 | 6,974,837 |
| Russian (ru) | 1,585,051 | 3,592,042 | 35,783,391 |
| Sanskrit (sa) | 11,960 | 22,472 | 73,380 |
| Serbian (sr) | 625,871 | 3,248,789 | 7,012,202 |
| Sindhi (sd) | 14,616 | 18,556 | 33,990 |
| Sinhala (si) | 20,363 | 29,794 | 90,866 |

| **Language** | **Pages** | **Names** | **Hyperlinks** |
|---|---|---|---|
| Slovak (sk) | 232,109 | 301,681 | 4,014,344 |
| Slovenian (sl) | 166,997 | 238,706 | 3,754,135 |
| Somali (so) | 6,716 | 9,595 | 53,132 |
| Spanish (es) | 1,547,372 | 3,313,727 | 37,749,593 |
| Sundanese (su) | 54,921 | 61,716 | 598,878 |
| Swahili (sw) | 53,926 | 74,634 | 693,049 |
| Swati (ss) | 514 | 610 | 4,344 |
| Swedish (sv) | 3,755,203 | 6,143,945 | 39,409,278 |
| Tagalog (tl) | 79,036 | 181,951 | 562,526 |
| Tamil (ta) | 129,591 | 168,718 | 1,110,037 |
| Telugu (te) | 71,819 | 98,189 | 841,549 |
| Thai (th) | 139,522 | 299,433 | 2,190,249 |
| Tigrinya (ti) | 307 | 390 | 696 |
| Tswana (tn) | 827 | 894 | 4,896 |
| Turkish (tr) | 338,865 | 593,365 | 5,657,757 |
| Ukrainian (uk) | 939,234 | 1,468,963 | 16,360,016 |
| Urdu (ur) | 156,300 | 353,391 | 1,142,953 |
| Uzbek (uz) | 132,666 | 450,865 | 764,566 |
| Vietnamese (vi) | 1,240,324 | 1,466,573 | 10,015,209 |
| Welsh (cy) | 106,556 | 154,043 | 1,254,901 |
| Wolof (wo) | 1,503 | 1,969 | 7,257 |
| Xhosa (xh) | 1,370 | 1,610 | 14,163 |
| Yoruba (yo) | 32,304 | 42,022 | 88,032 |
| Others* | 12,613,082 | 12,613,082 | - |
| **Total** | 53,849,351 | 89,270,463 | 777,210,183 |

Table A.1: Number of pages, entity names, and hyperlinks used in the 105 languages used for mGENRE. Entity names are more than the pages because we also includes redirections. Hyperlinks count is after filtering missed alignments to WIKIDATA and augmenting when there is no name in the source language. *These are names defined but never used in any hyperlink.

| **Language** | **Alias Table** | **mGENRE** | **Support** |
|---|---|---|---|
| Afrikaans (af) | 92.1 | 97.0 | 1,089,581 |
| Albanian (sq) | 87.4 | 94.1 | 978,394 |
| Amharic (am) | 85.6 | 94.2 | 75,575 |
| Arabic (ar) | 89.9 | 94.2 | 10,308,074 |
| Armenian (hy) | 89.2 | 94.3 | 3,082,000 |
| Assamese (as) | 85.8 | 94.6 | 61,209 |
| Azerbaijani (az) | 90.0 | 96.0 | 1,562,968 |
| Bambara (bm) | 80.6 | 89.8 | 2,191 |
| Basque (eu) | 94.0 | 97.6 | 4,305,648 |
| Belarusian (be) | 85.9 | 94.8 | 2,459,794 |
| Bengali (bn) | 79.7 | 90.4 | 960,484 |
| Bosnian (bs) | 86.5 | 93.7 | 1,916,515 |
| Breton (br) | 91.9 | 96.8 | 1,255,295 |
| Bulgarian (bg) | 88.8 | 95.4 | 4,655,641 |
| Burmese (my) | 86.4 | 93.7 | 98,992 |
| Catalan (ca) | 91.5 | 96.3 | 14,790,419 |
| Chinese (zh) | 88.0 | 93.5 | 17,262,417 |
| Croatian (hr) | 84.7 | 93.9 | 4,223,179 |
| Czech (cs) | 87.1 | 94.0 | 12,173,376 |
| Danish (da) | 90.6 | 95.5 | 5,621,483 |
| Dutch (nl) | 86.4 | 95.2 | 25,002,389 |
| English (en) | 84.6 | 92.6 | 134,477,329 |
| Esperanto (eo) | 89.7 | 95.5 | 5,570,306 |
| Estonian (et) | 91.2 | 97.7 | 4,700,888 |
| Finnish (fi) | 87.4 | 94.3 | 8,390,037 |
| French (fr) | 83.6 | 92.6 | 59,006,932 |
| Frysk (fy) | 91.3 | 95.2 | 1,206,432 |
| Fulah (ff) | 44.3 | 69.7 | 912 |
| Gaelic, (gd) | 90.9 | 94.6 | 180,186 |
| Galician (gl) | 90.9 | 95.9 | 4,709,070 |
| Ganda (lg) | 74.7 | 88.2 | 2,476 |
| Georgian (ka) | 87.9 | 95.7 | 1,369,094 |
| German (de) | 86.9 | 94.7 | 60,638,345 |
| Greek (el) | 84.1 | 91.9 | 3,310,875 |
| Guarani (gn) | 92.7 | 96.7 | 89,593 |
| Gujarati (gu) | 96.6 | 98.1 | 402,483 |
| Haitian (ht) | 95.6 | 90.7 | 677,064 |
| Hausa (ha) | 81.7 | 89.9 | 19,929 |
| Hebrew (he) | 90.6 | 93.5 | 9,947,354 |
| Hindi (hi) | 89.1 | 94.3 | 1,040,288 |
| Hungarian (hu) | 90.7 | 95.1 | 10,138,904 |

| | | | |
|---|---|---|---|
| Icelandic (is) | 89.8 | 94.9 | 772,213 |
| Igbo (ig) | 87.4 | 89.7 | 4,702 |
| Indonesian (id) | 91.8 | 94.8 | 7,882,254 |
| Irish (ga) | 90.1 | 96.3 | 435,135 |
| Italian (it) | 89.9 | 94.2 | 39,382,886 |
| Japanese (ja) | 92.1 | 96.2 | 45,957,053 |
| Javanese (jv) | 92.4 | 96.7 | 718,589 |
| Kannada (kn) | 87.8 | 93.5 | 227,731 |
| Kazakh (kk) | 91.0 | 97.3 | 1,564,344 |
| Khmer (km) | 85.1 | 90.5 | 73,950 |
| Kongo (kg) | 81.1 | 92.6 | 3,733 |
| Korean (ko) | 89.1 | 93.7 | 8,309,492 |
| Kurdish (ku) | 89.1 | 95.4 | 244,779 |
| Kyrgyz (ky) | 86.1 | 96.6 | 271,335 |
| Lao (lo) | 86.8 | 91.5 | 16,173 |
| Latin (la) | 88.0 | 95.8 | 1,986,307 |
| Latvian (lv) | 85.3 | 95.0 | 1,522,814 |
| Lingala (ln) | 84.4 | 93.7 | 15,518 |
| Lithuanian (lt) | 89.5 | 95.5 | 3,512,764 |
| Macedonian (mk) | 88.4 | 96.1 | 2,035,348 |
| Malagasy (mg) | 94.4 | 98.8 | 857,000 |
| Malay (ms) | 89.0 | 95.2 | 3,190,700 |
| Malayalam (ml) | 75.8 | 90.0 | 712,869 |
| Marathi (mr) | 85.0 | 94.2 | 355,536 |
| Mongolian (mn) | 86.1 | 93.5 | 208,847 |
| Nepali (ne) | 83.9 | 92.8 | 151,958 |
| Norwegian (no) | 88.7 | 95.4 | 10,234,086 |
| Oriya (or) | 87.3 | 93.2 | 79,261 |
| Oromo (om) | 80.8 | 91.4 | 7,153 |
| Panjabi (pa) | 85.4 | 92.1 | 145,204 |
| Pashto (ps) | 80.0 | 92.1 | 46,987 |
| Persian (fa) | 87.6 | 95.2 | 5,567,774 |
| Polish (pl) | 82.3 | 92.4 | 25,817,929 |
| Portuguese (pt) | 85.9 | 93.0 | 20,625,904 |
| Quechua (qu) | 93.8 | 96.2 | 247,508 |
| Romanian (ro) | 88.8 | 95.2 | 6,974,837 |
| Russian (ru) | 85.7 | 94.5 | 35,783,391 |
| Sanskrit (sa) | 86.2 | 95.8 | 73,380 |
| Serbian (sr) | 85.7 | 95.5 | 7,012,202 |
| Sindhi (sd) | 80.7 | 92.1 | 33,990 |
| Sinhala (si) | 80.1 | 89.9 | 90,866 |
| Slovak (sk) | 88.0 | 96.3 | 4,014,344 |

| | | | |
|---|---|---|---|
| Slovenian (sl) | 87.0 | 95.4 | 3,754,135 |
| Somali (so) | 84.5 | 92.2 | 53,132 |
| Spanish (es) | 86.5 | 92.3 | 37,749,593 |
| Sundanese (su) | 93.6 | 96.8 | 598,878 |
| Swahili (sw) | 91.9 | 97.2 | 693,049 |
| Swati (ss) | 81.2 | 91.2 | 4,344 |
| Swedish (sv) | 90.9 | 97.5 | 39,409,278 |
| Tagalog (tl) | 83.4 | 89.9 | 562,526 |
| Tamil (ta) | 84.2 | 93.8 | 1,110,037 |
| Telugu (te) | 89.2 | 95.4 | 841,549 |
| Thai (th) | 92.3 | 95.2 | 2,190,249 |
| Tigrinya (ti) | 57.8 | 79.0 | 696 |
| Tswana (tn) | 89.5 | 90.6 | 4,896 |
| Turkish (tr) | 89.0 | 93.9 | 5,657,757 |
| Ukrainian (uk) | 85.8 | 94.3 | 16,360,016 |
| Urdu (ur) | 91.0 | 96.3 | 1,142,953 |
| Uzbek (uz) | 73.8 | 98.4 | 764,566 |
| Vietnamese (vi) | 91.8 | 95.8 | 10,015,209 |
| Welsh (cy) | 94.4 | 96.4 | 1,254,901 |
| Wolof (wo) | 78.0 | 91.9 | 7,257 |
| Xhosa (xh) | 73.9 | 92.6 | 14,163 |
| Yoruba (yo) | 75.6 | 87.9 | 88,032 |
| **micro-avg** | 86.5 | 93.8 | - |
| **macro-avg** | 86.6 | 93.9 | - |
| **total** | - | - | 777,210,183 |

Table A.2: Accuracy of mGENRE and alias table on the 105 languages in our Wikipedia validation set. The support indicates how many datapoints where used to train where validation is done on 1,000 examples per language (less for Tigrinya and Fulah since we have less than a thousand hyperlinks).

# Bibliography


Adi Yossi, Kermany Einat, Belinkov Yonatan, Lavi Ofer, & Goldberg Yoav. (2017). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on p. 109).

Aghajanyan Armen, Huang Bernie, Ross Candace, Karpukhin Vladimir, Xu Hu, Goyal Naman, Okhonko Dmytro, Joshi Mandar, Ghosh Gargi, Lewis Mike, & Zettlemoyer Luke. (2022). CM3: A causal masked multimodal model of the internet. *CoRR*, *abs/2201.07520*arXiv (cit. on pp. 60, 67).

Akbik Alan, Bergmann Tanja, Blythe Duncan, Rasul Kashif, Schweter Stefan, & Vollgraf Roland. (2019). FLAIR: An easy-to-use framework for state-of-the-art NLP, In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics (demonstrations)*, Minneapolis, Minnesota, Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-4010. (Cit. on p. 59)

Alemi Alexander A., Fischer Ian, Dillon Joshua V., & Murphy Kevin. (2017). Deep variational information bottleneck, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on p. 137).

Anderson Peter, Fernando Basura, Johnson Mark, & Gould Stephen. (2017). Guided open vocabulary image captioning with constrained beam search, In *Proceedings of the 2017 conference on empirical methods in natural language processing*, Copenhagen, Denmark, Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1098. (Cit. on p. 51)

Antoniou Grigoris, & Harmelen Frank van. (2004). Web ontology language: Owl, In *Handbook on ontologies*. Springer. (Cit. on p. 7).

Asai Akari, Hashimoto Kazuma, Hajishirzi Hannaneh, Socher Richard, & Xiong Caiming. (2020). Learning to retrieve reasoning paths over wikipedia graph for question answering, In *8th international conference on learning repre-*

*sentations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*, OpenReview.net. (Cit. on pp. 3, 22, 41, 42).

Attardi Giusepppe. (2015). Wikiextractor. GitHub. (Cit. on p. 78).

Ba Jimmy Lei, Kiros Jamie Ryan, & Hinton Geoffrey E. (2016). Layer normalization. *Advances in NIPS 2016 Deep Learning Symposium* (cit. on pp. 15, 101).

Bach Sebastian, Binder Alexander, Montavon Grégoire, Klauschen Frederick, Müller Klaus-Robert, & Samek Wojciech. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, *10*(7) (cit. on p. 106).

Baehrens David, Schroeter Timon, Harmeling Stefan, Kawanabe Motoaki, Hansen Katja, & MÃžller Klaus-Robert. (2010). How to explain individual classification decisions. *Journal of Machine Learning Research*, *11*(Jun), 1803–1831 (cit. on p. 106).

Bahdanau Dzmitry, Cho Kyunghyun, & Bengio Yoshua. (2015). Neural machine translation by jointly learning to align and translate (Yoshua Bengio & Yann LeCun, Eds.). In Yoshua Bengio & Yann LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. (Cit. on p. 13).

Balakrishnan Anusha, Rao Jinfeng, Upasani Kartikeya, White Michael, & Subba Rajen. (2019). Constrained decoding for neural NLG from compositional representations in task-oriented dialogue, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1080. (Cit. on pp. 50, 51)

Barba Edoardo, Procopio Luigi, & Navigli Roberto. (2022). ExtEnD: Extractive entity disambiguation, In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Dublin, Ireland, Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.177. (Cit. on pp. 60, 68)

Bastings Jasmijn, Titov Ivan, Aziz Wilker, Marcheggiani Diego, & Sima'an Khalil. (2017). Graph convolutional encoders for syntax-aware neural machine translation, In *Proceedings of the 2017 conference on empirical methods in natural language processing*, Copenhagen, Denmark, Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1209. (Cit. on p. 25)

Bastings Jasmijn, Aziz Wilker, & Titov Ivan. (2019). Interpretable neural predictions with differentiable binary variables, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1284. (Cit. on pp. 112, 114, 118)

Bastings Jasmijn, Ebert Sebastian, Zablotskaia Polina, Sandholm Anders, & Filippova Katja. (2022). "will you find these shortcuts?" a protocol for evaluating the faithfulness of input salience methods for text classification, In *Proceedings of the 2022 conference on empirical methods in natural language processing*, Abu Dhabi, United Arab Emirates, Association for Computational Linguistics. (Cit. on p. 167).

Battaglia Peter W., Hamrick Jessica B., Bapst Victor, Sanchez-Gonzalez Alvaro, Zambaldi Vinícius Flores, Malinowski Mateusz, Tacchetti Andrea, Raposo David, Santoro Adam, Faulkner Ryan, Gülçehre Çaglar, Song H. Francis, Ballard Andrew J., Gilmer Justin, Dahl George E., Vaswani Ashish, Allen Kelsey R., Nash Charles, Langston Victoria, ... Pascanu Razvan. (2018). Relational inductive biases, deep learning, and graph networks. *CoRR*, *abs/1806.01261*arXiv (cit. on pp. 11, 12).

Bauer Lisa, Wang Yicheng, & Bansal Mohit. (2018). Commonsense for generative multi-hop question answering tasks, In *Proceedings of the 2018 conference on empirical methods in natural language processing*, Brussels, Belgium, Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1454. (Cit. on pp. 22, 25, 32, 33)

Bechhofer Sean, Van Harmelen Frank, Hendler Jim, Horrocks Ian, McGuinness Deborah L, Patel-Schneider Peter F, Stein Lynn Andrea, Et al. (2004). Owl web ontology language reference. *W3C recommendation*, *10*(2), 1–53 (cit. on p. 7).

Beitzel Steven M., Jensen Eric C., & Frieder Ophir. (2009). Average r-precision. In *Encyclopedia of database systems* (pp. 195–195). Boston, MA, Springer US. https://doi.org/10.1007/978-0-387-39940-9_491. (Cit. on p. 57)

Belinkov Yonatan, & Glass James. (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, *7*, 49–72. https://doi.org/10.1162/tacl_a_00254 (cit. on pp. 106, 109, 140)

Beltagy Iz, Peters Matthew E., & Cohan Arman. (2020). Longformer: The long-document transformer. *CoRR*, *abs/2004.05150*arXiv (cit. on pp. 33, 42, 96, 99, 100).

Bender Emily M., Gebru Timnit, McMillan-Major Angelina, & Shmitchell Shmargaret. (2021). On the dangers of stochastic parrots: Can language models be too big?, In *Proceedings of the 2021 acm conference on fairness, accountability, and transparency*, Virtual Event, Canada, Association for Computing Machinery. https://doi.org/10.1145/3442188.3445922. (Cit. on p. 163)

Bevilacqua Michele, Ottaviano Giuseppe, Lewis Patrick, Yih Wen-tau, Riedel Sebastian, & Petroni Fabio. (2022). Autoregressive search engines: Generating substrings as document identifiers. *CoRR*, *abs/2204.10628*arXiv. https://doi.org/10.48550/arXiv.2204.10628 (cit. on pp. 61, 67, 166)

Bollacker Kurt, Evans Colin, Paritosh Praveen, Sturge Tim, & Taylor Jamie. (2008). Freebase: A collaboratively created graph database for structuring human knowledge, In *Proceedings of the 2008 acm sigmod international conference on management of data*. (Cit. on p. 80).

Bommasani Rishi, Hudson Drew A., Adeli Ehsan, Altman Russ, Arora Simran, von Arx Sydney, Bernstein Michael S., Bohg Jeannette, Bosselut Antoine, Brunskill Emma, Brynjolfsson Erik, Buch Shyamal, Card Dallas, Castellon Rodrigo, Chatterji Niladri S., Chen Annie S., Creel Kathleen, Davis Jared Quincy, Demszky Dorottya, ... et al. (2021). On the opportunities and risks of foundation models. *CoRR*, *abs/2108.07258*arXiv (cit. on p. 19).

Bordes Antoine, Boureau Y-Lan, & Weston Jason. (2017). Learning end-to-end goal-oriented dialog, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on p. 3).

Botha Jan A., Shan Zifei, & Gillick Daniel. (2020). Entity Linking in 100 Languages, In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.630. (Cit. on pp. 70, 72–74, 77–83, 86, 87, 91, 92, 98)

Bottou Léon. (2012). Stochastic gradient descent tricks, In *Neural networks: Tricks of the trade*. Springer. (Cit. on p. 100).

Boyd Stephen, Boyd Stephen P, & Vandenberghe Lieven. (2004). *Convex optimization*. Cambridge university press. (Cit. on pp. 114, 148).

Broscheit Samuel. (2019). Investigating entity knowledge in BERT with simple neural end-to-end entity linking, In *Proceedings of the 23rd conference on computational natural language learning (conll)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/K19-1063. (Cit. on pp. 3, 46, 62, 101, 102)

Brown Tom B., Mann Benjamin, Ryder Nick, Subbiah Melanie, Kaplan Jared, Dhariwal Prafulla, Neelakantan Arvind, Shyam Pranav, Sastry Girish, Askell Amanda, Agarwal Sandhini, Herbert-Voss Ariel, Krueger Gretchen, Henighan Tom, Child Rewon, Ramesh Aditya, Ziegler Daniel M., Wu Jeffrey, Winter Clemens, ... Amodei Dario. (2020). Language models are few-shot learners (Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, & Hsuan-Tien Lin, Eds.). In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, & Hsuan-Tien Lin (Eds.), *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*. (Cit. on pp. 17, 140).

Bunescu Razvan, & Paşca Marius. (2006). Using encyclopedic knowledge for named entity disambiguation, In *11th conference of the European chapter of the association for computational linguistics*, Trento, Italy, Association for Computational Linguistics. (Cit. on p. 3).

Camburu Oana-Maria, Rocktäschel Tim, Lukasiewicz Thomas, & Blunsom Phil. (2018). e-SNLI: Natural language inference with natural language explanations, In *Advances in neural information processing systems (neurips)*. (Cit. on p. 116).

Cao Yu, Fang Meng, & Tao Dacheng. (2019). BAG: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering, In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, Minneapolis, Minnesota, Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1032. (Cit. on pp. 33, 41)

Chen Danqi, Fisch Adam, Weston Jason, & Bordes Antoine. (2017a). Reading Wikipedia to answer open-domain questions, In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Vancouver, Canada, Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1171. (Cit. on p. 46)

Chen Hanjie, & Ji Yangfeng. (2020). Learning variational word masks to improve the interpretability of neural text classifiers, In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.347. (Cit. on p. 137)

Chen Hanjie, Feng Song, Ganhotra Jatin, Wan Hui, Gunasekara Chulaka, Joshi Sachindra, & Ji Yangfeng. (2021). Explaining neural network predictions on sentence pairs via learning word-group masks, In *Proceedings of the 2021 conference of the north american chapter of the association for computational linguistics: Human language technologies*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.306. (Cit. on p. 137)

Chen Henry Y., Zhou Ethan, & Choi Jinho D. (2017b). Robust coreference resolution and entity linking on dialogues: Character identification on TV show transcripts, In *Proceedings of the 21st conference on computational natural language learning (CoNLL 2017)*, Vancouver, Canada, Association for Computational Linguistics. https://doi.org/10.18653/v1/K17-1023. (Cit. on p. 3)

Chen Jianbo, Song Le, Wainwright Martin, & Jordan Michael. (2018). Learning to explain: An information-theoretic perspective on model interpretation (Jennifer Dy & Andreas Krause, Eds.). In Jennifer Dy & Andreas Krause (Eds.), *International conference on machine learning*, Stockholmsmässan, Stockholm Sweden, PMLR. (Cit. on p. 112).

Chen Jiangui, Zhang Ruqing, Guo Jiafeng, Fan Yixing, & Cheng Xueqi. (2022). GERE: generative evidence retrieval for fact verification (Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, & Gabriella Kazai, Eds.). In Enrique Amigó, Pablo Castells, Julio Gonzalo,

Ben Carterette, J. Shane Culpepper, & Gabriella Kazai (Eds.), *SIGIR '22: The 45th international ACM SIGIR conference on research and development in information retrieval, madrid, spain, july 11 - 15, 2022*, ACM. https://doi.org/10.1145/3477495.3531827. (Cit. on p. 67)

Chen Jifan, Lin Shih-Ting, & Durrett Greg. (2019). Multi-hop question answering via reasoning chains. *CoRR*, *abs/1910.02610*arXiv (cit. on pp. 33, 41).

Chen Shuang, Wang Jinpeng, Jiang Feng, & Lin Chin-Yew. (2020). Improving entity linking by modeling latent entity type information, In *Proceedings of the aaai conference on artificial intelligence*. (Cit. on p. 60).

Cheng Xiao, & Roth Dan. (2013). Relational inference for wikification, In *Proceedings of the 2013 conference on empirical methods in natural language processing*, Seattle, Washington, USA, Association for Computational Linguistics. (Cit. on p. 73).

Chicco Davide. (2021). Siamese neural networks: An overview. In Hugh Cartwright (Ed.), *Artificial neural networks* (pp. 73–94). New York, NY, Springer US. https://doi.org/10.1007/978-1-0716-0826-5_3. (Cit. on p. 49)

Cho Kyunghyun, van Merriënboer Bart, Gulcehre Caglar, Bahdanau Dzmitry, Bougares Fethi, Schwenk Holger, & Bengio Yoshua. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation, In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar, Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1179. (Cit. on pp. 116, 117)

Chowdhery Aakanksha, Narang Sharan, Devlin Jacob, Bosma Maarten, Mishra Gaurav, Roberts Adam, Barham Paul, Chung Hyung Won, Sutton Charles, Gehrmann Sebastian, Schuh Parker, Shi Kensen, Tsvyashchenko Sasha, Maynez Joshua, Rao Abhishek, Barnes Parker, Tay Yi, Shazeer Noam, Prabhakaran Vinodkumar, ... Fiedel Noah. (2022). Palm: Scaling language modeling with pathways. *CoRR*, *abs/2204.02311*arXiv. https://doi.org/10.48550/arXiv.2204.02311 (cit. on p. 19)

Conneau Alexis, Khandelwal Kartikay, Goyal Naman, Chaudhary Vishrav, Wenzek Guillaume, Guzmán Francisco, Grave Edouard, Ott Myle, Zettlemoyer Luke, & Stoyanov Veselin. (2020). Unsupervised cross-lingual representation learning at scale, In *Proceedings of the 58th annual meeting of the association for computational linguistics*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.747. (Cit. on pp. 77, 79)

Cormen Thomas H, Leiserson Charles E, Rivest Ronald L, & Stein Clifford. (2009). *Introduction to algorithms*. MIT press. (Cit. on p. 53).

Csordás Róbert, van Steenkiste Sjoerd, & Schmidhuber Jürgen. (2021). Are neural nets modular? inspecting functional modularity through differentiable weight masks, In *International conference on learning representations*. (Cit. on p. 146).

Cucerzan Silviu. (2007). Large-scale named entity disambiguation based on Wikipedia data, In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, Prague, Czech Republic, Association for Computational Linguistics. (Cit. on p. 3).

Curry Amanda Cercas, Papaioannou Ioannis, Suglia Alessandro, Agarwal Shubham, Shalyminov Igor, Xu Xinnuo, Dušek Ondřej, Eshghi Arash, Konstas Ioannis, Rieser Verena, Et al. (2018). Alana v2: Entertaining and informative open-domain social dialogue using ontologies and entity linking. *Alexa Prize Proceedings* (cit. on p. 3).

Dai Zihang, Yang Zhilin, Yang Yiming, Carbonell Jaime, Le Quoc, & Salakhutdinov Ruslan. (2019). Transformer-XL: Attentive language models beyond a fixed-length context, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1285. (Cit. on p. 17)

Daiber Joachim, Jakob Max, Hokamp Chris, & Mendes Pablo N. (2013). Improving efficiency and accuracy in multilingual entity extraction, In *Proceedings of the 9th international conference on semantic systems*. (Cit. on pp. 62, 102).

Date Christopher John. (1975). *An introduction to database systems*. Addison-Wesley. (Cit. on p. 9).

Daza Angel, & Frank Anette. (2018). A sequence-to-sequence model for semantic role labeling, In *Proceedings of the third workshop on representation learning for NLP*, Melbourne, Australia, Association for Computational Linguistics. https://doi.org/10.18653/v1/W18-3027. (Cit. on p. 50)

De Cao Nicola, Aziz Wilker, & Titov Ivan. (2019a). Block neural autoregressive flow (Amir Globerson & Ricardo Silva, Eds.). In Amir Globerson & Ricardo Silva (Eds.), *Proceedings of the thirty-fifth conference on uncertainty in artificial intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, AUAI Press. (Cit. on p. xvi).

De Cao Nicola, Aziz Wilker, & Titov Ivan. (2019b). Question answering by reasoning across documents with graph convolutional networks, In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, Minneapolis, Minnesota, Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1240. (Cit. on pp. xv, 3, 112)

De Cao Nicola, Schlichtkrull Michael Sejr, Aziz Wilker, & Titov Ivan. (2020). How do decisions emerge across layers in neural models? interpretation with differentiable masking, In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.262. (Cit. on pp. xv, 140)

De Cao Nicola, & Aziz Wilker. (2020). The power spherical distribution, In *Proceedings of the 37th international conference on machine learning, ICML 2020 (innf+ workshop), 13-18 july 2020, virtual event*, PMLR. (Cit. on p. xvi).

De Cao Nicola, Izacard Gautier, Riedel Sebastian, & Petroni Fabio. (2021a). Autoregressive entity retrieval, In *9th international conference on learning representations, ICLR 2021, virtual event, austria, may 3-7, 2021*, OpenReview.net. (Cit. on pp. xv, 70, 72, 73, 96, 102, 103, 146).

De Cao Nicola, Aziz Wilker, & Titov Ivan. (2021b). Editing factual knowledge in language models, In *Proceedings of the 2021 conference on empirical methods in natural language processing*, Online, Punta Cana, Dominican Republic, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.522. (Cit. on p. xv)

De Cao Nicola, Aziz Wilker, & Titov Ivan. (2021c). Highly parallel autoregressive entity linking with discriminative correction, In *Proceedings of the 2021 conference on empirical methods in natural language processing*, Online, Punta Cana, Dominican Republic, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.604. (Cit. on pp. xv, 62, 66)

De Cao Nicola, Schmid Leon, Hupkes Dieuwke, & Titov Ivan. (2021d). Sparse interventions in language models with differentiable masking. (Cit. on pp. xvii, 137).

De Cao Nicola, Wu Ledell, Popat Kashyap, Artetxe Mikel, Goyal Naman, Plekhanov Mikhail, Zettlemoyer Luke, Cancedda Nicola, Riedel Sebastian, & Petroni Fabio. (2022). Multilingual Autoregressive Entity Linking. *Transactions of the Association for Computational Linguistics*, *10*, 274–290. https://doi.org/10.1162/tacl_a_00460 (cit. on pp. xvi, 66, 146)

Derczynski Leon, Maynard Diana, Rizzo Giuseppe, Van Erp Marieke, Gorrell Genevieve, Troncy Raphaël, Petrak Johann, & Bontcheva Kalina. (2015). Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, *51*(2), 32–49 (cit. on p. 57).

Devlin Jacob, Chang Ming-Wei, Lee Kenton, & Toutanova Kristina. (2019a). BERT: Pre-training of deep bidirectional transformers for language understanding, In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, Minneapolis, Minnesota, Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423. (Cit. on pp. 5, 23, 28, 58, 73, 140, 150, 151)

Devlin Jacob, Chang Ming-Wei, Lee Kenton, & Toutanova Kristina. (2019b). BERT: Pre-training of deep bidirectional transformers for language understanding, In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, Minneapolis, Minnesota, As-

sociation for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423. (Cit. on pp. 18, 41, 42, 51, 106, 116)

DeYoung Jay, Jain Sarthak, Rajani Nazneen Fatema, Lehman Eric, Xiong Caiming, Socher Richard, & Wallace Byron C. (2020). ERASER: A benchmark to evaluate rationalized NLP models, In *Proceedings of the 58th annual meeting of the association for computational linguistics*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.408. (Cit. on p. 116)

Dhingra Bhuwan, Jin Qiao, Yang Zhilin, Cohen William, & Salakhutdinov Ruslan. (2018). Neural models for reasoning over multiple mentions using coreference, In *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 2 (short papers)*, New Orleans, Louisiana, Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-2007. (Cit. on pp. 22, 23, 25, 32–34)

Dhingra Bhuwan, Cole Jeremy R., Eisenschlos Julian Martin, Gillick Daniel, Eisenstein Jacob, & Cohen William W. (2022). Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, *10*, 257–273. https://doi.org/10.1162/tacl_a_00459 (cit. on p. 162)

Diderot Denis, & d'Alembert Jean Le Rond. (1751). *Encyclopédie, ou, dictionnaire raisonné des sciences, des arts et des métiers* (Vol. 5). Pergamon Press. (Cit. on p. 1).

Dinan Emily, Roller Stephen, Shuster Kurt, Fan Angela, Auli Michael, & Weston Jason. (2019). Wizard of wikipedia: Knowledge-powered conversational agents. *Proceedings of the International Conference on Learning Representations (ICLR)* (cit. on p. 57).

Dredze Mark, McNamee Paul, Rao Delip, Gerber Adam, & Finin Tim. (2010). Entity disambiguation for knowledge base population, In *Proceedings of the 23rd international conference on computational linguistics (coling 2010)*, Beijing, China, Coling 2010 Organizing Committee. (Cit. on p. 3).

Elazar Yanai, Ravfogel Shauli, Jacovi Alon, & Goldberg Yoav. (2021). Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, *9*, 160–175 (cit. on p. 146).

Elman Jeffrey L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211 (cit. on p. 13).

Elsahar Hady, Vougiouklis Pavlos, Remaci Arslen, Gravier Christophe, Hare Jonathon, Laforest Frederique, & Simperl Elena. (2018). T-REx: A large scale alignment of natural language with knowledge base triples, In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*, Miyazaki, Japan, European Language Resources Association (ELRA). (Cit. on p. 57).

Fan Angela, Jernite Yacine, Perez Ethan, Grangier David, Weston Jason, & Auli Michael. (2019). ELI5: Long form question answering, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1346. (Cit. on pp. 22, 57)

Fang Yuwei, Sun Siqi, Gan Zhe, Pillai Rohit, Wang Shuohang, & Liu Jingjing. (2020). Hierarchical graph network for multi-hop question answering, In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.710. (Cit. on p. 41)

Fang Zheng, Cao Yanan, Li Qian, Zhang Dongjie, Zhang Zhenyu, & Liu Yanbing. (2019). Joint entity linking with deep reinforcement learning, In *The world wide web conference*. ACM. (Cit. on p. 60).

Feng Shi, Wallace Eric, Grissom II Alvin, Iyyer Mohit, Rodriguez Pedro, & Boyd-Graber Jordan. (2018). Pathologies of neural models make interpretations difficult, In *Proceedings of the 2018 conference on empirical methods in natural language processing*, Brussels, Belgium, Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1407. (Cit. on p. 106)

Feng Yifan, You Haoxuan, Zhang Zizhao, Ji Rongrong, & Gao Yue. (2019). Hypergraph neural networks, In *The thirty-third AAAI conference on artificial intelligence, AAAI 2019, the thirty-first innovative applications of artificial intelligence conference, IAAI 2019, the ninth AAAI symposium on educational advances in artificial intelligence, EAAI 2019, honolulu, hawaii, usa, january 27 - february 1, 2019*, AAAI Press. https://doi.org/10.1609/aaai.v33i01.33013558. (Cit. on p. 11)

Ferragina Paolo, & Manzini Giovanni. (2000). Opportunistic data structures with applications, In *41st annual symposium on foundations of computer science, FOCS 2000, 12-14 november 2000, redondo beach, california, USA*, IEEE Computer Society. https://doi.org/10.1109/SFCS.2000.892127. (Cit. on p. 67)

Ferrucci David A. (2012). Introduction to "this is watson". *IBM Journal of Research and Development*, *56*(3.4), 1:1–1:15. https://doi.org/10.1147/JRD.2012.2184356 (cit. on p. 46)

Févry Thibault, FitzGerald Nicholas, Soares Livio Baldini, & Kwiatkowski Tom. (2020a). Empirical evaluation of pretraining strategies for supervised entity linking (Dipanjan Das, Hannaneh Hajishirzi, Andrew McCallum, & Sameer Singh, Eds.). In Dipanjan Das, Hannaneh Hajishirzi, Andrew McCallum, & Sameer Singh (Eds.), *Conference on automated knowledge base construction, AKBC 2020, virtual, june 22-24, 2020*. https://doi.org/10.24432/C59G6S. (Cit. on pp. 62, 102)

Févry Thibault, Baldini Soares Livio, FitzGerald Nicholas, Choi Eunsol, & Kwiatkowski Tom. (2020b). Entities as experts: Sparse memory access with entity supervision, In *Proceedings of the 2020 conference on empirical*

*methods in natural language processing (emnlp)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.400. (Cit. on pp. 3, 19, 146)

Finn Chelsea, Abbeel Pieter, & Levine Sergey. (2017). Model-agnostic meta-learning for fast adaptation of deep networks (Doina Precup & Yee Whye Teh, Eds.). In Doina Precup & Yee Whye Teh (Eds.), *Proceedings of the 34th international conference on machine learning, ICML 2017, sydney, nsw, australia, 6-11 august 2017*, PMLR. (Cit. on p. 145).

FitzGerald Nicholas, Bikel Dan, Botha Jan, Gillick Daniel, Kwiatkowski Tom, & McCallum Andrew. (2021). MOLEMAN: Mention-only linking of entities with a mention annotation network, In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 2: Short papers)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-short.37. (Cit. on pp. 82, 83, 92)

Gabrilovich Evgeniy, Ringgaard Michael, & Subramanya Amarnag. (2013). Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). (Cit. on p. 56).

Ganea Octavian-Eugen, & Hofmann Thomas. (2017). Deep joint entity disambiguation with local neural attention, In *Proceedings of the 2017 conference on empirical methods in natural language processing*, Copenhagen, Denmark, Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1277. (Cit. on pp. 56, 60, 80, 98, 100)

Gillick Daniel, Kulkarni Sayali, Lansing Larry, Presta Alessandro, Baldridge Jason, Ie Eugene, & Garcia-Olano Diego. (2019). Learning dense representations for entity retrieval, In *Proceedings of the 23rd conference on computational natural language learning (conll)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/K19-1049. (Cit. on pp. 50, 51)

Gilmer Justin, Schoenholz Samuel S., Riley Patrick F., Vinyals Oriol, & Dahl George E. (2017). Neural message passing for quantum chemistry (Doina Precup & Yee Whye Teh, Eds.). In Doina Precup & Yee Whye Teh (Eds.), *Proceedings of the 34th international conference on machine learning, ICML 2017, sydney, nsw, australia, 6-11 august 2017*, PMLR. (Cit. on p. 11).

Goldberg Yoav. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, *57*(1), 345–420 (cit. on p. 12).

Goodfellow Ian J., Bengio Yoshua, & Courville Aaron C. (2016). *Deep learning*. MIT Press. (Cit. on pp. 7, 16, 18).

Gori Marco, Monfardini Gabriele, & Scarselli Franco. (2005). A new model for learning in graph domains, In *Proceedings. 2005 ieee international joint conference on neural networks.* (Cit. on p. 10).

Guan Chaoyu, Wang Xiting, Zhang Quanshi, Chen Runjin, He Di, & Xie Xing. (2019). Towards a deep and unified understanding of deep neural models in NLP (Kamalika Chaudhuri & Ruslan Salakhutdinov, Eds.). In Kamalika Chaudhuri & Ruslan Salakhutdinov (Eds.), *International conference on machine learning*, Long Beach, California, USA, PMLR. (Cit. on pp. 107, 108, 111, 117, 119, 120, 123, 124, 126, 134).

Guo Ruiqi, Sun Philip, Lindgren Erik, Geng Quan, Simcha David, Chern Felix, & Kumar Sanjiv. (2020). Accelerating large-scale inference with anisotropic vector quantization, In *Proceedings of the 37th international conference on machine learning, ICML 2020, 13-18 july 2020, virtual event*, PMLR. (Cit. on p. 50).

Guo Zhaochen, & Barbosa Denilson. (2018). Robust named entity disambiguation with random walks. *Semantic Web*, *9*(4), 459–479 (cit. on pp. 56, 60).

Gururangan Suchin, Swayamdipta Swabha, Levy Omer, Schwartz Roy, Bowman Samuel, & Smith Noah A. (2018). Annotation artifacts in natural language inference data, In *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 2 (short papers)*, New Orleans, Louisiana, Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-2017. (Cit. on p. 106)

Guu Kelvin, Lee Kenton, Tung Zora, Pasupat Panupong, & Chang Mingwei. (2020). Retrieval augmented language model pre-training (Hal Daumé III & Aarti Singh, Eds.). In Hal Daumé III & Aarti Singh (Eds.), *Proceedings of the 37th international conference on machine learning*, PMLR. (Cit. on pp. 19, 51).

Ha David, Dai Andrew M., & Le Quoc V. (2017). Hypernetworks, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on p. 147).

Hao Yaru, Dong Li, Wei Furu, & Xu Ke. (2019). Visualizing and understanding the effectiveness of BERT, In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1424. (Cit. on p. 111)

Hayes-Roth Frederick. (1983). *Building expert systems* (Vol. 1). Addison-Wesley. (Cit. on p. 9).

He Ruining, Ravula Anirudh, Kanagal Bhargav, & Ainslie Joshua. (2021). RealFormer: Transformer likes residual attention, In *Findings of the association for computational linguistics: Acl-ijcnlp 2021*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.81. (Cit. on pp. 33, 42)

Henderson Matthew L., Al-Rfou Rami, Strope Brian, Sung Yun-Hsuan, Lukács László, Guo Ruiqi, Kumar Sanjiv, Miklos Balint, & Kurzweil Ray. (2017). Efficient natural language response suggestion for smart reply. *CoRR*, *abs/1705.00652*arXiv (cit. on p. 50).

Hermann Karl Moritz, Kočiský Tomáš, Grefenstette Edward, Espeholt Lasse, Kay Will, Suleyman Mustafa, & Blunsom Phil. (2015). Teaching machines to read and comprehend, In *Proceedings of the 28th international conference on neural information processing systems - volume 1*, Montreal, Canada, MIT Press. (Cit. on p. 22).

Hochreiter Sepp, & Schmidhuber Jürgen. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780 (cit. on pp. 13, 98, 99, 148).

Hoffart Johannes, Yosef Mohamed Amir, Bordino Ilaria, Fürstenau Hagen, Pinkal Manfred, Spaniol Marc, Taneva Bilyana, Thater Stefan, & Weikum Gerhard. (2011). Robust disambiguation of named entities in text, In *Proceedings of the 2011 conference on empirical methods in natural language processing*, Edinburgh, Scotland, UK., Association for Computational Linguistics. (Cit. on pp. 3, 46, 56, 62, 98, 100, 102).

Hoffart Johannes, Seufert Stephan, Nguyen Dat Ba, Theobald Martin, & Weikum Gerhard. (2012). KORE: Keyphrase Overlap Relatedness for Entity Disambiguation, In *Proceedings of the 21st acm international conference on information and knowledge management*, Maui, Hawaii, USA, Association for Computing Machinery. https://doi.org/10.1145/2396761.2396832. (Cit. on p. 57)

Hoffmann Raphael, Zhang Congle, Ling Xiao, Zettlemoyer Luke, & Weld Daniel S. (2011). Knowledge-based weak supervision for information extraction of overlapping relations, In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, Portland, Oregon, USA, Association for Computational Linguistics. (Cit. on p. 98).

Hokamp Chris, & Liu Qun. (2017). Lexically constrained decoding for sequence generation using grid beam search, In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Vancouver, Canada, Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1141. (Cit. on p. 51)

Holstein Kenneth, Wortman Vaughan Jennifer, Daumé Hal, Dudik Miro, & Wallach Hanna. (2019). Improving fairness in machine learning systems: What do industry practitioners need?, In *Proceedings of the 2019 chi conference on human factors in computing systems*, Glasgow, Scotland Uk, Association for Computing Machinery. https://doi.org/10.1145/3290605.3300830. (Cit. on p. 106)

Hopfield John J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554–2558 (cit. on p. 7).

Huang Hongzhao, Heck Larry P., & Ji Heng. (2015). Leveraging deep neural networks and knowledge graphs for entity disambiguation. *CoRR*, *abs/1504.07678*arXiv (cit. on pp. 3, 46).

Humeau Samuel, Shuster Kurt, Lachaux Marie-Anne, & Weston Jason. (2020). Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring, In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*, OpenReview.net. (Cit. on pp. 46, 50).

Ilyas Ihab F., Rekatsinas Theodoros, Konda Vishnu, Pound Jeffrey, Qi Xiaoguang, & Soliman Mohamed A. (2022). Saga: A platform for continuous construction and serving of knowledge at scale (Zachary Ives, Angela Bonifati, & Amr El Abbadi, Eds.). In Zachary Ives, Angela Bonifati, & Amr El Abbadi (Eds.), *SIGMOD '22: International conference on management of data, philadelphia, pa, usa, june 12 - 17, 2022*, ACM. https://doi.org/10.1145/3514221.3526049. (Cit. on p. 10)

Izacard Gautier, Petroni Fabio, Hosseini Lucas, De Cao Nicola, Riedel Sebastian, & Grave Edouard. (2020). A memory efficient baseline for open domain question answering. (Cit. on pp. xvi, 22, 73).

Jacovi Alon, & Goldberg Yoav. (2020). Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness?, In *Proceedings of the 58th annual meeting of the association for computational linguistics*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.386. (Cit. on pp. 106, 116)

Jain Sarthak, & Wallace Byron C. (2019). Attention is not Explanation, In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, Minneapolis, Minnesota, Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1357. (Cit. on p. 106)

James William, Burkhardt Frederick, Bowers Fredson, & Skrupskelis Ignas K. (1890). *The principles of psychology* (Vol. 1). Macmillan London. (Cit. on p. 13).

Jang Eric, Gu Shixiang, & Poole Ben. (2017). Categorical reparameterization with gumbel-softmax, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on pp. 110, 112, 137).

Jegou Herve, Douze Matthijs, & Schmid Cordelia. (2010). Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, *33*(1), 117–128 (cit. on p. 50).

Ji Heng, Nothman Joel, Hachey Ben, & Florian Radu. (2015). Overview of TAC-KBP2015 Tri-lingual Entity Discovery and Linking, In *Tac*. (Cit. on pp. 70, 72, 76, 80, 81, 98).

Jiang Zhengbao, Xu Frank F., Araki Jun, & Neubig Graham. (2020). How can we know what language models know? *Transactions of the Association for Computational Linguistics*, *8*, 423–438. https://doi.org/10.1162/tacl_a_00324 (cit. on p. 146)

Jin Xisen, Wei Zhongyu, Du Junyi, Xue Xiangyang, & Ren Xiang. (2020). Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models, In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*, OpenReview.net. (Cit. on pp. 111, 112).

Johnson Jeff, Douze Matthijs, & Jégou Hervé. (2019). Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, *7*(3), 535–547. https://doi.org/10.1109/TBDATA.2019.2921572 (cit. on pp. 4, 46, 50)

Jordan Jordan I. (1986). Serial order: A parallel distributed processing approach (cit. on p. 13).

Joshi Mandar, Choi Eunsol, Weld Daniel, & Zettlemoyer Luke. (2017). TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension, In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Vancouver, Canada, Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1147. (Cit. on pp. 22, 57, 150)

Josifoski Martin, De Cao Nicola, Peyrard Maxime, Petroni Fabio, & West Robert. (2022). GenIE: Generative information extraction, In *Proceedings of the 2022 conference of the north american chapter of the association for computational linguistics: Human language technologies*, Seattle, United States, Association for Computational Linguistics. (Cit. on pp. xvii, 66).

Junczys-Dowmunt Marcin, Grundkiewicz Roman, Dwojak Tomasz, Hoang Hieu, Heafield Kenneth, Neckermann Tom, Seide Frank, Germann Ulrich, Aji Alham Fikri, Bogoychev Nikolay, Martins André F. T., & Birch Alexandra. (2018). Marian: Fast neural machine translation in C++, In *Proceedings of ACL 2018, system demonstrations*, Melbourne, Australia, Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-4020. (Cit. on p. 151)

Kannan Ravi Manoj Prabhakar, Singh Kuldeep, Mulang' Isaiah Onando, Shekarpour Saeedeh, Hoffart Johannes, & Lehmann Jens. (2021). CHOLAN: A modular approach for neural entity linking on Wikipedia and Wikidata, In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: Main volume*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.40. (Cit. on pp. 62, 102)

Karpukhin Vladimir, Oguz Barlas, Min Sewon, Lewis Patrick, Wu Ledell, Edunov Sergey, Chen Danqi, & Yih Wen-tau. (2020). Dense passage retrieval for open-domain question answering, In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, Online, Associ-

ation for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.550. (Cit. on pp. 3, 22, 42, 46, 50, 51, 58, 61, 63)

Kaushik Divyansh, & Lipton Zachary C. (2018). How much reading does reading comprehension require? a critical investigation of popular benchmarks, In *Proceedings of the 2018 conference on empirical methods in natural language processing*, Brussels, Belgium, Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1546. (Cit. on p. 106)

Kiefer Jack, & Wolfowitz Jacob. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 462–466 (cit. on p. 100).

Kim Been. (2015). *Interactive and interpretable machine learning models for human machine collaboration* (Doctoral dissertation). Massachusetts Institute of Technology. (Cit. on p. 106).

Kingma Diederik P., & Welling Max. (2014). Auto-encoding variational bayes (Yoshua Bengio & Yann LeCun, Eds.). In Yoshua Bengio & Yann LeCun (Eds.), *2nd international conference on learning representations, ICLR 2014, banff, ab, canada, april 14-16, 2014, conference track proceedings*. (Cit. on pp. 110, 114).

Kingma Diederik P., & Ba Jimmy. (2015). Adam: A method for stochastic optimization (Yoshua Bengio & Yann LeCun, Eds.). In Yoshua Bengio & Yann LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. (Cit. on pp. 31, 54, 81, 101, 119, 152).

Kipf Thomas N., & Welling Max. (2017). Semi-supervised classification with graph convolutional networks, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on pp. 21, 23, 27).

Kirkpatrick J., Pascanu Razvan, Rabinowitz Neil C., Veness J., Desjardins G., Rusu Andrei A., Milan K., Quan John, Ramalho Tiago, Grabska-Barwinska Agnieszka, Hassabis Demis, Clopath C., Kumaran D., & Hadsell Raia. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*, 3521–3526 (cit. on p. 146).

Kočiský Tomáš, Schwarz Jonathan, Blunsom Phil, Dyer Chris, Hermann Karl Moritz, Melis Gábor, & Grefenstette Edward. (2018). The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, *6*, 317–328. https://doi.org/10.1162/tacl_a_00023 (cit. on p. 22)

Kolitsas Nikolaos, Ganea Octavian-Eugen, & Hofmann Thomas. (2018). End-to-end neural entity linking, In *Proceedings of the 22nd conference on computational natural language learning*, Brussels, Belgium, Association for Computational Linguistics. https://doi.org/10.18653/v1/K18-1050. (Cit. on pp. 56, 57, 62, 98, 101–103)

Konstas Ioannis, Iyer Srinivasan, Yatskar Mark, Choi Yejin, & Zettlemoyer Luke. (2017). Neural AMR: Sequence-to-sequence models for parsing and generation, In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Vancouver, Canada, Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1014. (Cit. on p. 50)

Kovaleva Olga, Romanov Alexey, Rogers Anna, & Rumshisky Anna. (2019). Revealing the dark secrets of BERT, In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1445. (Cit. on p. 129)

Krueger David, Huang Chin-Wei, Islam Riashat, Turner Ryan, Lacoste Alexandre, & Courville Aaron. (2017). Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759* (cit. on p. 148).

Kulis Brian. (2013). Metric learning: A survey. *Foundations and Trends in Machine Learning*, *5*(4), 287–364. https://doi.org/10.1561/2200000019 (cit. on p. 50)

Kwiatkowski Tom, Palomaki Jennimaria, Redfield Olivia, Collins Michael, Parikh Ankur, Alberti Chris, Epstein Danielle, Polosukhin Illia, Devlin Jacob, Lee Kenton, Toutanova Kristina, Jones Llion, Kelcey Matthew, Chang Ming-Wei, Dai Andrew M., Uszkoreit Jakob, Le Quoc, & Petrov Slav. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, *7*, 452–466. https://doi.org/10.1162/tacl_a_00276 (cit. on pp. 22, 57, 150)

Lai Guokun, Xie Qizhe, Liu Hanxiao, Yang Yiming, & Hovy Eduard. (2017). RACE: Large-scale ReAding comprehension dataset from examinations, In *Proceedings of the 2017 conference on empirical methods in natural language processing*, Copenhagen, Denmark, Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1082. (Cit. on p. 22)

Lakretz Yair, Kruszewski German, Desbordes Theo, Hupkes Dieuwke, Dehaene Stanislas, & Baroni Marco. (2019). The emergence of number and syntax units in LSTM language models, In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, Minneapolis, Minnesota, Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1002. (Cit. on p. 146)

Le Phong, & Titov Ivan. (2018). Improving entity linking by modeling latent relations between mentions, In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Melbourne, Australia, Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1148. (Cit. on pp. 3, 46, 55, 56, 59)

Le Phong, & Titov Ivan. (2019). Boosting entity linking performance by leveraging unlabeled documents, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1187. (Cit. on p. 60)

Leaman Robert, & Gonzalez Graciela. (2008). BANNER: An executable survey of advances in biomedical named entity recognition [13th Pacific Symposium on Biocomputing, PSB 2008 ; Conference date: 04-01-2008 Through 08-01-2008], In *Pacific symposium on biocomputing 2008, psb 2008*. 13th Pacific Symposium on Biocomputing, PSB 2008 ; Conference date: 04-01-2008 Through 08-01-2008. (Cit. on p. 3).

Lee Hyunji, Yang Sohee, Oh Hanseok, & Seo Minjoon. (2022). Generative multi-hop retrieval. *CoRR*, *abs/2204.13596*arXiv. https://doi.org/10.48550/arXiv.2204.13596 (cit. on p. 67)

Lee Kenton, He Luheng, Lewis Mike, & Zettlemoyer Luke. (2017). End-to-end neural coreference resolution, In *Proceedings of the 2017 conference on empirical methods in natural language processing*, Copenhagen, Denmark, Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1018. (Cit. on p. 26)

Lee Nayeon, Li Belinda Z., Wang Sinong, Yih Wen-tau, Ma Hao, & Khabsa Madian. (2020). Language models as fact checkers?, In *Proceedings of the third workshop on fact extraction and verification (fever)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.fever-1.5. (Cit. on p. 150)

Lei Tao, Barzilay Regina, & Jaakkola Tommi. (2016). Rationalizing neural predictions, In *Proceedings of the 2016 conference on empirical methods in natural language processing*, Austin, Texas, Association for Computational Linguistics. https://doi.org/10.18653/v1/D16-1011. (Cit. on p. 112)

Lerer Adam, Wu Ledell, Shen Jiajun, Lacroix Timothee, Wehrstedt Luca, Bose Abhijit, & Peysakhovich Alex. (2019). Pytorch-biggraph: A large scale graph embedding system. *Proceedings of Machine Learning and Systems*, *1*, 120–131 (cit. on p. 50).

Leskovec Jure, Rajaraman Anand, & Ullman Jeffrey David. (2014). *Mining of massive datasets* (2nd). USA, Cambridge University Press. (Cit. on p. 59).

Levy Omer, Seo Minjoon, Choi Eunsol, & Zettlemoyer Luke. (2017). Zero-shot relation extraction via reading comprehension, In *Proceedings of the 21st conference on computational natural language learning (CoNLL 2017)*, Vancouver, Canada, Association for Computational Linguistics. https://doi.org/10.18653/v1/K17-1034. (Cit. on pp. 57, 150)

Lewis Mike, Liu Yinhan, Goyal Naman, Ghazvininejad Marjan, Mohamed Abdelrahman, Levy Omer, Stoyanov Veselin, & Zettlemoyer Luke. (2020a). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, In *Proceedings of the 58th annual*

*meeting of the association for computational linguistics*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.703. (Cit. on pp. 18, 48, 52, 77, 140, 150, 151)

Lewis Patrick, Perez Ethan, Piktus Aleksandra, Petroni Fabio, Karpukhin Vladimir, Goyal Naman, Küttler Heinrich, Lewis Mike, Yih Wen-tau, Rocktäschel Tim, Riedel Sebastian, & Kiela Douwe. (2020b). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin, Eds.). In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems*, Curran Associates, Inc. (Cit. on pp. 19, 22, 46, 51, 59, 63).

Lewis Patrick, Wu Yuxiang, Liu Linqing, Minervini Pasquale, Küttler Heinrich, Piktus Aleksandra, Stenetorp Pontus, & Riedel Sebastian. (2021). PAQ: 65 Million Probably-Asked Questions and What You Can Do With Them. *Transactions of the Association for Computational Linguistics*, *9*, 1098–1115. https://doi.org/10.1162/tacl_a_00415 (cit. on pp. 22, 73)

Li Belinda Z., Min Sewon, Iyer Srinivasan, Mehdad Yashar, & Yih Wen-tau. (2020). Efficient one-pass end-to-end entity linking for questions, In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.522. (Cit. on p. 98)

Li Jiwei, Monroe Will, & Jurafsky Dan. (2016). Understanding neural networks through representation erasure. *arXiv:1612.08220* (cit. on p. 106).

Liu Jiangming, Cohen Shay B., & Lapata Mirella. (2018). Discourse representation structure parsing, In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Melbourne, Australia, Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1040. (Cit. on p. 50)

Liu Xiao, Zheng Yanan, Du Zhengxiao, Ding Ming, Qian Yujie, Yang Zhilin, & Tang Jie. (2021). GPT Understands, Too. *arXiv preprint arXiv:2103.10385* (cit. on p. 146).

Liu Yinhan, Ott Myle, Goyal Naman, Du Jingfei, Joshi Mandar, Chen Danqi, Levy Omer, Lewis Mike, Zettlemoyer Luke, & Stoyanov Veselin. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, *abs/1907.11692*arXiv (cit. on pp. 18, 100).

Liu Yinhan, Gu Jiatao, Goyal Naman, Li Xian, Edunov Sergey, Ghazvininejad Marjan, Lewis Mike, & Zettlemoyer Luke. (2020). Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, *8*, 726–742. https://doi.org/10.1162/tacl_a_00343 (cit. on p. 77)

Logeswaran Lajanugen, Chang Ming-Wei, Lee Kenton, Toutanova Kristina, Devlin Jacob, & Lee Honglak. (2019). Zero-shot entity linking by reading entity descriptions, In *Proceedings of the 57th annual meeting of the association for*

*computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1335. (Cit. on pp. 3, 46)

Louizos Christos, Welling Max, & Kingma Diederik P. (2018). Learning sparse neural networks through l_0 regularization, In *6th international conference on learning representations, ICLR 2018, vancouver, bc, canada, april 30 - may 3, 2018, conference track proceedings*, OpenReview.net. (Cit. on pp. 111, 112, 114).

Luo Gang, Huang Xiaojiang, Lin Chin-Yew, & Nie Zaiqing. (2015). Joint entity recognition and disambiguation, In *Proceedings of the 2015 conference on empirical methods in natural language processing*, Lisbon, Portugal, Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1104. (Cit. on p. 100)

Luong Thang, Pham Hieu, & Manning Christopher D. (2015). Effective approaches to attention-based neural machine translation, In *Proceedings of the 2015 conference on empirical methods in natural language processing*, Lisbon, Portugal, Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1166. (Cit. on p. 13)

Maddison Chris J., Mnih Andriy, & Teh Yee Whye. (2017). The concrete distribution: A continuous relaxation of discrete random variables, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on pp. 110, 112, 137).

Maillard Jean, Karpukhin Vladimir, Petroni Fabio, Yih Wen-tau, Oguz Barlas, Stoyanov Veselin, & Ghosh Gargi. (2021). Multi-task retrieval for knowledge-intensive tasks, In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.89. (Cit. on p. 61)

Malkov Yu A, & Yashunin Dmitry A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, *42*(4), 824–836 (cit. on p. 50).

Marcheggiani Diego, & Titov Ivan. (2017). Encoding sentences with graph convolutional networks for semantic role labeling, In *Proceedings of the 2017 conference on empirical methods in natural language processing*, Copenhagen, Denmark, Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1159. (Cit. on pp. 25, 112)

Martinez-Rodriguez Jose L, Hogan Aidan, & Lopez-Arevalo Ivan. (2020). Information extraction meets the semantic web: A survey. *Semantic Web*, 1–81 (cit. on p. 3).

Martins Pedro Henrique, Marinho Zita, & Martins André F. T. (2019). Joint learning of named entity recognition and entity linking, In *Proceedings of*

*the 57th annual meeting of the association for computational linguistics: Student research workshop*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-2026. (Cit. on pp. 62, 98, 102)

Massarelli Luca, Petroni Fabio, Piktus Aleksandra, Ott Myle, Rocktäschel Tim, Plachouras Vassilis, Silvestri Fabrizio, & Riedel Sebastian. (2019). How decoding strategies affect the verifiability of generated text. *arXiv preprint arXiv:1911.03587* (cit. on p. 52).

McGuinness Deborah L, Van Harmelen Frank Et al. (2004). Owl web ontology language overview. *W3C recommendation*, *10*(10), 2004 (cit. on p. 7).

McNamee Paul, Mayfield James, Lawrie Dawn, Oard Douglas, & Doermann David. (2011). Cross-language entity linking, In *Proceedings of 5th international joint conference on natural language processing*, Chiang Mai, Thailand, Asian Federation of Natural Language Processing. (Cit. on pp. 70, 72, 73, 98).

Michel Paul, Levy Omer, & Neubig Graham. (2019). Are sixteen heads really better than one? (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett, Eds.). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32*. Curran Associates, Inc. (Cit. on p. 122).

Mikolov Tomás, Sutskever Ilya, Chen Kai, Corrado Gregory S., & Dean Jeffrey. (2013). Distributed representations of words and phrases and their compositionality (Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, & Kilian Q. Weinberger, Eds.). In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, & Kilian Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. (Cit. on pp. 16, 28).

Min Sewon, Boyd-Graber Jordan, Alberti Chris, Chen Danqi, Choi Eunsol, Collins Michael, Guu Kelvin, Hajishirzi Hannaneh, Lee Kenton, Palomaki Jennimaria, Raffel Colin, Roberts Adam, Kwiatkowski Tom, Lewis Patrick, Wu Yuxiang, Küttler Heinrich, Liu Linqing, Minervini Pasquale, Stenetorp Pontus, ... Yih Wen-tau. (2021). Neurips 2020 efficientqa competition: Systems, analyses and lessons learned (Hugo Jair Escalante & Katja Hofmann, Eds.). In Hugo Jair Escalante & Katja Hofmann (Eds.), *Proceedings of the neurips 2020 competition and demonstration track*, PMLR. (Cit. on pp. xvi, 73).

Mitchell Eric, Lin Charles, Bosselut Antoine, Finn Chelsea, & Manning Christopher D. (2022a). Fast model editing at scale, In *The tenth international conference on learning representations, ICLR 2022, virtual event, april 25-29, 2022*, OpenReview.net. (Cit. on pp. 162, 168).

Mitchell Eric, Lin Charles, Bosselut Antoine, Manning Christopher D, & Finn Chelsea. (2022b). Memory-based model editing at scale (Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, & Sivan Sabato, Eds.). In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, & Sivan Sabato (Eds.), *Proceedings of the 39th international conference on machine learning*, PMLR. (Cit. on pp. 162, 168).

Moro Andrea, Raganato Alessandro, & Navigli Roberto. (2014). Entity linking meets word sense disambiguation: A unified approach. *Transactions of the Association for Computational Linguistics*, *2*, 231–244. https://doi.org/10.1162/tacl_a_00179 (cit. on pp. 62, 102)

Mrini Khalil, Nie Shaoliang, Gu Jiatao, Wang Sinong, Sanjabi Maziar, & Firooz Hamed. (2022). Detection, disambiguation, re-ranking: Autoregressive entity linking as a multi-task problem, In *Findings of the association for computational linguistics: Acl 2022*, Dublin, Ireland, Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.findings-acl.156. (Cit. on pp. 62, 68, 102)

Mulang' Isaiah Onando, Singh Kuldeep, Prabhu Chaitali, Nadgeri Abhishek, Hoffart Johannes, & Lehmann Jens. (2020). Evaluating the impact of knowledge graph context on entity disambiguation models, In *Proceedings of the 29th acm international conference on information & knowledge management*. (Cit. on p. 60).

Murdoch W. James, & Szlam Arthur. (2017). Automatic rule extraction from long short term memory networks, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on p. 111).

Nalmpantis Angelos, Panagiotopoulos Apostolos, Gkountouras John, Papakostas Konstantinos, & Aziz Wilker. (2023). VISION DIFFMASK: faithful interpretation of vision transformers with differentiable patch masking. *CoRR*, *abs/2304.06391* arXiv. https://doi.org/10.48550/arXiv.2304.06391 (cit. on p. 138)

Nie Weili, Zhang Yang, & Patel Ankit. (2018). A theoretical explanation for perplexing behaviors of backpropagation-based visualizations (Jennifer Dy & Andreas Krause, Eds.). In Jennifer Dy & Andreas Krause (Eds.), *International conference on machine learning*, Stockholmsmässan, Stockholm Sweden, PMLR. (Cit. on p. 106).

Nie Yixin, Wang Songhe, & Bansal Mohit. (2019). Revealing the importance of semantic retrieval for machine reading at scale, In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1258. (Cit. on p. 3)

Nogueira Rodrigo, Jiang Zhiying, Pradeep Ronak, & Lin Jimmy. (2020). Document ranking with a pretrained sequence-to-sequence model, In *Findings of the association for computational linguistics: Emnlp 2020*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.63. (Cit. on p. 52)

Noy Natasha, Gao Yuqing, Jain Anshu, Narayanan Anant, Patterson Alan, & Taylor Jamie. (2019). Industry-scale knowledge graphs: Lessons and challenges. *Communications of the ACM, 62 (8)*, 36–43 (cit. on p. 10).

Nuzzolese Andrea Giovanni, Gentile Anna Lisa, Presutti Valentina, Gangemi Aldo, Garigliotti Darío, & Navigli Roberto. (2015). Open knowledge extraction challenge, In *Semantic web evaluation challenges*. Springer. (Cit. on p. 57).

Onoe Yasumasa, & Durrett Greg. (2020). Fine-grained entity typing for domain independent entity linking, *34*, 8576–8583. https://doi.org/10.1609/aaai.v34i05.6380 (cit. on p. 66)

Ott Myle, Edunov Sergey, Baevski Alexei, Fan Angela, Gross Sam, Ng Nathan, Grangier David, & Auli Michael. (2019). Fairseq: A fast, extensible toolkit for sequence modeling, In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics (demonstrations)*, Minneapolis, Minnesota, Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-4009. (Cit. on pp. 54, 81)

Pan Sinno Jialin, & Yang Qiang. (2010). A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191 (cit. on p. 16)

Peng Nanyun, Poon Hoifung, Quirk Chris, Toutanova Kristina, & Yih Wen-tau. (2017). Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, *5*, 101–115. https://doi.org/10.1162/tacl_a_00049 (cit. on p. 25)

Pennington Jeffrey, Socher Richard, & Manning Christopher. (2014a). GloVe: Global vectors for word representation, In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar, Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162. (Cit. on p. 16)

Pennington Jeffrey, Socher Richard, & Manning Christopher. (2014b). GloVe: Global vectors for word representation, In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar, Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162. (Cit. on pp. 28, 32)

Pershina Maria, He Yifan, & Grishman Ralph. (2015). Personalized page rank for named entity disambiguation, In *Proceedings of the 2015 conference of the north American chapter of the association for computational linguistics: Human language technologies*, Denver, Colorado, Association for Computational Linguistics. https://doi.org/10.3115/v1/N15-1026. (Cit. on p. 100)

Peters Matthew E., Neumann Mark, Iyyer Mohit, Gardner Matt, Clark Christopher, Lee Kenton, & Zettlemoyer Luke. (2018). Deep contextualized word representations, In *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)*, New Orleans, Louisiana, Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1202. (Cit. on pp. 17, 23, 28)

Peters Matthew E., Neumann Mark, Logan Robert, Schwartz Roy, Joshi Vidur, Singh Sameer, & Smith Noah A. (2019). Knowledge enhanced contextual word representations, In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1005. (Cit. on pp. 62, 102)

Petroni Fabio, Rocktäschel Tim, Riedel Sebastian, Lewis Patrick, Bakhtin Anton, Wu Yuxiang, & Miller Alexander. (2019). Language models as knowledge bases?, In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1250. (Cit. on pp. 48, 146)

Petroni Fabio, Lewis Patrick, Piktus Aleksandra, Rocktäschel Tim, Wu Yuxiang, Miller Alexander H., & Riedel Sebastian. (2020). How context affects language models' factual predictions, In *Automated knowledge base construction*. (Cit. on p. 52).

Petroni Fabio, Piktus Aleksandra, Fan Angela, Lewis Patrick, Yazdani Majid, De Cao Nicola, Thorne James, Jernite Yacine, Karpukhin Vladimir, Maillard Jean, Plachouras Vassilis, Rocktäschel Tim, & Riedel Sebastian. (2021). KILT: A benchmark for knowledge intensive language tasks, In *Proceedings of the 2021 conference of the north american chapter of the association for computational linguistics: Human language technologies*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.200. (Cit. on pp. xvi, 49, 52, 56, 57, 61, 73, 150, 151)

Piccinno Francesco, & Ferragina Paolo. (2014). From tagme to wat: A new entity annotator, In *Proceedings of the first international workshop on entity recognition and disambiguation*, Gold Coast, Queensland, Australia, Association for Computing Machinery. https://doi.org/10.1145/2633211.2634350. (Cit. on pp. 3, 46, 62, 98, 102)

Post Matt, & Vilar David. (2018). Fast lexically constrained decoding with dynamic beam allocation for neural machine translation, In *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long*

*papers)*, New Orleans, Louisiana, Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1119. (Cit. on p. 51)

Radford Alec, Narasimhan Karthik, Salimans Tim, Sutskever Ilya, Et al. (2018a). Improving language understanding by generative pre-training (cit. on p. 17).

Radford Alec, Narasimhan Karthik, Salimans Tim, & Sutskever Ilya. (2018b). Improving language understanding with unsupervised learning. *Technical report, OpenAI* (cit. on pp. 23, 28).

Radford Alec, Wu Jeffrey, Child Rewon, Luan David, Amodei Dario, & Sutskever Ilya. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9 (cit. on pp. 17, 48, 140).

Rae Jack W., Borgeaud Sebastian, Cai Trevor, Millican Katie, Hoffmann Jordan, Song H. Francis, Aslanides John, Henderson Sarah, Ring Roman, Young Susannah, Rutherford Eliza, Hennigan Tom, Menick Jacob, Cassirer Albin, Powell Richard, van den Driessche George, Hendricks Lisa Anne, Rauh Maribeth, Huang Po-Sen, ... Irving Geoffrey. (2021). Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, *abs/2112.11446*arXiv (cit. on p. 19).

Raffel Colin, Shazeer Noam, Roberts Adam, Lee Katherine, Narang Sharan, Matena Michael, Zhou Yanqi, Li Wei, & Liu Peter J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1–67 (cit. on pp. 18, 140).

Raiman Jonathan, & Raiman Olivier. (2018). Deeptype: Multilingual entity linking by neural type system evolution, In *Proceedings of the aaai conference on artificial intelligence*. (Cit. on p. 60).

Raison Martin, Mazaré Pierre-Emmanuel, Das Rajarshi, & Bordes Antoine. (2018). Weaver: Deep co-encoding of questions and documents for machine reading. *In Proceedings of the International Conference on Machine Learning (ICML)* (cit. on pp. 22, 23, 25, 32, 33).

Rajpurkar Pranav, Zhang Jian, Lopyrev Konstantin, & Liang Percy. (2016a). SQuAD: 100,000+ questions for machine comprehension of text, In *Proceedings of the 2016 conference on empirical methods in natural language processing*, Austin, Texas, Association for Computational Linguistics. https://doi.org/10.18653/v1/D16-1264. (Cit. on pp. 22, 150)

Rajpurkar Pranav, Zhang Jian, Lopyrev Konstantin, & Liang Percy. (2016b). SQuAD: 100,000+ questions for machine comprehension of text, In *Proceedings of the 2016 conference on empirical methods in natural language processing*, Austin, Texas, Association for Computational Linguistics. https://doi.org/10.18653/v1/D16-1264. (Cit. on pp. 118, 128)

Reimers Nils, & Gurevych Iryna. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks, In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, Hong Kong,

China, Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410. (Cit. on p. 49)

Rezende Danilo Jimenez, Mohamed Shakir, & Wierstra Daan. (2014). Stochastic backpropagation and approximate inference in deep generative models (Eric P. Xing & Tony Jebara, Eds.). In Eric P. Xing & Tony Jebara (Eds.), *International conference on machine learning*, Bejing, China, PMLR. (Cit. on p. 110).

Ribeiro Marco, Singh Sameer, & Guestrin Carlos. (2016a). "why should I trust you?": Explaining the predictions of any classifier, In *Proceedings of the 2016 conference of the north American chapter of the association for computational linguistics: Demonstrations*, San Diego, California, Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-3020. (Cit. on p. 106)

Ribeiro Marco Tulio, Singh Sameer, & Guestrin Carlos. (2016b). Model-agnostic interpretability of machine learning. *International Conference on Machine Learning (ICML) Workshop on Human Interpretability in Machine Learning* (cit. on p. 140).

Robbins Herbert, & Monro Sutton. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400–407 (cit. on p. 100).

Roberts Adam, Raffel Colin, & Shazeer Noam. (2020). How much knowledge can you pack into the parameters of a language model?, In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.437. (Cit. on pp. 146, 150)

Robertson Stephen, Zaragoza Hugo Et al. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, *3*(4), 333–389 (cit. on p. 51).

Rockafellar R Tyrrell. (1970). *Convex analysis* (Vol. 18). Princeton university press. (Cit. on p. 14).

Rocktäschel Tim, Grefenstette Edward, Hermann Karl Moritz, Kočiskỳ Tomáš, & Blunsom Phil. (2016). Reasoning about entailment with neural attention. *International Conference on Learning Representations (ICLR)* (cit. on p. 106).

Röder Michael, Usbeck Ricardo, Hellmann Sebastian, Gerber Daniel, & Both Andreas. (2014). $N^3$-a collection of datasets for named entity recognition and disambiguation in the nlp interchange format., In *Lrec*. (Cit. on p. 57).

Röder Michael, Usbeck Ricardo, & Ngonga Ngomo Axel-Cyrille. (2018). Gerbil–benchmarking named entity recognition and linking consistently. *Semantic Web*, *9*(5), 605–625 (cit. on pp. 48, 56, 100).

Rogers Anna, Kovaleva Olga, & Rumshisky Anna. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, *8*, 842–866. https://doi.org/10.1162/tacl_a_00349 (cit. on pp. 18, 19, 111)

Roller Stephen, Dinan Emily, Goyal Naman, Ju Da, Williamson Mary, Liu Yinhan, Xu Jing, Ott Myle, Smith Eric Michael, Boureau Y-Lan, & Weston Jason. (2021). Recipes for building an open-domain chatbot, In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: Main volume*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.24. (Cit. on p. 46)

Rongali Subendhu, Soldaini Luca, Monti Emilio, & Hamza Wael. (2020). Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing, In *Proceedings of the web conference 2020*, Association for Computing Machinery. (Cit. on p. 50).

Rossi Emanuele, Chamberlain Ben, Frasca Fabrizio, Eynard Davide, Monti Federico, & Bronstein Michael M. (2020). Temporal graph networks for deep learning on dynamic graphs. *CoRR*, *abs/2006.10637*arXiv (cit. on p. 11).

Rumelhart David E., & McClelland James L. (1987). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition: Foundations* (pp. 318–362). (Cit. on p. 13).

Sarawagi Sunita. (2008). *Information extraction*. Now Publishers Inc. (Cit. on p. 3).

Scarselli Franco, Gori Marco, Tsoi Ah Chung, Hagenbuchner Markus, & Monfardini Gabriele. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80. https://doi.org/10.1109/TNN.2008.2005605 (cit. on p. 10)

Schlichtkrull Michael, Kipf Thomas N., Bloem Peter, van den Berg Rianne, Titov Ivan, & Welling Max. (2018). Modeling relational data with graph convolutional networks (Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, & Mehwish Alam, Eds.). In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, & Mehwish Alam (Eds.), *The semantic web*, Cham, Springer International Publishing. (Cit. on pp. 11, 25, 27).

Schlichtkrull Michael Sejr, De Cao Nicola, & Titov Ivan. (2021). Interpreting graph neural networks for NLP with differentiable edge masking, In *9th international conference on learning representations, ICLR 2021, virtual event, austria, may 3-7, 2021*, OpenReview.net. (Cit. on pp. xvi, 105, 112, 135, 136).

Schulz Karl, Sixt Leon, Tombari Federico, & Landgraf Tim. (2020). Restricting the flow: Information bottlenecks for attribution, In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*, OpenReview.net. (Cit. on pp. 107, 108, 111, 117, 119, 120, 122, 124, 125, 134).

Sennrich Rico, Haddow Barry, & Birch Alexandra. (2016). Improving neural machine translation models with monolingual data, In *Proceedings of*

*the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Berlin, Germany, Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1009. (Cit. on p. 151)

Seo Min Joon, Kembhavi Aniruddha, Farhadi Ali, & Hajishirzi Hannaneh. (2017). Bidirectional attention flow for machine comprehension, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on pp. 22, 23, 25, 32, 41).

Serrano Sofia, & Smith Noah A. (2019). Is attention interpretable?, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1282. (Cit. on p. 106)

Sevegnani Karin, Howcroft David M., Konstas Ioannis, & Rieser Verena. (2021). OTTers: One-turn topic transitions for open-domain dialogue, In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.194. (Cit. on pp. 3, 46)

Shahbazi Hamed, Fern Xiaoli Z, Ghaeini Reza, Obeidat Rasha, & Tadepalli Prasad. (2019). Entity-aware ELMo: Learning Contextual Entity Representation for Entity Disambiguation. *arXiv preprint arXiv:1908.05762* (cit. on p. 60).

Shapley Lloyd S. (1953). A value for n-person games. Princeton, Princeton University Press. (Cit. on p. 111).

Shen Yelong, Huang Po-Sen, Gao Jianfeng, & Chen Weizhu. (2017). Reasonet: Learning to stop reading in machine comprehension, In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*. ACM. (Cit. on p. 22).

Shin Taylor, Razeghi Yasaman, Logan IV Robert L., Wallace Eric, & Singh Sameer. (2020). AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts, In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.346. (Cit. on p. 146)

Shrikumar Avanti, Greenside Peyton, & Kundaje Anshul. (2017). Learning important features through propagating activation differences (Doina Precup & Yee Whye Teh, Eds.). In Doina Precup & Yee Whye Teh (Eds.), *International conference on machine learning*, International Convention Centre, Sydney, Australia, PMLR. (Cit. on p. 106).

Sil Avirup, Kundu Gourab, Florian Radu, & Hamza Wael. (2018). Neural cross-lingual entity linking (Sheila A. McIlraith & Kilian Q. Weinberger, Eds.). In Sheila A. McIlraith & Kilian Q. Weinberger (Eds.), *Proceedings of the thirty-second AAAI conference on artificial intelligence, (aaai-18), the 30th*

*innovative applications of artificial intelligence (iaai-18), and the 8th AAAI symposium on educational advances in artificial intelligence (eaai-18), new orleans, louisiana, usa, february 2-7, 2018*, AAAI Press. (Cit. on pp. 73, 80, 83, 86).

Simonyan Karen, Vedaldi Andrea, & Zisserman Andrew. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. *Workshop at International Conference on Learning Representations* (cit. on p. 106).

Singh Chandan, Murdoch W. James, & Yu Bin. (2019). Hierarchical interpretations for neural network predictions, In *7th international conference on learning representations, ICLR 2019, new orleans, la, usa, may 6-9, 2019*, OpenReview.net. (Cit. on pp. 111, 112).

Sinitsin Anton, Plokhotnyuk Vsevolod, Pyrkin Dmitriy, Popov Sergei, & Babenko Artem. (2020). Editable neural networks, In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*, OpenReview.net. (Cit. on pp. 140, 145, 149, 162).

Sixt Leon, Granz Maximilian, & Landgraf Tim. (2020). When explanations lie: Why many modified BP attributions fail, In *Proceedings of the 37th international conference on machine learning, ICML 2020, 13-18 july 2020, virtual event*, PMLR. (Cit. on p. 106).

Slawski Bill. (2015). https://www.seobythesea.com/2015/09/disambiguate-entities-in-queries-and-pages/. (Cit. on p. 46)

Smellie William. (1768). *Encyclopædia britannica*. Britannica. (Cit. on p. 1).

Smith Shaden, Patwary Mostofa, Norick Brandon, LeGresley Patrick, Rajbhandari Samyam, Casper Jared, Liu Zhun, Prabhumoye Shrimai, Zerveas George, Korthikanti Vijay, Zheng Elton, Child Rewon, Aminabadi Reza Yazdani, Bernauer Julie, Song Xia, Shoeybi Mohammad, He Yuxiong, Houston Michael, Tiwary Saurabh, & Catanzaro Bryan. (2022). Using deepspeed and megatron to train megatron-turing NLG 530b, A large-scale generative language model. *CoRR*, *abs/2201.11990*arXiv (cit. on p. 19).

Socher Richard, Perelygin Alex, Wu Jean, Chuang Jason, Manning Christopher D., Ng Andrew, & Potts Christopher. (2013). Recursive deep models for semantic compositionality over a sentiment treebank, In *Proceedings of the 2013 conference on empirical methods in natural language processing*, Seattle, Washington, USA, Association for Computational Linguistics. (Cit. on pp. 118, 120).

Song Linfeng, Wang Zhiguo, Yu Mo, Zhang Yue, Florian Radu, & Gildea Daniel. (2018). Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *CoRR*, *abs/1809.02040*arXiv (cit. on pp. 25, 32, 33).

Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya, & Salakhutdinov Ruslan. (2014a). Dropout: A simple way to prevent neural networks

from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958 (cit. on pp. 52, 100, 150).

Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya, & Salakhutdinov Ruslan. (2014b). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958 (cit. on pp. 31, 54, 81).

Steinmetz Nadine, & Sack Harald. (2013). Semantic multimedia information retrieval based on contextual descriptions (Philipp Cimiano, Oscar Corcho, Valentina Presutti, Laura Hollink, & Sebastian Rudolph, Eds.). In Philipp Cimiano, Oscar Corcho, Valentina Presutti, Laura Hollink, & Sebastian Rudolph (Eds.), *The semantic web: Semantics and big data*, Berlin, Heidelberg, Springer Berlin Heidelberg. (Cit. on pp. 62, 98, 102).

Stephenson Cory, suchismita padhy, Ganesh Abhinav, Hui Yue, Tang Hanlin, & Chung SueYeon. (2021). On the geometry of generalization and memorization in deep neural networks, In *International conference on learning representations*. (Cit. on p. 161).

Sturmfels Pascal, Lundberg Scott, & Lee Su-In. (2020). Visualizing the impact of feature attribution baselines [https://distill.pub/2020/attribution-baselines]. *Distill*. https://doi.org/10.23915/distill.00022 (cit. on p. 134)

Sun Tony, Gaut Andrew, Tang Shirlyn, Huang Yuxin, ElSherief Mai, Zhao Jieyu, Mirza Diba, Belding Elizabeth, Chang Kai-Wei, & Wang William Yang. (2019a). Mitigating gender bias in natural language processing: Literature review, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1159. (Cit. on p. 106)

Sun Xiaofei, Yang Diyi, Li Xiaoya, Zhang Tianwei, Meng Yuxian, Qiu Han, Wang Guoyin, Hovy Eduard H., & Li Jiwei. (2021a). Interpreting deep learning models in natural language processing: A review. *CoRR*, *abs/2110.10470*arXiv (cit. on p. 138).

Sun Yu, Wang Shuohuan, Li Yukun, Feng Shikun, Chen Xuyi, Zhang Han, Tian Xin, Zhu Danxiang, Tian Hao, & Wu Hua. (2019b). Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (cit. on p. 146).

Sun Yu, Wang Shuohuan, Li Yu-Kun, Feng Shikun, Tian Hao, Wu Hua, & Wang Haifeng. (2020a). ERNIE 2.0: A continual pre-training framework for language understanding, In *The thirty-fourth AAAI conference on artificial intelligence, AAAI 2020, the thirty-second innovative applications of artificial intelligence conference, IAAI 2020, the tenth AAAI symposium on educational advances in artificial intelligence, EAAI 2020, new york, ny, usa, february 7-12, 2020*, AAAI Press. (Cit. on p. 19).

Sun Yu, Wang Shuohuan, Li Yukun, Feng Shikun, Tian Hao, Wu Hua, & Wang Haifeng. (2020b). Ernie 2.0: A continual pre-training framework for language understanding. *Proceedings of the AAAI Conference on Artificial*

*Intelligence*, *34*(05), 8968–8975. https://doi.org/10.1609/aaai.v34i05.6428 (cit. on p. 146)

Sun Yu, Wang Shuohuan, Feng Shikun, Ding Siyu, Pang Chao, Shang Junyuan, Liu Jiaxiang, Chen Xuyi, Zhao Yanbin, Lu Yuxiang, Liu Weixin, Wu Zhihua, Gong Weibao, Liang Jianzhong, Shang Zhizhou, Sun Peng, Liu Wei, Ouyang Xuan, Yu Dianhai, ... Wang Haifeng. (2021b). ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *CoRR*, *abs/2107.02137*arXiv (cit. on p. 19).

Sundararajan Mukund, Taly Ankur, & Yan Qiqi. (2017). Axiomatic attribution for deep networks (Doina Precup & Yee Whye Teh, Eds.). In Doina Precup & Yee Whye Teh (Eds.), *International conference on machine learning*, International Convention Centre, Sydney, Australia, PMLR. (Cit. on pp. 106–108, 117, 124, 125).

Sutskever Ilya, Martens James, & Hinton Geoffrey. (2011). Generating text with recurrent neural networks, In *Proceedings of the 28th international conference on international conference on machine learning*, Bellevue, Washington, USA, Omnipress. (Cit. on pp. 52, 100, 150).

Sutskever Ilya, Vinyals Oriol, & Le Quoc V. (2014). Sequence to sequence learning with neural networks (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger, Eds.). In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, Curran Associates, Inc. (Cit. on pp. 52, 53, 100, 150).

Szegedy Christian, Vanhoucke Vincent, Ioffe Sergey, Shlens Jon, & Wojna Zbigniew. (2016). Rethinking the inception architecture for computer vision, In *2016 ieee conference on computer vision and pattern recognition (cvpr)*. https://doi.org/10.1109/CVPR.2016.308. (Cit. on pp. 52, 100, 150)

Taghanaki Saeid Asgari, Havaei Mohammad, Berthier Tess, Dutil Francis, Di Jorio Lisa, Hamarneh Ghassan, & Bengio Yoshua. (2019). Infomask: Masked variational latent representation to localize chest disease (Dinggang Shen, Tianming Liu, Terry M. Peters, Lawrence H. Staib, Caroline Essert, Sean Zhou, Pew-Thian Yap, & Ali Khan, Eds.). In Dinggang Shen, Tianming Liu, Terry M. Peters, Lawrence H. Staib, Caroline Essert, Sean Zhou, Pew-Thian Yap, & Ali Khan (Eds.), *Medical image computing and computer assisted intervention – miccai 2019*, Cham, Springer International Publishing. (Cit. on p. 111).

Tang Zeyun, Shen Yongliang, Ma Xinyin, Xu Wei, Yu Jiale, & Lu Weiming. (2020). Multi-hop reading comprehension across documents with path-based graph convolutional network (Christian Bessiere, Ed.) [Main track]. In Christian Bessiere (Ed.), *Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI-20*, International Joint Conferences on Artificial Intelligence Organization. Main track. https://doi.org/10.24963/ijcai.2020/540. (Cit. on pp. 33, 41)

Tay Yi, Dehghani Mostafa, Bahri Dara, & Metzler Donald. (2022a). Efficient transformers: A survey [Just Accepted]. *ACM Comput. Surv.* https://doi.org/10.1145/3530811 (cit. on pp. 19, 42, 166)

Tay Yi, Tran Vinh Q., Dehghani Mostafa, Ni Jianmo, Bahri Dara, Mehta Harsh, Qin Zhen, Hui Kai, Zhao Zhe, Gupta Jai, Schuster Tal, Cohen William W., & Metzler Donald. (2022b). Transformer memory as a differentiable search index (Alice H. Oh, Alekh Agarwal, Danielle Belgrave, & Kyunghyun Cho, Eds.). In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, & Kyunghyun Cho (Eds.), *Advances in neural information processing systems*. (Cit. on pp. 67, 166).

Tenney Ian, Xia Patrick, Chen Berlin, Wang Alex, Poliak Adam, McCoy R Thomas, Kim Najoung, Durme Benjamin Van, Bowman Sam, Das Dipanjan, & Pavlick Ellie. (2019). What do you learn from context? probing for sentence structure in contextualized word representations, In *International conference on learning representations*. (Cit. on p. 111).

Thorne James, Vlachos Andreas, Christodoulopoulos Christos, & Mittal Arpit. (2018). FEVER: A large-scale dataset for fact extraction and VERification, In *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)*, New Orleans, Louisiana, Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1074. (Cit. on pp. 57, 150)

Tieleman Tijmen, & Hinton Geoffrey. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, *4*(2), 26–31 (cit. on pp. 117, 119, 149).

Tishby Naftali, & Zaslavsky Noga. (2015). Deep learning and the information bottleneck principle, In *2015 IEEE information theory workshop, ITW 2015, jerusalem, israel, april 26 - may 1, 2015*, IEEE. https://doi.org/10.1109/ITW.2015.7133169. (Cit. on p. 137)

Tsai Chen-Tse, & Roth Dan. (2016). Cross-lingual wikification using multilingual embeddings, In *Proceedings of the 2016 conference of the north American chapter of the association for computational linguistics: Human language technologies*, San Diego, California, Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1072. (Cit. on pp. 72, 73, 76, 80, 81, 83, 86)

Tu Ming, Wang Guangtao, Huang Jing, Tang Yun, He Xiaodong, & Zhou Bowen. (2019). Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1260. (Cit. on pp. 33, 41)

Upadhyay Shyam, Gupta Nitish, & Roth Dan. (2018). Joint multilingual supervision for cross-lingual entity linking, In *Proceedings of the 2018 conference on empirical methods in natural language processing*, Brussels, Belgium, Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1270. (Cit. on pp. 73, 80, 83, 86)

van Aken Betty, Winter Benjamin, Löser Alexander, & Gers Felix A. (2019). How Does BERT Answer Questions? A Layer-Wise Analysis of Transformer Representations, In *Proceedings of the 28th acm international conference on information and knowledge management*. (Cit. on p. 111).

van Hulst Johannes M., Hasibi Faegheh, Dercksen Koen, Balog Krisztian, & de Vries Arjen P. (2020). Rel: An entity linker standing on the shoulders of giants, In *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval*, Virtual Event, China, Association for Computing Machinery. https://doi.org/10.1145/3397271.3401416. (Cit. on pp. 62, 101, 102)

Vashishth Shikhar, Upadhyay Shyam, Tomar Gaurav Singh, & Faruqui Manaal. (2019). Attention interpretability across NLP tasks. *arXiv:1909.11218* (cit. on p. 106).

Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N., Kaiser Lukasz, & Polosukhin Illia. (2017). Attention is all you need (Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, & Roman Garnett, Eds.). In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, & Roman Garnett (Eds.), *Advances in neural information processing systems 30: Annual conference on neural information processing systems 2017, december 4-9, 2017, long beach, ca, USA*. (Cit. on pp. 13–16, 23, 25, 38, 41, 42, 96, 140).

Velickovic Petar, Cucurull Guillem, Casanova Arantxa, Romero Adriana, Liò Pietro, & Bengio Yoshua. (2018). Graph attention networks, In *6th international conference on learning representations, ICLR 2018, vancouver, bc, canada, april 30 - may 3, 2018, conference track proceedings*, OpenReview.net. (Cit. on p. 41).

Verga Pat, Sun Haitian, Soares Livio Baldini, & Cohen William W. (2020). Facts as experts: Adaptable and interpretable neural memory over symbolic knowledge. *CoRR*, *abs/2007.00849*arXiv (cit. on pp. 3, 19, 146).

Vig Jesse, Gehrmann Sebastian, Belinkov Yonatan, Qian Sharon, Nevo Daniel, Singer Yaron, & Shieber Stuart. (2020). Investigating gender bias in language models using causal mediation analysis (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin, Eds.). In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems*, Curran Associates, Inc. (Cit. on pp. 141, 146, 161).

Vincent Pascal, Larochelle Hugo, Lajoie Isabelle, Bengio Yoshua, & Manzagol Pierre-Antoine. (2010). Stacked denoising autoencoders: Learning useful

representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, *11*, 3371–3408. https://doi.org/10.5555/1756006.1953039 (cit. on p. 18)

Vinyals Oriol, Fortunato Meire, & Jaitly Navdeep. (2015). Pointer networks (Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, & Roman Garnett, Eds.). In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, & Roman Garnett (Eds.), *Advances in neural information processing systems 28: Annual conference on neural information processing systems 2015, december 7-12, 2015, montreal, quebec, canada*. (Cit. on p. 41).

Voita Elena, Talbot David, Moiseev Fedor, Sennrich Rico, & Titov Ivan. (2019a). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1580. (Cit. on p. 122)

Voita Elena, Sennrich Rico, & Titov Ivan. (2019b). The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives, In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1448. (Cit. on p. 129)

Voita Elena, Sennrich Rico, & Titov Ivan. (2019c). The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives, In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1448. (Cit. on p. 140)

Vrandečić Denny. (2012). Wikidata: A new platform for collaborative data collection, In *Proceedings of the 21st international conference on world wide web*, Lyon, France, Association for Computing Machinery. https://doi.org/10.1145/2187980.2188242. (Cit. on pp. 24, 72, 76)

Weissenborn Dirk, Wiese Georg, & Seiffe Laura. (2017). Making neural QA as simple as possible but not simpler, In *Proceedings of the 21st conference on computational natural language learning (CoNLL 2017)*, Vancouver, Canada, Association for Computational Linguistics. https://doi.org/10.18653/v1/K17-1028. (Cit. on pp. 22, 25, 32)

Welbl Johannes, Stenetorp Pontus, & Riedel Sebastian. (2018). Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, *6*, 287–302. https:

//doi.org/10.1162/tacl_a_00021 (cit. on pp. 21, 22, 24, 25, 29, 31–34, 36, 40, 42)

Wen Tsung-Hsien, Vandyke David, Mrkšić Nikola, Gašić Milica, Rojas-Barahona Lina M., Su Pei-Hao, Ultes Stefan, & Young Steve. (2017). A network-based end-to-end trainable task-oriented dialogue system, In *Proceedings of the 15th conference of the European chapter of the association for computational linguistics: Volume 1, long papers*, Valencia, Spain, Association for Computational Linguistics. (Cit. on p. 3).

Wenzek Guillaume, Lachaux Marie-Anne, Conneau Alexis, Chaudhary Vishrav, Guzmán Francisco, Joulin Armand, & Grave Edouard. (2020). CCNet: Extracting high quality monolingual datasets from web crawl data, In *Proceedings of the 12th language resources and evaluation conference*, Marseille, France, European Language Resources Association. (Cit. on pp. 77, 79).

Wieting John, & Gimpel Kevin. (2018). ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations, In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Melbourne, Australia, Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1042. (Cit. on p. 151)

Williams Jason D., Asadi Kavosh, & Zweig Geoffrey. (2017). Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning, In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, Vancouver, Canada, Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1062. (Cit. on p. 3)

Williams Ronald J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3), 229–256. https://doi.org/10.1007/BF00992696 (cit. on pp. 109, 114, 120)

Winograd Terry. (1971). *Procedures as a representation for data in a computer program for understanding natural language* (tech. rep.). Massachusetts Institute of Technology Laboratory for Computer Science. (Cit. on p. 12).

Wolf Thomas, Debut Lysandre, Sanh Victor, Chaumond Julien, Delangue Clement, Moi Anthony, Cistac Pierric, Rault Tim, Louf Remi, Funtowicz Morgan, Davison Joe, Shleifer Sam, von Platen Patrick, Ma Clara, Jernite Yacine, Plu Julien, Xu Canwen, Le Scao Teven, Gugger Sylvain, ... Rush Alexander. (2020). Transformers: State-of-the-art natural language processing, In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-demos.6. (Cit. on pp. 118, 119, 151)

Wu Ledell, Petroni Fabio, Josifoski Martin, Riedel Sebastian, & Zettlemoyer Luke. (2020). Scalable zero-shot entity linking with dense entity retrieval, In *Proceedings of the 2020 conference on empirical methods in natural language*

*processing (emnlp)*, Online, Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.519. (Cit. on pp. 3, 46, 50, 51, 56, 57, 59, 61, 63, 70, 73, 76)

Wu Yonghui, Schuster Mike, Chen Zhifeng, Le Quoc V., Norouzi Mohammad, Macherey Wolfgang, Krikun Maxim, Cao Yuan, Gao Qin, Macherey Klaus, Klingner Jeff, Shah Apurva, Johnson Melvin, Liu Xiaobing, Kaiser Lukasz, Gouws Stephan, Kato Yoshikiyo, Kudo Taku, Kazawa Hideto, ... Dean Jeffrey. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, *abs/1609.08144* arXiv (cit. on p. 52).

Xiong Caiming, Zhong Victor, & Socher Richard. (2017). Dynamic coattention networks for question answering, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on pp. 22, 41).

Xiong Wenhan, Li Xiang Lorraine, Iyer Srini, Du Jingfei, Lewis Patrick S. H., Wang William Yang, Mehdad Yashar, Yih Scott, Riedel Sebastian, Kiela Douwe, & Oguz Barlas. (2021). Answering complex open-domain questions with multi-hop dense retrieval, In *9th international conference on learning representations, ICLR 2021, virtual event, austria, may 3-7, 2021*, OpenReview.net. (Cit. on p. 42).

Yamada Ikuya, Shindo Hiroyuki, Takeda Hideaki, & Takefuji Yoshiyasu. (2016). Joint learning of the embedding of words and entities for named entity disambiguation, In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, Berlin, Germany, Association for Computational Linguistics. https://doi.org/10.18653/v1/K16-1025. (Cit. on pp. 56, 60, 80, 100)

Yang Xiyuan, Gu Xiaotao, Lin Sheng, Tang Siliang, Zhuang Yueting, Wu Fei, Chen Zhigang, Hu Guoping, & Ren Xiang. (2019). Learning dynamic context augmentation for global entity linking, In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1026. (Cit. on p. 60)

Yang Yi, Irsoy Ozan, & Rahman Kazi Shefaet. (2018a). Collective entity disambiguation with structured gradient tree boosting, In *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)*, New Orleans, Louisiana, Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1071. (Cit. on pp. 46, 60)

Yang Zhilin, Qi Peng, Zhang Saizheng, Bengio Yoshua, Cohen William, Salakhutdinov Ruslan, & Manning Christopher D. (2018b). HotpotQA: A dataset for diverse, explainable multi-hop question answering, In *Proceedings of the 2018 conference on empirical methods in natural language process-*

*ing*, Brussels, Belgium, Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1259. (Cit. on pp. 22, 40, 57)

Yih Wen-tau, Toutanova Kristina, Platt John C., & Meek Christopher. (2011). Learning discriminative projections for text similarity measures, In *Proceedings of the fifteenth conference on computational natural language learning*, Portland, Oregon, USA, Association for Computational Linguistics. (Cit. on p. 50).

Ying Zhitao, Bourgeois Dylan, You Jiaxuan, Zitnik Marinka, & Leskovec Jure. (2019). GNNExplainer: Generating Explanations for Graph Neural Networks (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett, Eds.). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32*. Curran Associates, Inc. (Cit. on p. 111).

Yu Yue, Chen Jie, Gao Tian, & Yu Mo. (2019). DAG-GNN: DAG structure learning with graph neural networks (Kamalika Chaudhuri & Ruslan Salakhutdinov, Eds.). In Kamalika Chaudhuri & Ruslan Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning*, PMLR. (Cit. on p. 11).

Zaheer Manzil, Guruganesh Guru, Dubey Kumar Avinava, Ainslie Joshua, Alberti Chris, Ontanon Santiago, Pham Philip, Ravula Anirudh, Wang Qifan, Yang Li, & Ahmed Amr. (2020). Big bird: Transformers for longer sequences (H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, & H. Lin, Eds.). In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems*, Curran Associates, Inc. (Cit. on pp. 33, 42).

Zhang Michael, Lucas James, Ba Jimmy, & Hinton Geoffrey E. (2019a). Lookahead optimizer: K steps forward, 1 step back (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett, Eds.). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32*. Curran Associates, Inc. (Cit. on pp. 117, 119).

Zhang Susan, Roller Stephen, Goyal Naman, Artetxe Mikel, Chen Moya, Chen Shuohui, Dewan Christopher, Diab Mona, Li Xian, Lin Xi Victoria, Mihaylov Todor, Ott Myle, Shleifer Sam, Shuster Kurt, Simig Daniel, Koura Punit Singh, Sridhar Anjali, Wang Tianlu, & Zettlemoyer Luke. (2022a). OPT: open pre-trained transformer language models. *CoRR*, *abs/2205.01068* arXiv. https://doi.org/10.48550/arXiv.2205.01068 (cit. on p. 19)

Zhang Wenzheng, Hua Wenyue, & Stratos Karl. (2022b). EntQA: Entity linking as question answering, In *International conference on learning representations*. (Cit. on pp. 62, 102).

Zhang Yuhao, Qi Peng, & Manning Christopher D. (2018a). Graph convolution over pruned dependency trees improves relation extraction, In *Proceedings*

*of the 2018 conference on empirical methods in natural language processing*, Brussels, Belgium, Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1244. (Cit. on p. 25)

Zhang Yuyu, Dai Hanjun, Kozareva Zornitsa, Smola Alexander J., & Song Le. (2018b). Variational reasoning for question answering with knowledge graph, In *Proceedings of the thirty-second aaai conference on artificial intelligence and thirtieth innovative applications of artificial intelligence conference and eighth aaai symposium on educational advances in artificial intelligence*, New Orleans, Louisiana, USA, AAAI Press. (Cit. on p. 25).

Zhang Zhengyan, Han Xu, Liu Zhiyuan, Jiang Xin, Sun Maosong, & Liu Qun. (2019b). ERNIE: Enhanced language representation with informative entities, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1139. (Cit. on pp. 19, 146)

Zheng Jin G, Howsmon Daniel, Zhang Boliang, Hahn Juergen, McGuinness Deborah, Hendler James, & Ji Heng. (2015). Entity linking for biomedical literature. *BMC medical informatics and decision making*, *15*(1), 1–9 (cit. on p. 3).

Zhou Shuyan, Rijhwani Shruti, & Neubig Graham. (2019). Towards zero-resource cross-lingual entity linking, In *Proceedings of the 2nd workshop on deep learning approaches for low-resource nlp (deeplo 2019)*, Hong Kong, China, Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-6127. (Cit. on pp. 73, 80, 83, 86)

Zhu Chen, Rawat Ankit Singh, Zaheer Manzil, Bhojanapalli Srinadh, Li Daliang, Yu Felix, & Kumar Sanjiv. (2020). Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363* (cit. on pp. 140, 145–147, 149, 153, 154, 157).

Zhuang Yimeng, & Wang Huadong. (2019). Token-level dynamic self-attention network for multi-passage reading comprehension, In *Proceedings of the 57th annual meeting of the association for computational linguistics*, Florence, Italy, Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1218. (Cit. on pp. 33, 42)

Zintgraf Luisa M., Cohen Taco S., Adel Tameem, & Welling Max. (2017). Visualizing deep neural network decisions: Prediction difference analysis, In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, OpenReview.net. (Cit. on p. 106).

# Samenvatting

Entiteiten staan centraal in hoe we kennis representeren en aggregeren. In encyclopedieën zoals Wikipedia is informatie bijvoorbeeld gegroepeerd op basis van entiteiten (één entiteit per artikel). Hoewel hedendaagse natuurlijke taalverwerkingstechnologie (NLP) bijzonder succesvol is geworden in het beantwoorden van vragen, ondervinden moderne neurale netwerken nog steeds moeilijkheden met het integreren van gestructureerde informatie over entiteiten in hun beslissingsproces. In dit proefschrift, "Entiteit-gecentreerde neurale modellen voor natuurlijke taalverwerking", onderzoeken we hoe we effectievere neurale netwerken kunnen bouwen die informatie over entiteiten benutten om natuurlijke taal te begrijpen. We richten ons voornamelijk op drie onderzoeksvragen:

**Hoe kunnen we entiteiten gebruiken om taken met betrekking tot natuurlijke taalverwerking effectiever aan te pakken?** We introduceren een neuraal netwerk dat redeneringen integreert die gebaseerd zijn op informatiespreiding binnen een enkel document en over meerdere documenten (Hoofdstuk 3). We kaderen dit als een inferentieprobleem op een graaf. Vermeldingen van entiteiten zijn knopen (nodes) in deze graaf, terwijl de zijden (edges) relaties tussen verschillende vermeldingen representeren (bijv. coreferenties binnen en tussen documenten). Convolutionele neurale netwerken voor grafen (GCN's) worden op deze grafen toegepast en getraind om redeneringen over meerdere stappen uit te voeren. Onze Entiteit-GCN-methode is schaalbaar en compact en behaalde, ten tijde van schrijven (d.w.z. 2018), state-of-the-art resultaten op WikiHop, een populaire dataset voor het automatisch beantwoorden van vragen met meerdere documenten.

**Hoe kunnen we grote, vooraf getrainde taalmodellen gebruiken om entiteiten in de tekst te identificeren en te disambigueren?** Het eerste systeem dat wij voorstellen, vraagt entiteiten op door hun unieke namen te genereren, van links naar rechts, token voor token op een autoregressieve manier

(Hoofdstuk 4). Ons model vermindert beperkingen van de gevestigde "two-tower dot-product" -modellen die mogelijk gedetailleerde interacties tussen tekst en entiteiten in een kennisbank missen. Bovendien verminderen we het geheugengebruik van huidige modellen aanzienlijk (tot 15 keer). Dit komt doordat de parameters van onze encoder-decoder architectuur schalen met de grootte van het vocabulaire, in plaats van met het aantal entiteiten. We breiden onze aanpak ook uit naar een grote, meertalige setting met meer dan 100 talen (Hoofdstuk 5). In deze setting matchen we met entiteitsnamen van zo veel mogelijk talen, waardoor we connecties tussen de invoerbrontaal en de doelnaam kunnen benutten. Tot slot introduceren we een zeer efficiënte methode die autoregressieve linking paralleliseert over alle potentiële vermeldingen. Deze methodegebruikt een ondiepe en efficiënte decoder, wat het model tot 70 keer sneller maakt, zonder prestatieverlies (Hoofdstuk 6).

**Hoe kunnen we de interne kennis van een model over entiteiten interpreteren en beheersen?** We presenteren een nieuwe techniek voor post-hoc interpretatie in Hoofdstuk 7. Deze techniek isbedoeld om te onderzoeken hoe beslissingen tot stand komen in verschillende lagen van neurale netwerken. Ons systeem leert subsets van vectoren te maskeren met behoud van differentieerbaarheid. Dit stelt ons in staat omattributie-heatmaps te visualiseren en te analyseren hoe beslissingen worden gevormd in de verschillende lagen van het neurale netwerk. We gebruiken dit systeem om BERT-modellen te bestuderen op sentiment classificatie en het automatisch beantwoorden van vragen. We laten bovendien zien dat deze techniek toepasbaar is op het convolutionele neurale netwerk voor grafen„ gepresenteerd in Hoofdstuk 3. Ten slotte introduceren we een methode die kan worden gebruikt om deze feitelijke kennis over entiteiten te bewerken. Dit maakt het mogelijk om 'bugs' of onverwachte voorspellingen te herstellen zonder dat dure "hertraining" of "finetuning" nodig is (Hoofdstuk 8).

# Abstract

Entities are at the center of how we represent and aggregate knowledge. For instance, in Encyclopedias such as Wikipedia information is grouped according to entities (e.g., one entity per article). However, although contemporary NLP technology has become remarkably successful in machine-driven question answering, modern neural network models struggle to incorporate structured information about entities into their decision process. In this thesis, "Entity Centric Neural Models for Natural Language Processing", we investigate how to build effective neural network models exploiting entity information for natural language understanding. We mainly consider three research questions:

**How can we exploit entities to tackle Natural Language Understanding tasks?** We introduce a neural model that integrates reasons relying on information spread within and across multiple documents (chapter 3). We frame it as an inference problem on a graph. Mentions of entities are nodes of this graph, while edges encode relations between different mentions (e.g., within- and cross-document co-reference). Graph convolutional networks (GCNs) are applied to these graphs and trained to perform multi-step reasoning. Our Entity-GCN method is scalable and compact, and it achieved state-of-the-art results at the time of writing (i.e., 2018) on WikiHop, a popular multi-document question-answering dataset.

**How can we exploit large pre-trained language models to identify and disambiguate entities in the text?** We propose the first system that retrieves entities by generating their unique names, left to right, token-by-token in an autoregressive fashion (chapter 4). Our model mitigates the limitations of well-established two-tower dot-product-based models that potentially miss fine-grained interactions between text and entities in a Knowledge Base. Additionally, we significantly reduced the memory footprint of current systems (up to 15 times) because the parameters of our encoder-decoder architecture scale with vocabulary

size, not the entity count. We also extend our approach to a large multilingual setting with more than 100 languages (chapter 5). In this setting, we match against entity names of as many languages as possible, which allows exploiting language connections between source input and target name. Finally, we also propose a very efficient approach that parallelizes autoregressive linking across all potential mentions and relies on a shallow and efficient decoder which allows a $>$70 faster model with no performance drop (chapter 6).

**How can we interpret and control a model's internal knowledge about entities?** We introduce a novel post-hoc interpretation technique for inspecting how decisions emerge across layers in neural models in chapter 7. Our system learns to mask-out subsets of vectors while maintaining differentiability. This lets us not only plot attribution heatmaps but also analyze how decisions are formed across network layers. We use this system to study BERT models on sentiment classification and question answering additionally showing that this technique can be applied to the graph-based model presented in chapter 3. Finally, we also propose a method that can be used to edit this factual knowledge about entities and, thus, fix 'bugs' or unexpected predictions without the need for expensive re-training or fine-tuning (chapter 8).

# Titles in the ILLC Dissertation Series

ILLC DS-2018-13: **Jeroen Zuiddam**
*Algebraic complexity, asymptotic spectra and entanglement polytopes*

ILLC DS-2019-01: **Carlos Vaquero**
*What Makes A Performer Unique? Idiosyncrasies and commonalities in expressive music performance*

ILLC DS-2019-02: **Jort Bergfeld**
*Quantum logics for expressing and proving the correctness of quantum programs*

ILLC DS-2019-03: **András Gilyén**
*Quantum Singular Value Transformation & Its Algorithmic Applications*

ILLC DS-2019-04: **Lorenzo Galeotti**
*The theory of the generalised real numbers and other topics in logic*

ILLC DS-2019-05: **Nadine Theiler**
*Taking a unified perspective: Resolutions and highlighting in the semantics of attitudes and particles*

ILLC DS-2019-06: **Peter T.S. van der Gulik**
*Considerations in Evolutionary Biochemistry*

ILLC DS-2019-07: **Frederik Möllerström Lauridsen**
*Cuts and Completions: Algebraic aspects of structural proof theory*

ILLC DS-2020-01: **Mostafa Dehghani**
*Learning with Imperfect Supervision for Language Understanding*

ILLC DS-2020-02: **Koen Groenland**
*Quantum protocols for few-qubit devices*

ILLC DS-2020-03: **Jouke Witteveen**
*Parameterized Analysis of Complexity*

ILLC DS-2020-04: **Joran van Apeldoorn**
*A Quantum View on Convex Optimization*

ILLC DS-2020-05: **Tom Bannink**
*Quantum and stochastic processes*

ILLC DS-2020-06: **Dieuwke Hupkes**
*Hierarchy and interpretability in neural models of language processing*

ILLC DS-2020-07: **Ana Lucia Vargas Sandoval**
*On the Path to the Truth: Logical & Computational Aspects of Learning*

ILLC DS-2020-08: **Philip Schulz**
*Latent Variable Models for Machine Translation and How to Learn Them*

ILLC DS-2020-09: **Jasmijn Bastings**
*A Tale of Two Sequences: Interpretable and Linguistically-Informed Deep Learning for Natural Language Processing*

ILLC DS-2020-10: **Arnold Kochari**
*Perceiving and communicating magnitudes: Behavioral and electrophysiological studies*

ILLC DS-2020-11: **Marco Del Tredici**
*Linguistic Variation in Online Communities: A Computational Perspective*

ILLC DS-2020-12: **Bastiaan van der Weij**
*Experienced listeners: Modeling the influence of long-term musical exposure on rhythm perception*

ILLC DS-2020-13: **Thom van Gessel**
*Questions in Context*

ILLC DS-2020-14: **Gianluca Grilletti**
*Questions & Quantification: A study of first order inquisitive logic*

ILLC DS-2020-15: **Tom Schoonen**
*Tales of Similarity and Imagination. A modest epistemology of possibility*

ILLC DS-2020-16: **Ilaria Canavotto**
*Where Responsibility Takes You: Logics of Agency, Counterfactuals and Norms*

ILLC DS-2020-17: **Francesca Zaffora Blando**
*Patterns and Probabilities: A Study in Algorithmic Randomness and Computable Learning*

ILLC DS-2021-01: **Yfke Dulek**
*Delegated and Distributed Quantum Computation*

ILLC DS-2021-02: **Elbert J. Booij**
*The Things Before Us: On What it Is to Be an Object*

ILLC DS-2021-03: **Seyyed Hadi Hashemi**
*Modeling Users Interacting with Smart Devices*

ILLC DS-2021-04: **Sophie Arnoult**
*Adjunction in Hierarchical Phrase-Based Translation*

ILLC DS-2021-05: **Cian Guilfoyle Chartier**
*A Pragmatic Defense of Logical Pluralism*

ILLC DS-2021-06: **Zoi Terzopoulou**
*Collective Decisions with Incomplete Individual Opinions*

ILLC DS-2021-07: **Anthia Solaki**
*Logical Models for Bounded Reasoners*

ILLC DS-2021-08: **Michael Sejr Schlichtkrull**
*Incorporating Structure into Neural Models for Language Processing*

ILLC DS-2021-09: **Taichi Uemura**
*Abstract and Concrete Type Theories*

ILLC DS-2021-10: **Levin Hornischer**
*Dynamical Systems via Domains: Toward a Unified Foundation of Symbolic and Non-symbolic Computation*

ILLC DS-2021-11: **Sirin Botan**
*Strategyproof Social Choice for Restricted Domains*

ILLC DS-2021-12: **Michael Cohen**
*Dynamic Introspection*

ILLC DS-2021-13: **Dazhu Li**
*Formal Threads in the Social Fabric: Studies in the Logical Dynamics of Multi-Agent Interaction*

ILLC DS-2022-01: **Anna Bellomo**
*Sums, Numbers and Infinity: Collections in Bolzano's Mathematics and Philosophy*

ILLC DS-2022-02: **Jan Czajkowski**
*Post-Quantum Security of Hash Functions*

ILLC DS-2022-03: **Sonia Ramotowska**
*Quantifying quantifier representations: Experimental studies, computational modeling, and individual differences*

ILLC DS-2022-04: **Ruben Brokkelkamp**
*How Close Does It Get?: From Near-Optimal Network Algorithms to Suboptimal Equilibrium Outcomes*

ILLC DS-2022-05: **Lwenn Bussière-Carae**
*No means No! Speech Acts in Conflict*

ILLC DS-2023-01: **Subhasree Patro**
*Quantum Fine-Grained Complexity*

ILLC DS-2023-02: **Arjan Cornelissen**
*Quantum multivariate estimation and span program algorithms*

ILLC DS-2023-03: **Robert Paßmann**
*Logical Structure of Constructive Set Theories*

ILLC DS-2023-04: **Samira Abnar**
*Inductive Biases for Learning Natural Language*

ILLC DS-2023-05: **Dean McHugh**
*Causation and Modality: Models and Meanings*

ILLC DS-2023-06: **Jialiang Yan**
*Monotonicity in Intensional Contexts: Weakening and: Pragmatic Effects under Modals and Attitudes*

ILLC DS-2023-07: **Yiyan Wang**
*Collective Agency: From Philosophical and Logical Perspectives*

ILLC DS-2023-08: **Lei Li**
*Games, Boards and Play: A Logical Perspective*

ILLC DS-2023-09: **Simon Rey**
*Variations on Participatory Budgeting*

ILLC DS-2023-10: **Mario Giulianelli**
*Neural Models of Language Use: Studies of Language Comprehension and Production in Context*

ILLC DS-2023-11: **Guillermo Menéndez Turata**
*Cyclic Proof Systems for Modal Fixpoint Logics*

ILLC DS-2023-12: **Ned J.H. Wontner**
*Views From a Peak: Generalisations and Descriptive Set Theory*

ILLC DS-2024-01: **Jan Rooduijn**
*Fragments and Frame Classes: Towards a Uniform Proof Theory for Modal Fixed Point Logics*

ILLC DS-2024-02: **Bas Cornelissen**
*Measuring musics: Notes on modes, motifs, and melodies*