Modelling Syntactic and Semantic Tasks with Linguistically Enriched Recursive Neural Networks

MSc Thesis *(Afstudeerscriptie)*

written by

Jonathan Mallinson
(born December 29, 1989 in Ipswich, United Kingdom)

under the supervision of **Dr Willem Zuidema**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam*.

Date of the public defense:
May 13, 2015

Members of the Thesis Committee:
Dr Jakub Szymanik
Dr Ivan Titov
Dr Henk Zeevat
Dr Willem Zuidema

Institute for Logic, Language and Computation

Abstract

In this thesis a compositional distributional semantic approach, the Recursive Neural Network, is used to syntactically-semantically compose non-symbolic representations of words. Unlike previous Recursive Neural Network models which use either no linguistic enrichment or significant symbolic syntactic enrichment, I propose minimal linguistic enrichments which are both semantic and syntactic. I achieve this by enriching the Recursive Neural Networks' models with core syntactic/semantic linguistic types: head, argument and adjunct. This approach brings together formal linguistics and computational linguistics, as such I give a broad account of these theories. The syntactic understanding of the model is tested by a parsing task and the semantic understanding is tested by a paraphrase detection task. The results of these tasks not only show the benefits of linguistic enrichment but also raise further questions of study.

Acknowledgements

I would first like to thank my supervisor Jelle Zuidema for not only suggesting the topic of my thesis but also providing support throughout my thesis period. I thank the members of my thesis committee: Ivan Titvo, Henk Zeevat and Jakub Szymanik, who took the time to read my thesis and provide feedback. Furthermore, I would also like to thank Tanja Kassenaar and Fenneke Kortenbach for arranging my defense at short notice.

Cuong Hoang, Jo Daiber, Ehsan Khoddammohammadi, Phong Le and Ivan Titov provided invaluable technical help and advice, for which I would like to thank them. I would also give special thanks to Nikolas Nisidis, Kuan Ko-Hung and Yuning Feng who provided feedback on my thesis presentation thus subjecting themselves to my presentation twice.

While writing my thesis, my time in Amsterdam was made all the more enjoyable by my flatmates: Eefje Schut, Marijn Aerts, Yuning Feng, Ciyang Qing, Jingting Wu, and my friends from the ILLC and CWI, creating a wonderful atmosphere in both my life and studies.

A final thank you goes to my supportive family: Keren, Steve and Sarah.

Contents

Chapter 1

Introduction

1.1 Motivation

The study of natural language is a diverse but fruitful field of research which spans multiple disciplines. However, within this thesis I will focus my interest on the works of both formal linguistics and computational linguistics. Computational linguistics generally takes a task-based approach, building a model which best quantitatively fulfils a particular task. Formal linguistics on the other hand is phenomenon-driven and, as such, seeks to explain a particular phenomenon. Formal linguistics is interested in developing a theory that captures all the details and edge cases of the phenomenon. Computational linguistics is reliant on machine learning techniques to capture the phenomenon and generally finds it difficult to capture edge cases; instead the focus is on capturing the common cases (*fat head*) of the problem. My approach tries to find a middle ground between these two groups, where the framework of the model fits within linguistic theories (linguistically justified). However, the model learns on a dataset and is evaluated on standard computational linguistic tasks - trying to solve the *fat head* of the problem.

Due to the complexity of natural language it is often decomposed into several distinct modules: lexical, morphological, syntactic and semantic/pragmatic, as seen in figure 1.1 (Pinker, 1999). Within this thesis I intend to produce a computational model of syntax, semantics and their interface. Not only are syntax and semantics core to language, but computational models of syntax and semantics have been used in many different NLP approaches, including: machine translation (Yamada and Knight, 2001), semantic role labelling (Surdeanu and Turmo, 2005), question answering (Lin and Pantel, 2001) and sentiment analysis (Socher et al., 2012).

Recently, approaches to modelling language have been split into two camps: symbolic and non-symbolic. Within formal linguistics the majority of frameworks and models are

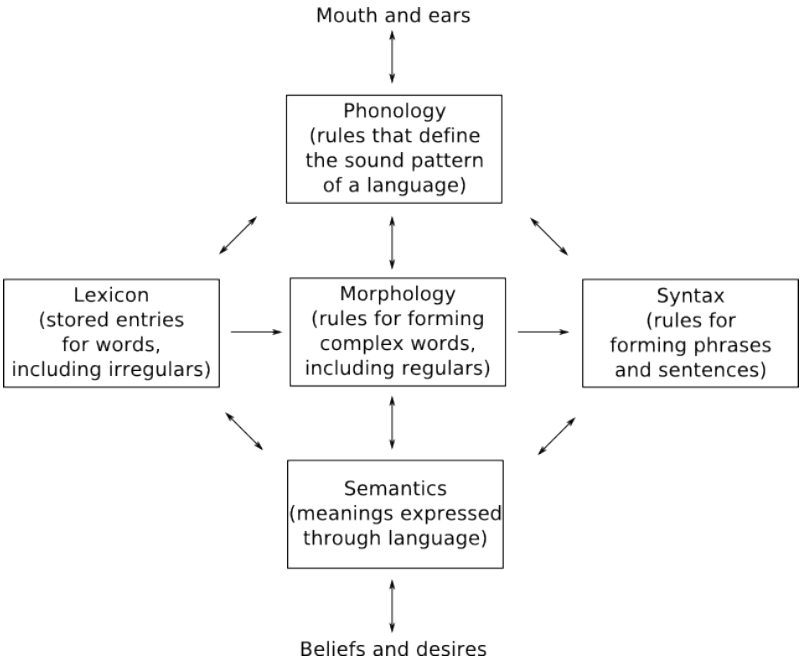


FIGURE 1.1: Models of language, as provided by Pinker (1999)

symbolic, partially due to the difficulty of working with non-symbolic models. Computational linguistic approaches, however are more evenly split between symbolic and non-symbolic approaches. I take a non-symbolic approach, due to the inherit flexibility it offers. To do so I draw from the connectionist paradigm and use a deep learning neural network as the basis of my model. Not only has deep learning seen an increase in power and attention in recent years but also connectionism offers a step towards neural-plausible models of language.

To encourage a linguistically-justified approach the model will fulfil a series of requirements. First, the approach will be a joint model of syntax and semantics. A joint model avoids the problem found within stochastic pipeline models (Zeevat, 2014), as follows. If a pipeline has enough modules, then even if each individual module gives a high likelihood, the likelihood of the entire interpretation is low. Consider a pipeline of the five modules of language where each module gives a 0.8 likelihood to the highest scoring interpretation of an utterance, we then obtain $0.8^5 = 0.33$. This low confidence of understanding the utterance does not match up well against personal experience of language. Also, a joint model provides an explicit interface between syntax and semantics allowing information from one to assist the other, enriching the model. Secondly, the model will adhere to the principle of composition where the semantic meaning of the utterance is taken from the meaning and interactions of the individual words as determined by a lexicon and syntactic structure.

These requirements motivate a compositional distributional semantic approach, where

an approach called the Recursive Neural Network is used to syntactically-semantically compose non-symbolic representations of words. Recursive Neural Networks are a general case of the popular machine learning framework: the Recurrent Neural Network. However, unlike Recurrent Neural Networks which model temporal aspects, Recursive Neural Networks can model structure; in this case they will model the semantic-syntactic structure of an utterance (Baldi and Pollastri, 2003). Although Recursive Neural Network have previously been used to model syntax and semantics, I try to fill a hole left in the literature. Previously, Recursive Neural Networks models have used either no linguistic enrichment (Socher et al., 2010) or significant syntactic enrichment (Socher et al., 2013) which brought the model back towards a symbolic approach. I instead propose linguistic enrichment which is both semantic and syntactic but also less complex than previous approaches. I achieve this by enriching the Recursive Neural Networks with core syntactic/semantic linguistic types.

1.2 Thesis outline

The thesis is split into six additional chapters. I start with an introduction to symbolic approaches to syntax and semantics. I give formal linguistic approaches to syntax, then computational linguistic realisations of these approaches. Next, I address semantics with a focus on Montague grammar from formal linguistics, which is then contrasted against semantic role labelling found within computational semantics. Chapter three focuses on non-symbolic distributional semantic approaches to lexical semantics before moving on to compositional distributional semantics. I start chapter three by developing the motivation for distributional semantics before giving a simple example of a possible distributional semantic approach. I next give possible extensions to this model including how distributional semantics can be incorporated into multi-word settings forming compositional distributional semantics. Within the fourth chapter I introduce the Recursive Neural Network. To do so I first outline connectionism and multiple types of neural networks. I then discuss the Recursive Neural Network and how it is used within my approach. The fifth chapter discusses my extension of the Recursive Neural Network. I provide both the motivation for my approach and the specifics of the extension. Chapter six contains the implementational details of my approach and the results achieved by it. I include information regarding both the syntactic parsing task and the semantic paraphrase tasks taken, as well as a qualitative analysis of my models. Finally I conclude with the achievements of this thesis as well as providing an outlook into further developments.

Chapter 2

Symbolic Natural Language

2.1 Introduction

Within this chapter I will outline previous work on syntax and semantics, partially due to my reliance on these works and partially to later contrast my approach against them. I will first outline linguistic syntactic theories, both within the generative and Tesniére tradition. I then give an account of a computational implementation of the generative approach, the probabilistic context-free grammar. I next discuss semantics, first introducing the iconic Montagovian tradition which heavily influences my model. This is then compared to the popular semantic role labelling approach found within computational linguistics.

2.2 Syntax

Syntax is one of the most studied modules of language and as such has a wide range of theoretical explanations. Within my thesis I will follow the generative tradition, where syntax defines an internal treelike syntactic structure composed of phrases for an utterance (see figure 2.1) (Chomsky, 1988). Phrases group together words and other phrases which then behave as a single unit. These units or *constituents* can be moved to different syntactic positions without being broken apart. The task of syntactic understanding is to produce a formal grammar of the language, which defines the syntactic structure for all and only the syntactically valid sentences of the language.

Although I follow a generative approach, I later take inspiration from Tesnière grammar. Within Tesnière grammar the syntactic structure is determined by a series of *dependency relations* (Nivre, 2005). The verb of the utterance is the structural centre

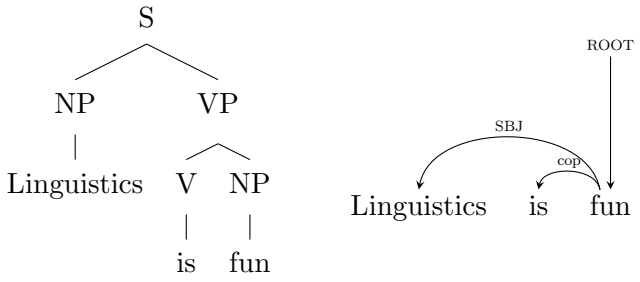


FIGURE 2.1: Syntactic structure of the utterance "Linguistics is fun". On the left the generative syntactic representation and on the right the Tesniére dependency representation.

and all other words are either directly or indirectly connected to it through dependencies. Tesnière grammar offers a flatter representation than the generative approach, as it lacks the intermediate phrasal nodes. A comparison between the generative approach and the Tesnière dependency approach can be seen in figure 2.1. Within computational linguistics a simplified, more generic *dependency grammar* is commonly used.

2.2.1 Computational syntactic parsing

The role of syntactic parsing is to compute the syntactic structure of a given utterance. This is usually achieved by constructing a suitable grammar, which is then used to infer the structure from the utterance. For English there are many grammar formalisms to chose from including: *Context-free grammars* (CFGs) and *Context-sensitive grammars* (CSGs). CSGs are more powerful but also more complex than CFGs. While English does have limited context-sensitive phenomena such as WH-fronting, which would not be supported within CFGs, these phenomena, do not explicitly appear within most evaluation metrics. CFGs are popular choice, as they provide an *acceptable* trade-off between expressiveness and computational complexity. A formal grammar is considered *context-free* when its production rules can be applied regardless of the context of a nonterminal. Context-free rules come in the following form:

$$V \rightarrow w \tag{2.1}$$

where V is a single nonterminal symbol and w is a string of terminals or nonterminals. In natural language syntax the nonterminals refer to syntactic categories such as VP (Verb phrase) and the terminals refer to words. An example grammar can be seen in figure 2.2.

If we consider the sentence *"John loved Mary"* and the example CFG (figure 2.2) then there are two possible syntactic structures for the utterance, which can be seen in figure

- N → "Man"
- N → "women"
- ADJ → "old"
- CONJ → "and"
- NP → N
- NP → ADJ N
- NP → NP CONJ NP

FIGURE 2.2: Example CFG

2.3. To disambiguate the correct structure, a probabilistic element must be introduced. This motivates the use of a probabilistic context-free grammar (PCFG); each grammar rule having a likelihood associated with it. Parsing becomes the task of efficiently finding the syntactic structure from the utterance which the grammar dictates is most likely. The probability of the derivation is the sum of all the grammar rules used, as defined below:

$$P(Derivation) = \prod_{r_i \in derivatives} P(r_i | LHS(r_i)) \tag{2.2}$$

where r_i is the probability of rules and $LHS(r_i)$ is the left hand side of the rule r_i. There are two main approaches to computing the syntactic structure. A top-down parser begins with the start symbol (normally S) then matches rules from the left-hand-side until it reaches terminal symbols (words from the utterance). A bottom-up parser on the other hand starts with the words of the sentences then matches rules from their right-hand-side until it reaches the start symbol (S).

The construction of a PCFG can be done either with supervision or without. I will focus on the supervised approach, where a treebank provides the syntactic structures for a *hopefully* representative set of natural language sentences. One of the earliest approaches to constructing a grammar from a treebank was to use the relative frequency

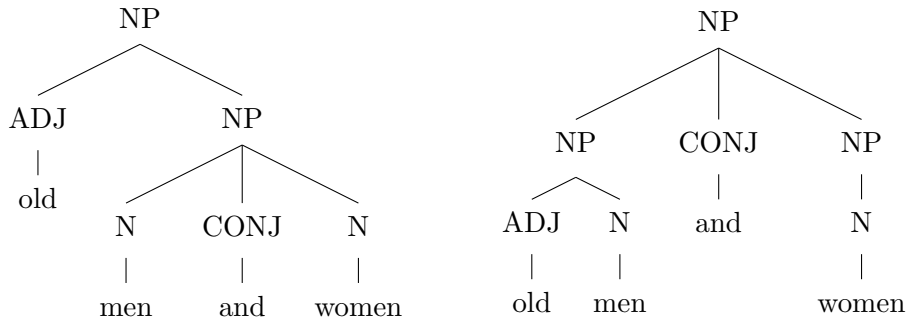


FIGURE 2.3: Two possible syntactic structure of the utterance *old men and women*

of the rules ($\mathcal{F}_R(V \rightarrow w)$) found within the treebank as the likelihood for each context-free rule, as seen in equation 2.3.

$$rf(V \rightarrow w) = \frac{\mathcal{F}_R(V \rightarrow w)}{\sum_{\beta:V \rightarrow B \in R} \mathcal{F}_R(V \rightarrow \beta)} \tag{2.3}$$

This approach however only offered limited success, as the syntactic rules as read from the treebank are seemingly too coarse to capture natural language syntax. Improvements have been suggested from smoothing probabilities to refining the syntactic rules (Klein and Manning, 2003). One possible refinement includes adding context for PCFG rule in the form of *parent annotation* (Charniak and Carroll, 1994), where the parent of the node is concatenated to the child label. For instance an NP with a parent S will now be labelled $NP\hat{}S$. This however breaks the Markovian assumption of the grammar, that all constituents with the same syntactic category are equivalent. Parent annotation has seen generalisation within vertical Markovization which extends the notion of parent annotation by including n members from its vertical history (parents' parent) (Klein and Manning, 2003). However, vertical Markovization increase data sparsity, as each label appears fewer times within the corpus, thus making generalisation difficult. Petrov and Charniak (2011) argue that these refinements are ad-hoc and unsystematic; instead they propose an unsupervised approach to refining the grammar by hierarchically splitting PCFG rules into sub-rules.

2.3 Semantics

Semantics is tasked with providing the conventional *meaning* of an utterance (Speaks, 2014). Within formal semantics there are three core concepts: composition, truth and entailment. Composition determines how linguistic items are combined to form a new item. Truth determines under what conditions an utterance is true or false. Formal semantics borrows ideas from philosophy and uses the concept of possible worlds, where an utterance is true with respect to a possible world (Menzel, 2014). As such formal semantics is model theoretic and truth is determined to a particular world. Entailment is the relationship between two sentences, where the truth of one requires the truth of the other.

2.3.1 Montague Grammar

Montague grammar provides a treatment of the semantics of natural language using intentional logic (Dowty, 1979). It builds upon Freges philosophy, where the meaning of

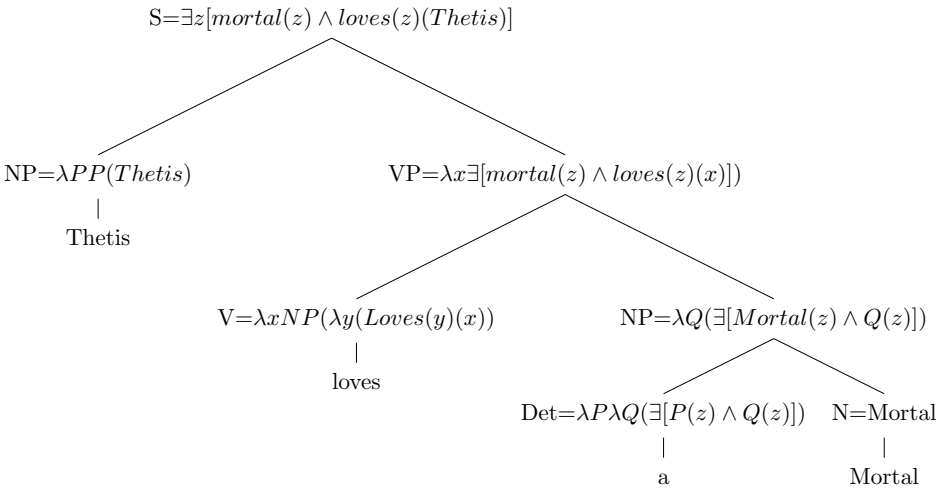


FIGURE 2.4: Semantic tree for *Thetis loves a mortal* as adapted from Schubert (2014)

the utterances is built up in a compositional manner from interim meanings, as given by lambda expressions. Every lexical item has a lambda expression associated with it and every grammar rule defines a composition function dictating how the lambda expressions should be composed. An example semantically annotated tree can be seen in figure 2.4, where the syntactic structure is used to guide which constituents compose with each other.

The standard way of composing two constituents is functional applications (beta reduction). Consider the expression $\lambda x \exists[mortal(z) \wedge loves(z)(x)])$, $\lambda PP(Thetis)$ is then applied to it, with the resulting composed meaning: $\exists z[mortal(z) \wedge loves(z)(Thetis)]$. However, more complex linguistic phenomenon such as quantifier scoping issues and WH-fronting cannot always use just beta reduction, but instead more complex composition procedures are used.

2.3.2 Statistical Semantics

The focus and success within NLP on semantics have been split between sentiment analysis and semantic role labelling (SRL). Neither approach is model theoretical and as such does not reference a particular world. Sentiment analysis which determines whether an utterance is positive or negative has received little attention within formal semantics. However, SRL can be seen as a computational implementation of *Thematic relations* (Carlson, 1984).

SRL is composed of two tasks; (1) the identification of predicates and arguments, (2) determination of the semantic roles of said arguments. The following utterances have been annotated with their semantic roles:

- [$_{A0}$ Eve] pushed [$_{A1}$ Mary]
- [$_{A0}$ Eve] grabbed [$_{A1}$ Mary]
- [$_{A0}$ Eve] will push [$_{A1}$ Mary]
- [$_{A1}$ Mary] was pushed by [$_{A0}$ Eve]

From the above we see each argument of the predicates *push* and *grab* have been identified and labelled with their semantic role, A0 or A1. We also see that *Mary*, regardless of syntactic position, always receives the A1 role. The A1 label is shorthand used to indicate that the bearer of the role fulfils the patient role of the predicate. Dowty (1991) seminally defines the patient role through the use of proto examples, such that the bearer of the patient role must show a family resemblance to the proto-patient. *Eve* on the other hand receives the A0 role, shorthand for the Agent role, as she bears a resemblance to the proto-agent.

Although SRL has semantic properties it can also be considered as an intermediate layer between syntax and semantics and as such not a full semantic model (Carlson, 1984). This motivates the search for a strong semantic model which will be discussed in the next chapter.

Chapter 3

Language Without Symbols

3.1 Introduction

In the previous chapter I gave an account of symbolic approaches to syntax and semantics, where words, phrases and meaning are represented by arbitrary symbols. However, non-symbolic/feature-rich representations of language have become increasingly popular in recent years. Non-symbolic approaches represent linguistic items by non-arbitrary multidimensional vectors; the value of these vectors positions the item in a linguistic space.

Non-symbolic approaches allow for direct measurement of similarity between items by measuring the *distance* between their vectorial representations. This distance explicitly gives us the ability to generalise, where information learnt about one linguistic item can be applied to other linguistic items. This has led to non-symbolic approaches being used in several areas of NLP, including: machine translation (Chiang et al., 2009), semantic role labelling (Ponzetto and Strube, 2006), parsing (Socher et al., 2011) and part of speech tagging (Giménez and Marquez, 2004).

In this chapter I will detail how non-symbolic approaches to semantics can be applied to lexical items. Next, moving onto non-symbolic syntactic semantic representations of multi-word expressions and sentence; a tradition I build upon for my model.

3.2 Distributional lexical semantics

One of the more popular approaches to non-symbolic lexical semantics is distributional semantics (DS). Distributional semantics can be seen as a realisation of the Distributional Hypothesis (DH); words gain their meaning from a distributional analysis over

language and its use. Therefore, words that occur in similar contexts have similar semantic meaning (Harris, 1954). DS models use vectors to keep track of the contexts within which words appear. This vector then represents the *meaning* of the word. Unlike Montague grammar where there is no way to show similarity between items. For example $\lambda.xDead(x)$ and $\lambda.xDeceased(x)$. However, *Dead* and *deceased* appear in similar contexts and as such their vectorial representation will be similar.

Consider the following example utterances with the unknown word *bardiwac*, inspired by Evert (2010):

- A bottle of *bardiwac* is on the table
- *Bardiwac* goes well with fish
- Too much *bardiwac* and I get drunk
- *Bardiwac* is lovely after a hard day of work

The DH states that we implicitly compare the distribution for the word *bardiwac* to other lexical items and we find its distribution is most similar to those of alcoholic drinks.

DS is a very popular approach to lexical semantics and while there have been attempts at making hand-crafted symbolic lexical semantic databases, most notably WordNet (Fellbaum, 1998), these approaches are expensive to create, slow to update and generally cover fewer words than DS approaches. WordNet provides semantic information for $\sim 160,000$ words. DS systems in contrast are unsupervised in nature and therefore are cheaper to create and contain semantic information about significantly wider range of words. Furthermore, DS has compared favourably to WordNet in a wide range of semantic tasks (Lewis and Steedman, 2013) (Specia et al., 2012) (Budanitsky and Hirst, 2006) (Šarić et al., 2012). In lexical substitution tasks DS-based approaches were shown to perform at the same level as native English speakers with a college education (Rapp, 2004). This success has lead to DS being used not only as a semantic analysis for words but also being integrated into NLP systems including: machine translation (Alkhouli et al., 2014), semantic role labelling (Choi and Palmer, 2011) and question answering (Lewis and Steedman, 2013).

While DS achieves strong computational linguistic results, it also has strong linguistic and psychological plausibility. DS can be seen as an implementation of the feature-based theory of semantic representation of the Generative Lexicon (Pustejovsky, 1991), where each lexical item in the generative lexicon has four structures: lexical typing structure, argument structure, event structure and qualia structure. The qualia structure encodes distinctive features of the lexical item, such as size, form and colour. A distributional representation would *hopefully* capture these properties implicitly. However, unlike previous

$$\text{man} = \begin{bmatrix} to : 2 \\ a : 2 \\ but : 1 \\ because : 1 \\ than : 1 \\ are : 1 \end{bmatrix} \qquad \text{forgotten} = \begin{bmatrix} in : 1 \\ the : 1 \\ misfortunes : 1 \\ are : 1 \\ he : 1 \\ had : 1 \\ all : 1 \\ his : 1 \end{bmatrix}$$

FIGURE 3.1: Distributional semantic representation of man and forgotten.

attempts at creating lexical entries, DS takes an unsupervised data-driven approach to creating the lexical entries, better paralleling how children learn language.

3.3 Implementation

I will now give a brief introduction to how distributional semantic systems are implemented. I will later show how these approaches are incorporated into multi-word compositional distributional semantics approaches.

One of the simpler approaches to DS uses co-occurrences as a way to construct semantics vectors for lexical items. For each word, frequency information regarding words which appear *closely* to it is kept. (*Closely* refers to a certain number of words apart as defined by the co-occurrence window size). From the following passage I will give example semantic vector representations of words.

"Within two minutes, or even less, he had forgotten all his troubles. Not because his troubles were one whit less heavy and bitter to him than a man's are to a man, but because a new and powerful interest bore them down and drove them out of his mind for the time–just as men's misfortunes are forgotten in the excitement of new enterprises." (Twain, 1988)

When the co-occurrence window is two words long we get the vectorial representations seen in figure 3.1. Due to the small size of the passage it is difficult to see similarities within the text. Therefore, a longer piece of text is used to create lexical vectors within figure 3.2.

3.3.1 Parameters

In the previous section a simplistic DS system was explained. However, alternative DS approaches have been proposed, with many different choices to be considered when designing a system to extract distributional semantic from a corpus. First, context needs to be defined. In the example implementation a co-occurrence window of two was chosen; this window can be enlarged or shrunk. When enlarging the window weighting is often applied, such that words appearing nearer the target word are given more importance. In the example implementation, the context window spanned across sentence boundaries. However, not all models take this approach, and in yet other models the window spans across paragraph boundaries.

Secondly, the corpus must be decided; large corpora are generally considered better as they offer a more complete view of language distributions. The type of corpus also needs to be determined, in particular whether the corpus is in-domain or out-domain with respect to the application one has in mind. The corpus can also be annotated with part of speech tags, syntactic information or word senses. Each piece of information can act as a new dimension or as a weighting.

Thirdly, frequency information must be considered. In the example above, frequency information came in the form of raw frequencies. Alternatively, they could also be logged frequency or smoothed frequencies. Information theoretic measures such as entropy or pointwise mutual information have also been used within DS models. These approaches try to better capture the true distribution of words from a limited corpus.

Dimension reduction is a common tactic which represents words in a lower dimensional space. This not only decreases the amount of information stored about each word; the compressed vector avoids *"the curse of dimensionality"*, as well as hopefully capturing more generalisable latent semantic information.

$$\text{cat} = \begin{bmatrix} get : 54 \\ see : 70 \\ use : 2 \\ hear : 9 \\ eat : 9 \\ kill : 32 \end{bmatrix} \qquad \text{dog} = \begin{bmatrix} get : 210 \\ see : 64 \\ use : 6 \\ hear : 33 \\ eat : 50 \\ kill : 11 \end{bmatrix} \qquad \text{banana} = \begin{bmatrix} get : 12 \\ see : 5 \\ use : 9 \\ hear : 0 \\ eat : 23 \\ kill : 0 \end{bmatrix}$$

FIGURE 3.2: Distributional semantic representation of cat, dog and banana.

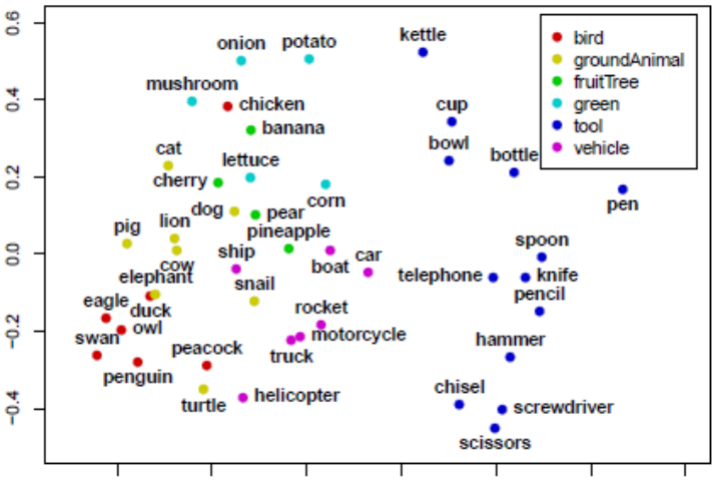


FIGURE 3.3: Words mapped to their semantic position. Adapted from Evert (2010)

3.3.2 Similarity

One of the core advantage of distributional semantics is the ability to measure similarity between words and their vectors by measuring relative positions in the semantic space (figure 3.3). These similarity metrics have a wide range of uses including: finding synonyms, as well as clustering semantically-related concepts (Baker and McCallum, 1998). There are two approaches to measuring similarity within DS: distance-based and angle-based approaches, as seen within figure 3.4. The most common distance-based approach is Euclidean, however Minkowski distance and Manhattan distance have also been used. When measuring similarity using angular approaches cosine is the most commonly used; the Ochiai coefficient is a less well used alternative.

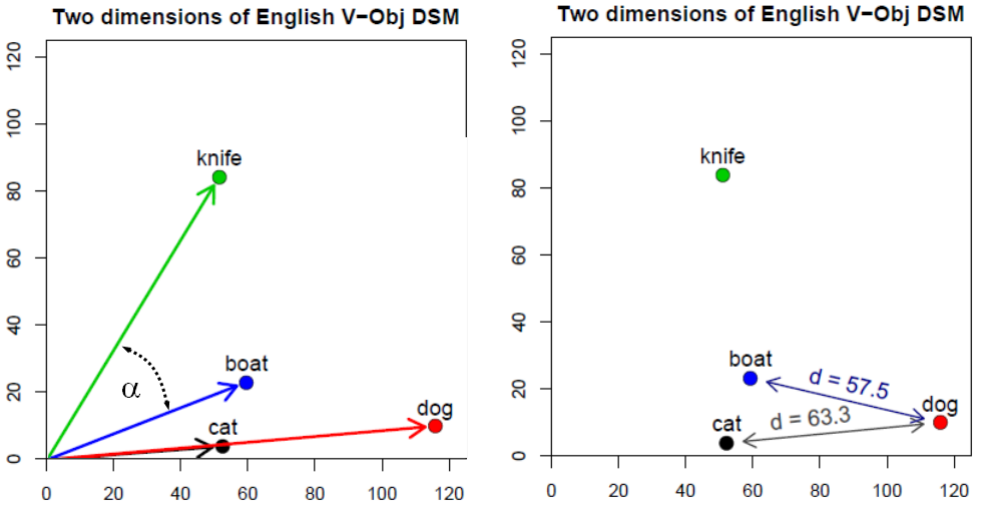


FIGURE 3.4: Comparison between distance and angle based approaches to similarity, as adapted from Baroni et al. (2014a)

3.3.3 Limitations

Although distributional semantics is a popular approach within the NLP community, it is not without critics, who disprove of it from engineering, philosophical and linguistic positions.

Philosophically it encounters the same symbol grounding problem that symbolic approaches face (Massé et al., 2008). Meaning in DS is defined from other words (context) with no connection to the sensory world. However, there has been recent work that integrates information from images into DS models partially negating this criticism (Bruni et al., 2012).

From an engineering perspective polysemy may be difficult to capture within DS, as each word receives only one vector. The size of vector representing a word with one sense is the same as for a word with multiple word senses. Experimentally this however does not seem to be problematic, as many studies have shown that polysemy is capturable within DS (Pantel and Lin, 2002) (Boleda et al., 2012).

From a linguistic perspective it has been argued within the weak DH that DS does not capture *meaning* (qualia) but instead semantic paradigmatic properties (combinatorial behaviour) of words (Sahlgren, 2008). This often seen with antonyms with DS, as they often are given similar distributions. This is particularly problematic in synonym generation tasks where antonyms will be suggested as a synonym.

3.4 Compositional distributional semantics

3.4.1 Introduction

For many years, the standard way to represent compositional semantics, was to use lambda calculus (Montague, 1970), and the most successful way to model lexical semantics, based on the vector representations from *distributional semantics* (e.g., Lund et al., 1995), seemed incompatible (Le and Zuidema, 2014b). However, there has been a recent trend in trying to combine distributional semantics and compositional semantics, forming distributional compositional semantics. Within this section I will outline several approaches to compositional distributional semantics and the reasons I did not take these approaches.

3.4.2 Composition by vector mixtures

Early attempts at composing multiple vectors involved simple linear algebra operations, starting with vector addition and later point wise vector multiplication, which better captures interaction between the values of the input vectors. While both approaches have been shown to be effective in capturing the semantics of multiple words, they fail to capture structural relationships and word order. These problems can be seen in the utterances "the dog bit the man" and "the man bit the dog" which would compute identical vectors and therefore within vector mixture models, identical meaning. Structurally, both approaches are symmetric; each vector contributes equally. However, this does not match linguistic theory, where some linguistic types dominate the compositional relationship. For instance, a verb phrase is normally composed of a verb and a noun phrase. As the verb is the head word, linguistically it is more important, which cannot be expressed in mixture models. One proposed solution to this weakness involved scaling the input vectors to indicate importance (Mitchell and Lapata, 2008). However, scaling still fails to capture the syntactic structure of the utterance. Therefore, it is difficult to see this as a Montagovian approach as Montague uses syntax to guide the semantic composition, where a different syntactic structure would give a different semantic meaning.

3.4.3 Composition with distributional functions

3.4.3.1 Combined Distributional and Logical Semantics

The approach of Lewis and Steedman (2013) layers distributional semantics on top of a formal semantic representation. The addition of distributional semantics improves the flexibility of the semantic representation, giving significant improvements to question answering tasks.

The process of Lewis and Steedman (2013) is a pipeline: First the input utterance is semantically parsed, giving a lambda expression for the utterance using a semantic parser (Curran et al. (2007)). To disambiguate polysemous predicates, entity-typing is applied to predicate arguments; each predicate is typed with two argument types. Finally, the typed predicates are replaced with a link to a typed semantic cluster. The clusters represent semantically-similar concepts as determined by distributional semantics. The utterance is now represented by a lambda expression with the predicates representing concepts and not individual words. An example of the process can be seen in figure 3.5.

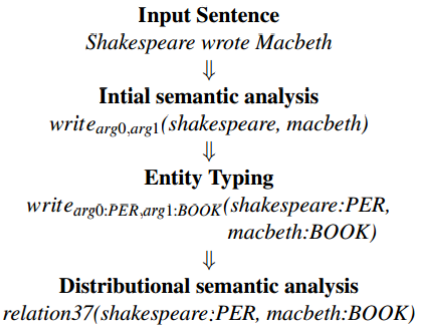


FIGURE 3.5: Pipeline of *Combined Distributional and Logical Semantics* approach adapted from Lewis and Steedman (2013)

While the approach offers improvements over a standard symbolic parse, it lacks feedback between the distributional layer and the semantic parser; the distributional semantic information cannot affect the structure computed for the utterance. Furthermore, the approach is reliant on using an existing symbolic semantic parser. These are significant weakness to this approach motivate an alternative approach.

3.4.3.2 Tensor approach

The tensor approach to compositional vector semantics is focused on functional application from formal semantics. Nouns, determiner phrases and sentences are vectors but adjectives, verbs, determiners, prepositions and conjunctions are modelled using distributional functions, allowing for a separate treatment of functional words and content words. Baroni et al. (2014b) propose that the distributional functions take the form of linear transformations. First order (one argument) distributional functions (such as adjectives or intransitive verbs) are encoded as matrices. The application of a first-order function to an argument is carried out using a matrix-vector multiplication. Second order (two arguments) such as transitive verbs or conjunctions are represent by a three dimensional tensor. Learning the tensor representations is done using standard machine learning techniques as applied to a treebank.

While the tensor approach seems reassuringly similar to formal semantics, it has several drawbacks. First, the model encodes a lot of *a priori* information, in the form of the dimensionality of the word. Secondly, the highly parameterised approach over-fits the data. Thirdly, the learning of these representations is challenging from a machine learning perspective and no convincing results have been reported.

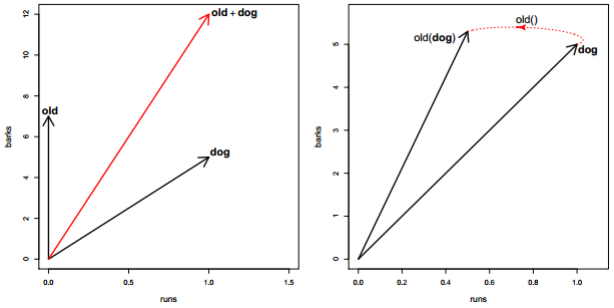


FIGURE 3.6: mixture model on the left and tensor functional application on the right (Baroni et al., 2014a)

3.4.4 Summary of approaches

Below I list several popular approaches to compositional distributional semantics. Where (a,b) are vectors and (A,B) are matrices.

Composition function	Name	Source
p= a+b	Vector addition	
p = 0.5(a + b)	Vector average	
p = [a;b]	vector concatenation	
p= a $\otimes$ b	Element-wise vector multiplication	
P = Ab + Ba	Linear MVR	(Mitchell and Lapata, 2010)
p= Aa+Bb	Scaled vector addition	(Mitchell and Lapata, 2008)
p=tanh(W[a;b])	RNN	(Socher et al., 2010)
p = tanh(W[Ba;Ab])	MV-rnn	(Socher et al., 2012)

Chapter 4

Recursive Neural Network

4.1 Introduction

At the end of the last chapter I presented several approaches to modelling compositional distributional semantics and gave the disadvantages of such approaches. In this chapter I will explain the approach I have taken; the Recursive Neural Network (RNN). I however first introduce the idea of connectionism and Artificial Neural Networks, traditions which I build upon. I start by introducing connectionism and its applicability to language, then give a general description of Neural Networks and in particular the feedforward network. I then introduce the **Recurrent** Neural Network which is later contrasted with the **Recursive** Neural Network. Finally, the Recursive Neural Network is explained in detail; including how it is used to model syntax and semantics.

Connectionism is an approach to modelling cognition, where knowledge underlying cognitive activities is stored in the connections among neurons (McClelland et al., 2010). Connectionism borrowed ideas from neuroscience, leading to the idealised artificial neuron. Networks of artificial neurons (Neural Networks) have a long history of being used for a wide range of machine learning problems. However, they are particularly appealing to use in the modelling of syntax and semantics due to the close relationship between language and cognition. Recent advances in deep learning have also made Neural Networks a particularly exciting technique to work with. Deep learning avoids the problem of hand-crafting features which is not only a time-consuming task but often leads to errors, where features are either overspecified or underspecified. Deep Neural Networks, when applied to language problems receive all the benefits and flexibility non symbolic approaches offer (as seen in section 3) and have inherit deep learning advantages.

4.2 Neural Networks

A Neural Network consists of a series of connected artificial neurons; each neuron implements a logistic regression function. Where, a neuron (figure 4.1), takes in a series of inputs x_i to which weights w_{ij} are applied. These weighted inputs are summed and an activation function is applied giving the output value. For compactness I will define a neuron in vector form:

$$a = f(w^T x + b) \tag{4.1}$$

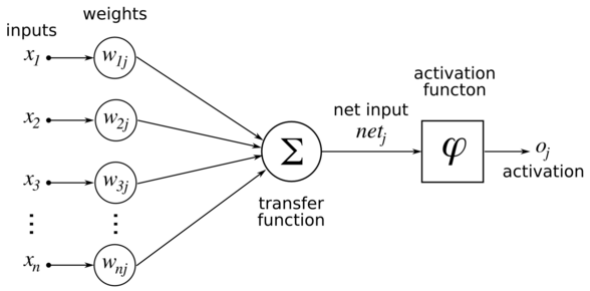


FIGURE 4.1: An artificial neuron[1]

where $w \in \mathbb{R}^n$ are the weights, $x \in \mathbb{R}^n$ the inputs and b is the bias, f is the activation function. Tanh ($\frac{1-e^{-2x}}{1+e^{-2x}}$) and the sigmoid function ($\frac{1}{1+e^{-t}}$) are popular activation functions (figure 4.2). (Tanh is a rescaled and shifted sigmoid function.)

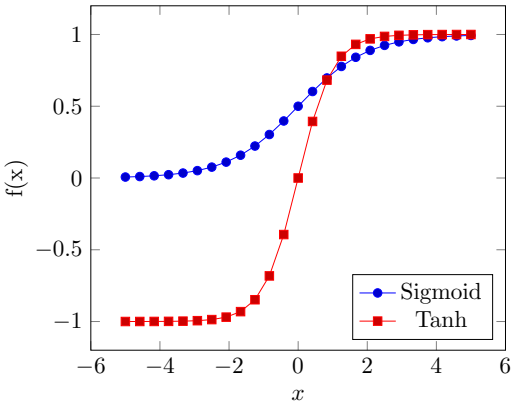


FIGURE 4.2: tanh and the sigmoid function

A neural network consists of many artificial neurons connected together in any topological arrangement. One of the earlier and most common topological arrangement is the feedforward network. The feedforward network consists of a series of layers of neurons,

[1]By Perceptron. Mitchell, Machine Learning, p87. [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

where information flows in one direction through it, each neuron in a layer is connected to every neuron in the next layer. An example can be seen in figure 4.3. The feedforward network conceptually consists of three types of layers. The input layer, hidden layers and an output layer. The input to the network consists of the features chosen to represent the symbolic item. The hidden layers sit between the input and the output layers. The output layer then outputs the *answer*. Feedforward networks with a non zero number of hidden layers has been shown to approximate the solution for any problem (universal approximator) (Hornik et al., 1989) (Cybenko, 1989). The input layer within the feedforward network is defined as:

$$z = Wx + b \tag{4.2}$$

$$a = f(z) \tag{4.3}$$

where $W \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, the activation function f is applied in an element-wise manner. All other layers are defined in formulas 4.4 and 4.5. An n superscript is included to distinguish between layers.

$$z^{n+1} = W^{n+1}a^n + b^{n+1} \tag{4.4}$$

$$a^{n+1} = f(z^{n+1}) \tag{4.5}$$

To produce meaningful answers the weights and the bias of each layer within the neural network must be learnt. To do so a loss function, such as the squared error rate, is defined and then minimized. A popular way to minimize the loss function is through the use of backpropagation where weights within the network are adjusted depending on how much they contributed to the error. The process, as the name suggests, works backwards from the output layer to the input layer using the errors calculated in the previous layer for the new layer. The layers closest to the output layer being the most influential layers in regards to the error.

Although powerful, the feedforward approach has several problems. Firstly, the input size has to be known ahead of time, as there must be a corresponding number of input neurons. This is problematic when modelling sentences which contain a variable number of words. Secondly, when backpropagation is used to train a network with a large number of hidden layers, the error contributed will be small in the layers closest to the input, making adjusting weights difficult. An alternative neural network architecture was proposed; the Recurrent Neural Networks. I will focus on the Simple Recurrent Neural Network (SRNN) implementation (Elman, 1990). Within the SRNN the connections

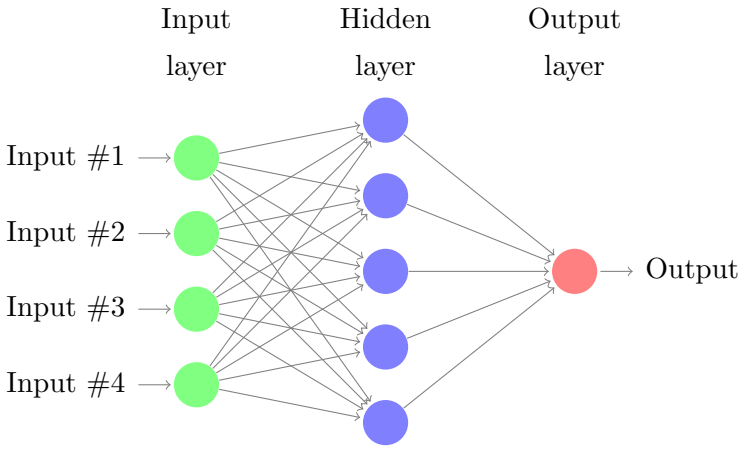


FIGURE 4.3: A three layer feedforward neural network

between units form a directed cycle; these cycles create an internal state which allows the network to exhibit temporal behaviour. In essence the network uses previous inputs to guide future outputs. In figure 4.4 we see that this temporal behaviour takes the form of the previous input being used as a context for new inputs. At each step the hidden units are copied to form new context units. SRNNs have been popular approaches for modelling compositional distributional semantics, where the words are fed into the SRNN one word at a time; The previous words forms the context vector for the next word. This repeats till there are no words left, giving a single output representing all the words.

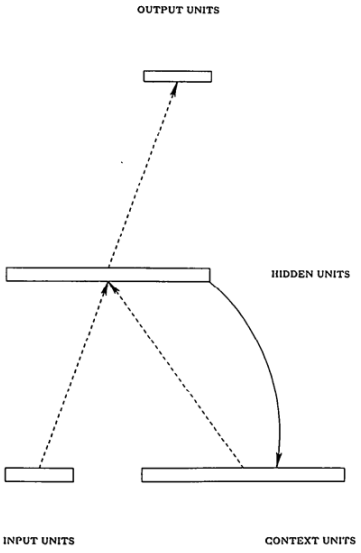


FIGURE 4.4: Simple Recurrent Neural Network. Adapted from (Elman, 1990)

4.3 Recursive Neural Networks

4.3.1 Introduction

The Recursive Neural Network (RNN) is a generalisation of the previously mentioned SRNN and is the basis of my approach for modelling syntax and semantics (Irsoy and Cardie, 2014). When the SRNN is used to model language, words are given as an input in a temporal order where all the previous words combine with the next word. We can therefore think of the SRNN as a left-branching binary tree. The RNN removes the restriction of being left-branching and instead allows any continuous[2] binary tree. This internal structure allows us to model the syntactic structure of the utterance and not just the temporal order with utterance. A comparison of the two approaches can be seen within figure 4.5. As the RNN considers the syntactic structure when composing, the approach is closer to Montague grammar where it is not just the semantic elements being composed but how they are being composed. With the many possible structures for which a RNN could construct, an additional scoring element is introduced at each node, where the likelihood of the entire tree is the sum the likelihood of all nodes, paralleling the change from CFG to PCFG.

Using the RNN as the basis of the model we are offered the flexibility of distributional semantics applied to the entire sentence structure, where one root vector represents captures the meaning of the entire sentence. As with individual words we can now compare the distance between sentences, which will later be used in paraphrase detection tasks. The RNN based approach differs significantly from the tensor approach discussed

[2]The tree has no crossing elements.

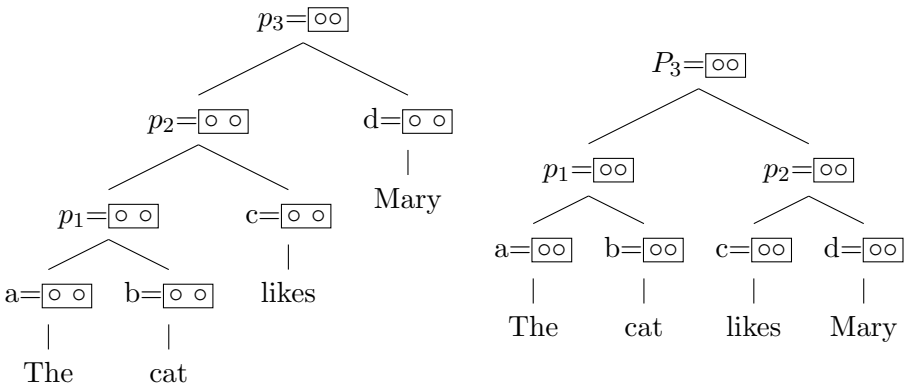


FIGURE 4.5: Two representations of the utterance "The cat likes Mary". On the left a Recurrent Neural Network representation capturing the temporal word order of the utterance. On the right a Recursive Neural Network capturing the syntax of the utterance.

in section 3.4.3.2. Unlike the tensor approach which requires *a priori* the dimensionality of each word, the RNN fixes the dimension of the words uniformly. When compared to symbolic approaches similarities can be seen with vertical Markovization in the PCFG-based approach. The RNN is a non-Markovian process where the effects of words directly influence the root. However, unlike PCFG vertical Markovization annotation, the RNN can make use of *infinite* vertical history without the problems of data sparsity, due to its ability to generalise.

Within this section I will provide in detail the RNN approach to language, as given by Socher et al. (2010). I start by explaining the framework itself, before moving onto its training including how backpropagation works and the learning algorithm chosen.

4.3.2 Mapping words to syntactic/semantic space

As with distributional semantics, words within a RNN framework have two representations, the symbolic w representation and a corresponding N dimensional vectorial syntactic/semantic representation a_w. A sentence is defined as a list of tuples $x = [(w_1, a_{w_1})...(w_m, a_{w_m})]$. Within the RNN model vectorial representations of words can either be learnt directly or an existing *flat* distributional semantics lexicon can be used.

4.3.3 Composition

Composition is key to the RNN model and takes inspiration from the Montague approach, where the meaning of two constituents is combined into a new meaning. However, unlike Montague grammar which is symbolic and uses lambda expressions this approach is non-symbolic, the intention is for the vectors to act as enriched lambda expressions.

Within an RNN a composition function is defined where two N dimensional word vectors can be combined into one N dimensional parent vector. The vectorial representations of both words are given as inputs to a neural network which then outputs one vector, representing the composition of these two items. To do so, the two words vectors are concatenated into one vector, to act as a single input. This concatenated vector is then multiplied by a weight matrix before applying an element-wise activation function. Formally we define this as:

$$P(i,j) = f(W[a_{w_i}; a_{w_j}] + b) = f\left(W \begin{bmatrix} a_{w_i} \\ a_{w_j} \end{bmatrix}\right) \tag{4.6}$$

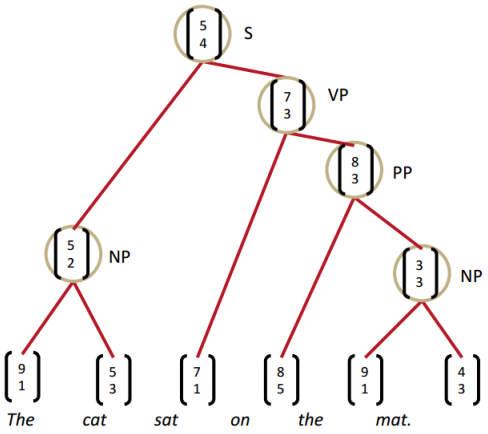


FIGURE 4.6: RNN approach, as adapted from (Socher et al., 2012)

where f is the activation function (in this approach tanh is used), $W \in \mathbb{R}^{n \times 2n}$ and $P(i,j)$ is the vectorial representation for the parent of two children a_{w_i} and a_{w_j}. Note that $[a_{w_i}; a_{w_j}]$ represents the concatenation of two vectors.

The difference between the RNN and the feedforward network is the recursive nature of the RNN, where vectors created from the composition of two words can then be composed using a general formula, such that intermediate p node can be combined with words or other intermediate nodes in a similar way. A general formula is shown in formula 4.7.

$$P(i,j) = f(W[c_i; c_j] + b) = f\left(W \begin{bmatrix} c_i \\ c_j \end{bmatrix}\right) \tag{4.7}$$

The recursive nature allows for the tree to be built in a bottom up manner(figure 4.6), where the vectors of each node capture the interaction of all their children vectors. As there are multiple possible syntactic structures for an utterance a statistical aspect is introduced for the purpose of disambiguation. Therefore, a score is given at each non terminal node indicating the RNN confidence of the correctness of the node. The score is calculated by from the inner product of the parent vector with a row vector $W^{score} \in R^{1 \times n}$, as shown below:

$$s(i,j) = W^{score} P(i,j) \tag{4.8}$$

Conceptually we can think of the Neural Network (W, W^{score}), which I later refer to as the composition function, receiving two input vectors, then outputting one composed vector and a confidence score, as seen in figure 4.7. As the RNN is recursive, a tree can

be defined be a series of these outputs, thus:

$$RNN(x, \hat{y}, \theta) \tag{4.9}$$

where, θ is the set of the parameters of the models containing W, W^{score} and $\hat{y}$ is the structure of the tree, i.e. which nodes are composed together. With multiple possible trees for each utterance x, a score is given to each tree $\hat{y}$, which is the sum of all local scores:

$$s(RNN(\theta, x, \hat{y})) = \sum_{d \in \hat{y}} s(d) \tag{4.10}$$

where d is a subtree of the tree $\hat{y}$. Thus the most likely tree for the utterance x, parametrised by θ is:

$$\hat{y} = \arg\max_{\hat{y'}}(s(RNN(\theta, x, \hat{y'})) \tag{4.11}$$

4.3.3.1 Parsing with RNN

The CYK algorithm is used to find the tree which satisfies formula 4.11 (highest scoring tree). Due to the comparatively[3] expensive nature of parsing within the RNN framework the beam search heuristic is used to find the approximate highest scoring tree. Those readers not interested in the technical details of the model can skip the remainder of the section, with the takeaway message that parsing is done in a bottom up fashion, as defined within section 2.2.1.

[3]When compared to symbolic PCFG parsing.

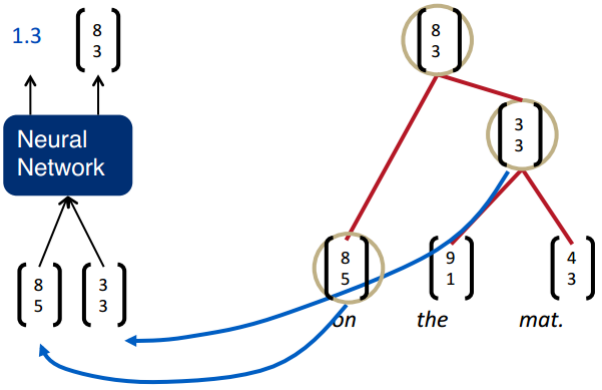


FIGURE 4.7: RNN inputs and outputs as inspired by (Socher et al., 2012)

Finding the highest scoring tree from an utterance, with RNNs takes a different approach from symbolic PCFG approaches. PCFG approaches use discrete syntactic labels whereas RNNs use continuous vectors, meaning pruning cannot be done on label equality. For this reason and the cost of computing vectors, a beam search heuristic is employed, to find the approximate highest scoring syntactic tree. The beam search is applied over the standard bottom-up CYK algorithm at the cell level. The algorithm is defined as follows:

```
Let S consist of n tokens: a_1...a_n. for each i = 2 to n – Length of span do
    for each j = 1 to n-i+1 – Start of span do
        for each k = 1 to i-1 – Partition of span do
            for treeLeft in P[j,k] do
                for treeRight in P[j+k,i-k] do
                    P[j,i].append(RNN(x,[treeLeft; treeRight],θ))
                end
            end
        end
        Prune(P[j,i],beamWidth)
    end
end
```

Where Prune(P[j,i],N) only keeps the N highest scoring trees[4] and removes the rest.

From Socher et al. (2010) a greedy search was shown to be adequate, hence the beam size is often set to one. In this thesis, experiments on pruning at the span level rather than at the cell level did not offer better results and was found to be slower.

4.3.4 Learning

Unlike Montague grammar, the RNN parameters (θ) must be learnt from data using machine learning techniques. Therefore, a loss function must be defined depending on the goals of the model. For the RNN model I choose a loss function that maximizes the score for the correct syntactic structure. While this may appear to be a purely syntactic loss function and at odds with the goal of modelling both syntax and semantic. This is not the case, firstly, the syntactic structure plays a large role in semantic understanding; the syntactic structure guides the semantic composition. The work of Levin (1993), shows that those verbs that are semantically related behave in a syntactically similar manner. Secondly, the RNN is not Markovian but instead the root vector captures all

[4]The trees scores are determined by formula 4.10.

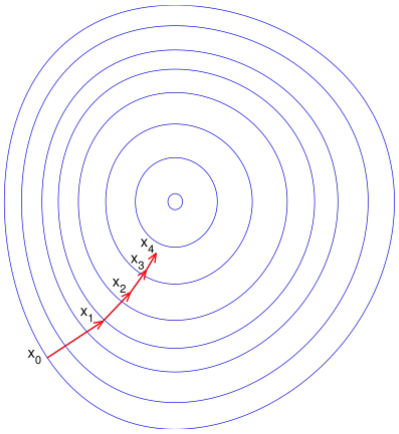


FIGURE 4.8: Example of gradient descent [5]

the information regarding all of its children, including the terminal words. This process is not syntactic in nature, but instead better resembles semantic Montague grammar.

Within this section I will explain the loss function I have chosen, called the Max-Margin, framework which tries to increase the score of the correct tree and decrees the score of the incorrect tree. Those readers not interested in the mechanics of machine learning can skip to section 4.4.

4.3.4.1 Max-Margin estimation

There are two main types of machine learning models generative models and discriminative models. Generative approaches are based on the likelihood of the joint variables $P(X, Y)$; whereas discriminative approaches are based on conditional likelihood $P(X|Y)$. Discriminative approaches have generally been to give more favourable results, as such I will use a Max-Margin framework as proposed for parsing by Taskar et al. (2004).

The Max-Margin framework defines a *loss function* which gradient descent will try to minimize. The loss function gives a score of *goodness* to a particular set of a parameters. These scores can be mapped within a space against parameters. Gradient descent starts with an initial set of parameter values and iteratively moves toward a set of parameter values that minimize the function, which can be seen in figure 4.8. This iterative minimization is achieved calculating the gradient for the loss function and updating the weights with it, as seen in online gradient descent, equation 4.12.

[5] By Gradientdescent.png:The original uploader was Olegalexandrov at English Wikipedia derivative work: Zerodamage (This file was derived from:Gradient descent.png) Public domain, via Wikimedia Commons

$$\mathbf{x}_{n+1} = \mathbf{x}_n - LR\nabla F(\mathbf{x}_n),\ n \geq 0. \tag{4.12}$$

where LR is the learning rate and $\nabla F(\mathbf{x}_n)$ is the gradient of the loss function. Below I list the specifics of the Max-Margin framework. Intuitively, the objective of the Max-Margin framework is that the highest scoring tree past a specified margin of error produced from the model should be the correct tree.

This margin of error comes in the form of a structured loss $\delta(y_i, \hat{y})$ for predicting $\hat{y}$ for the gold tree y_i. Where the further incorrect the tree is the bigger the loss. *Incorrectness* is calculated by by counting the number of nodes with an incorrect span, formula 4.13.

$$\delta(y_i, \hat{y}) = \sum_{d \in N(\hat{y})} k\{d \notin N(y_i)\} \tag{4.13}$$

k is a real valued hyperparameter. The Max-Margin trains the RNN such that the highest scoring tree will be the correct tree up to a margin, over all other possible tree $\hat{y} \in Y(x_i)$:

$$s(RNN(\theta, x_i, y_i)) \geq \arg\max_{\hat{y}}(s(RNN(\theta, x_i, \hat{y})) + \delta(y_i, \hat{y})) \tag{4.14}$$

To prevent the learning algorithm from overfitting the training data regularization is introduced, adding a penalty for complexity. This regularization can be seen as a form of smoothing which has been shown to be advantageous within PCFG. The regularized loss function used for learning is:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} r_i(\theta) + \frac{\lambda}{2}||\theta||_2^2 \tag{4.15}$$

$$r_i(\theta) = \max_{\hat{y} \in Y(x_i)} (s(RNN(x_i, \hat{y}) + \delta(y_i, \hat{y})) - s(RNN(x_i, y_i)) \tag{4.16}$$

Where m refers to the batch size, ranging from the length of the corpus (batch training) to one (on-line learning).

4.3.4.2 Gradient

For the gradient descent the gradient of loss function must defined, however the objective J of equation 4.15 is not differentiable due to hinge loss (Socher et al., 2010). The subgradient method is used instead, which computes a gradient-like direction called the subgradient (Ratliff et al., 2007):

$$\sum_i \frac{\partial s(x_i, y_{max})}{\partial \theta} - \frac{\partial s(x_i, y_i)}{\partial \theta} \tag{4.17}$$

AdaGrad is a popular gradient descent algorithm, which has been show to achieve state of the art performance when training RNNs. Unlike other approaches, AdaGrad alters its update rate feature by feature based on historical learning information. Frequently occurring features in the gradients get small learning rates and infrequent features get higher ones (Duchi et al., 2011). The update to the weights of all individual features is as follows:

$$x_{t+1} = x_t - N \cdot G_t^{-(1/2)} \odot g_t \tag{4.18}$$

where $x \in R^{1\times n}$ is the weight, with the subscript indicating what time step it is at, $g_t \in R^{1\times n}$ is the current gradient, N is the learning rate, $G \in R^{1\times n}$ is the historical gradient and $\odot$ is element-wise multiplication. We see this approach is very similar to online gradient descent (section 4.3.4.1), with the addition of the historical gradient. The historic gradient minimizes the sensitivity to the learning rate, making the model less dependent on hyperparameters. We set the historical gradient at each iteration as:

$$G_{t+1} = G_t + (g_t)^2 \tag{4.19}$$

4.3.4.3 Backpropagation Through Structure

To calculate the subgraident, Backpropagation Through Structure (BTS) is used. BTS is a modification of backpropagation used for RNN and not feedforward networks [6]. As there are two parameters to learn I take derivatives with respect to W and Ws. As adapted from Socher (2014), I will first show how to calculate with respect to W ($\frac{\partial s(x_i,y_i)}{\partial W}$).

[6] BTS is a general case of Backpropagation Through Time, which is restricted to recurrent neural networks.

BTS works from the root down calculating how much each node contributed to the error. The local error of the root P is the derivative of the scoring function of the vector:

$$\delta^p = f'(p) \otimes W^{score} \tag{4.20}$$

where $\otimes$ is the hadamard product (entrywise product) and the derivative of f (tanh) is:

$$1 - tanh^2 p \tag{4.21}$$

The error δ is then passed down to each of the child of P.

$$\delta^{p,down} = (W^T \delta^p) \otimes [c1, c2] \tag{4.22}$$

As the structure is a tree the error $\delta^{p,down}$ is split in half, each child takes their corresponding error message. δ is :

$$\delta^{c1} = \delta^{p,down}[1 : N] \tag{4.23}$$

If c1 is a vector representing a word then this is the error the word representation contributed to the whole tree. However, if C1 is an internal node then the scoring of the node also contributed to the error in the same way formula 4.19 the local score is added:

$$\delta^{c1} = \delta^{p,down}[1 : N] + f'(c1) \otimes W^{score} \tag{4.24}$$

The error message δ is then summed for all nodes to give the total error. When taking the gradient with respect to W^{score} ($\frac{\partial s(x_i, y_i)}{\partial W^{score}}$) the error at each node is the sum of the derivative of the vector at each node.

4.4 Conclusion

In this section I outline the RNN approach to syntax and semantics as inspired by Socher et al. (2010). The results from Socher et al. (2010) while encouraging fall short of the state of the art. Socher et al. (2010) propose several enrichments[7] which do achieve state of the performance, these approaches however detract from the Montagovian aspect of

[7] See appendix B.

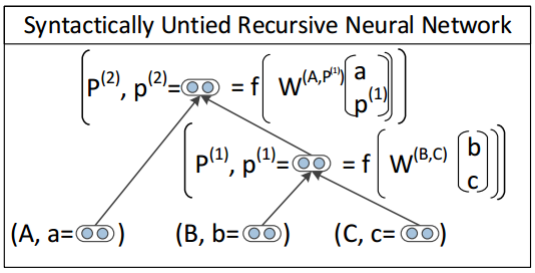


FIGURE 4.9: CVG-RNN approach as adapted from Socher et al. (2013)

the model. Instead, I will focus on the approach of Socher et al. (2013) (CVG), which combines a symbolic and a non symbolic approach. It uses the quick symbolic approach to provide a KBest parse list, the non symbolic approach then reranks this list. Unlike the approach within this section where there is only one composition function, Socher et al. (2013) model uses a composition function for each pair of syntactic categories (figure 4.9). This results in the model consisting of over 900 such composition functions. The new combinatory function seen within formula 4.25 :

$$p = f\left(W^{(B,A)} \begin{bmatrix} a \\ b \end{bmatrix}\right) \tag{4.25}$$

where B and A refer to syntactic categories. The scoring formula incorporates both the symbolic probability and the scoring layer from the RNN:

$$s(p) = (v^{(B,A)})^T p + \log P(P \to BA) \tag{4.26}$$

Socher et al. (2013) not only obtained state of the art results but also introduced a hybrid symbolic non-symbolic model. This approach provides motivation for my model which I explain in the next chapter.

Chapter 5

Enriched Recursive Neural Networks

5.1 Introduction

In the previous section I outlined the one composition function RNN for syntax and semantic modelling. I then discussed the state of the art performance achieved by Socher et al. (2013). This approach however jumped from the use of one composition function within the original model to over 900. Not only does this increase in the number of composition functions drastically expand the search space for an already computational expensive model, but it does not show that such a large increase in the number of composition functions is needed. I will explore whether a small number of composition functions can instead achieve similar improvements.

One previous solution to finding a small number of composition functions is for each N most frequent syntactic rules to have a unique composition function. All other rules share one composition function. However, this is *ad hoc* and not in line with linguistic theory. Instead, I will find core linguistic composition functions which behave uniquely. To do so I define core linguistic types which behave differently when they compose. While there are many possible core linguistic types leading to core composition functions. I will take advantage from work found within the previous symbolic NLP literature. The seminal paper Collins (1997) proposes to use both head, argument and adjunct annotations to enrich their symbolic parsing model. The syntactic rules of the model in addition to including the constituents syntax category also included the head or argument or adjunct category. While Collins (1997) successfully applied this information to a symbolic model, I propose applying this approach to the non-symbolic RNN. However, a direct application would lead to more composition functions than proposed in Socher et al. (2013), as the

syntactic rules are further refined. Instead, I will categorize constituents using just head, argument and adjunct categories, discarding the syntactic categories. A PCFG approach just using these distinction would be far too coarse[1]. Due to the semantic aspect of the RNN model this does not seem to be as detrimental, an RNN based approach with one type achieving results higher than standard PCFGs. Following this approach but applying it to a neural architecture rather a probabilistic grammar, I will categorize constituents within the syntactic tree with head, argument or adjunct types, where different W and W^{score} matrices are used to compose differently typed linguistic items are composed.

The choice of head, argument adjunct is particularly appealing within the RNN model as these categories have both a syntactic and semantic aspect, further cementing the joint syntactic-semantic approach within the model. Within x-bar theory these categories also form three of the four core linguistic types, the fourth category being the specifier (Jackendorff, 1977). Partially due to the lack of annotation resources, and partially due to the exclusion in later Chomskan approaches, I do not make use of the specifier type. Due to the focus on the head constituent this approach also brings us closer to dependency grammar.

In this chapter I will provide a specification of both head, argument and adjuncts. I then explain how the model will be changed in order to account for multiple composition functions. Finally, I discuss the specifics of my proposed models, including both reranking and binarization.

5.1.1 Head

Across the many differing linguistic traditions there are is a broad agreement that there at least two types of constituent heads and dependents. Syntactically, the head is the constituent which syntactically dominates the entire phrase; it determines the semantic/syntactic type (Corbett et al., 1993). Below lists the eight candidate criteria for the identification of a constituent as a syntactic head as written within Corbett et al. (1993):

- Is the constituent the semantic argument, that is, the constituent whose meaning serves as argument to some functor?
- Is it the determinant of concord, that is, the constituent with which co-constituents must agree?
- Is it the morphosyntactic locus, that is, the constituent which bears inflections marking syntactic relations between the whole construct and other syntactic units?

[1]Results of this approach can be seen in section 6.4.2

- Is it the subcategorizand, that is, the constituent which is subcategorized with respect to its sisters?
- Is it the governor, that is, the constituent which selects the morphological form of its sisters?
- Is it the distributional equivalent, that is, the constituent whose distribution is identical to that of the whole construct?
- Is it the obligatory constituent, that is, the constituent whose removal forces the whole construct to be recategorized?
- Is it the ruler in dependency theory, that is, the constituent on which others depend in a dependency analysis?

5.1.2 Arguments and Adjuncts

A further distinction can be made between dependents that are arguments and those that are adjuncts (Kay, 2005). Syntactically, arguments are constituents that are syntactically required by the verb, whereas adjuncts are optional. Semantically, argument meaning is specified by the verb. Adjuncts meaning is static across all verbs. Consider the utterance "John pushed Mary yesterday". Both "John" and "Mary" are arguments hence gain their meaning from the verb as a pusher and a person being pushed respectively. "yesterday" is an adjunct hence its meaning is independents of the verb.

5.1.3 Annotation

An existing corpus is annotated with head, adjunct and argument information using an extended version of the heuristic found within Collins (1997), seen in appendix C. The heuristic considers the labels of: siblings, parents and children to determine head, argument or adjunct type. These labels include the syntax category of the constituent and semantic information. Where, the semantic information comes in the form of labelling the constituent from a limited number of theta roles.

5.1.4 Algorithmic changes

To incorporate multiple composition functions the RNN model is redefined such that multiple W and W^{score} matrices are used. To do so the model is now parameterized by the collections CW and CW^{score}. This requires a change in the CYK algorithm

to account for two children being composed with different composition functions. This change can be seen below:

Let S consist of n tokens: $a_1 ... a_n$. **for** *each i = 2 to n – Length of span* **do**
  for *each j = 1 to n-i+1 – Start of span* **do**
    for *each k = 1 to i-1 – Partition of span* **do**
      for *treeLeft in P[j,k]* **do**
        for *treeRight in P[j+k,i-k]* **do**
          for W, W^{score} *in* CW, CW^{score} **do**
            $P[j,i].append(RNN(x, [treeLeft; treeRight], W, W^{score}))$
          end
        end
      end
    end
    Prune(P[j,i],beamWidth)
  end
end

Where prune now prunes not only on which constituents to combine but also which composition function is used to do so. With multiple composition functions the changes to the calculation of the subgradient are small, due to the chain rule where the error is calculated assuming that each W matrix used is independents from one and other.

5.2 Models

To examine the impact that the head, argument and adjunct distinctions makes, I propose six models with varying levels of linguistic enrichment. The first model, the *BRNN*, is a near[2] replication of the work of Socher et al. (2010), with no linguistic enrichment. There is just one composition function (figure 5.1).

The second model (RNN-Head) enriches the model with head information. The model makes a distinction between two types of constituents: the linguistic head and those that are not (the dependents). As such two composition functions are defined, one which composes heads and dependents, and one which composes dependents and dependents. An example of the different composition functions can be seen in figure 5.2. The dependent-dependent composition function is a result of binarization. If the tree had no ternary or greater constituents there would be no need for the second composition function, as each binary parent phrase has one child which is the head constituent.

[2] AdaGrad was chosen as the learning algorithm and not LBFGS.

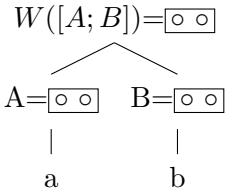


FIGURE 5.1: Composition function for the BRNN

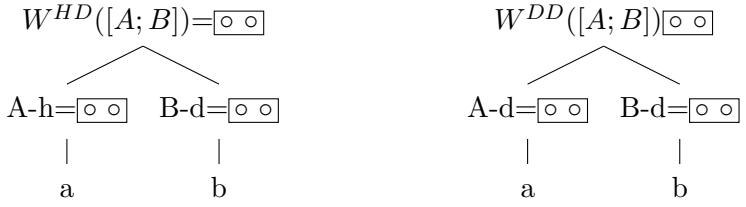


FIGURE 5.2: Composition function for the RNN-Head. The h flag indicates the constituent is the linguistic head. the d flag indicates the constituent as a dependents. The superscript on W indicates the composition function used.

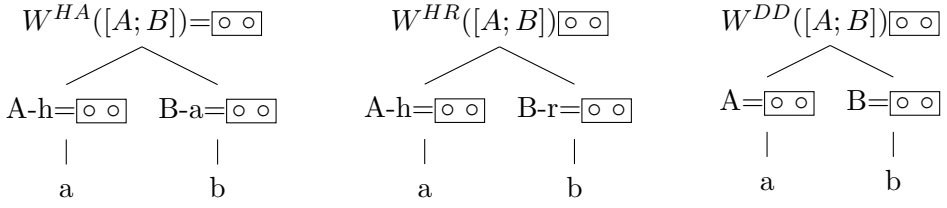


FIGURE 5.3: Composition function for the RNN-HeadArgumentAdjunct. The h flag indicates the constituent is the linguistic head. The r flag indicates the constituent is an argument. The a flag indicates the constituent is an adjunct. The superscript on W indicates the composition function used.

The third model (RNN-HeadArgumentAdjunct) expands upon RNN-Head by making a distinction between dependents that are arguments and those that are adjuncts. As such there are three types of constituents: heads, arguments and adjuncts, leading to three composition functions: Head-Argument, Head-Adjunct and dependent-dependent (figure 5.3). The dependent-dependent composition function could be broken down into three further composition functions: argument-adjunct, argument-argument and adjunct-adjunct. However, I choose not to take this approach, as it is nonsensical to compose two arguments within the Montagovian tradition. Instead, I predict that the composition function tries to keep as much information about both dependents as possible. Furthermore, by minimizing the number of composition functions the model is less computationally expensive.

While in English the head constituent is normally the leftmost child of the phrase, this is not always the case. The previously-explained models cannot account for this difference in position of the head constituents. This is problematic as the position of the constituents partially determines the output of the composition function. Consider

an alternative way to express the composition function previously detailed:

$$P(i,j) = f(W[c_i; c_j] + b) = f(w_1[c_i] + w_2[c_j] + b) \tag{5.1}$$

where $w_1, w_2 \in R^{N,N}$ and $W = [W_1; W_2]$. From this we see that the left and right constituent have their own weight matrix. As such it is justified to give a consistent matrix to each linguistic type. While one possible approach would be to simply associate one half size matrix (w_n) with each type, this does not fully capture the interaction of the two constituents. Instead, I change the concatenation order used as an input to the neural network. This makes the distinction between head concatenation order and temporal concatenation order. This gives the appearance to the Neural Network that the head constituent always appears as the left node (An example can be seen in figure 5.4). This allows for both items to be considered, as well as treating each linguistic type consistently.

$$P(i,j) = f(W[c_i; c_j] + b = f\left(W \begin{bmatrix} c_i^{Head} \\ c_j \end{bmatrix}\right) \tag{5.2}$$

The fourth model DRNN-Head has the same linguistic enrichment as RNN-Head, however, concatenation order head concatenation order. An example can be seen in table 5.1.

The fifth model DRNN-HeadArgumentAdjunct also is similar to RNN-HeadArgumentAdjunct such that is has three composition functions. Again I use head concatenation order.

The sixth model TRNN-Head model (table 5.1) makes a distinction between head and dependents. However, unlike the DRNN-Head it captures both head information and node position. Instead of encoding the head position using the concatenation order, I instead have two composition functions depending on whether the head is the left or right constituent. As such there are three composition functions, dependents-dependents, leftHead-dependent and rightHead-dependent. This approach could be seen as the closest approach to Socher et al. (2013), where each right side of a PCFG rule is used to determine the composition functions. In this case there are eight context-free grammar rules with four unique right sides.

- Head → Head dependents
- Head → dependents Head
- Head → dependents dependents
- Head → Head Head
- dependents → Head dependents

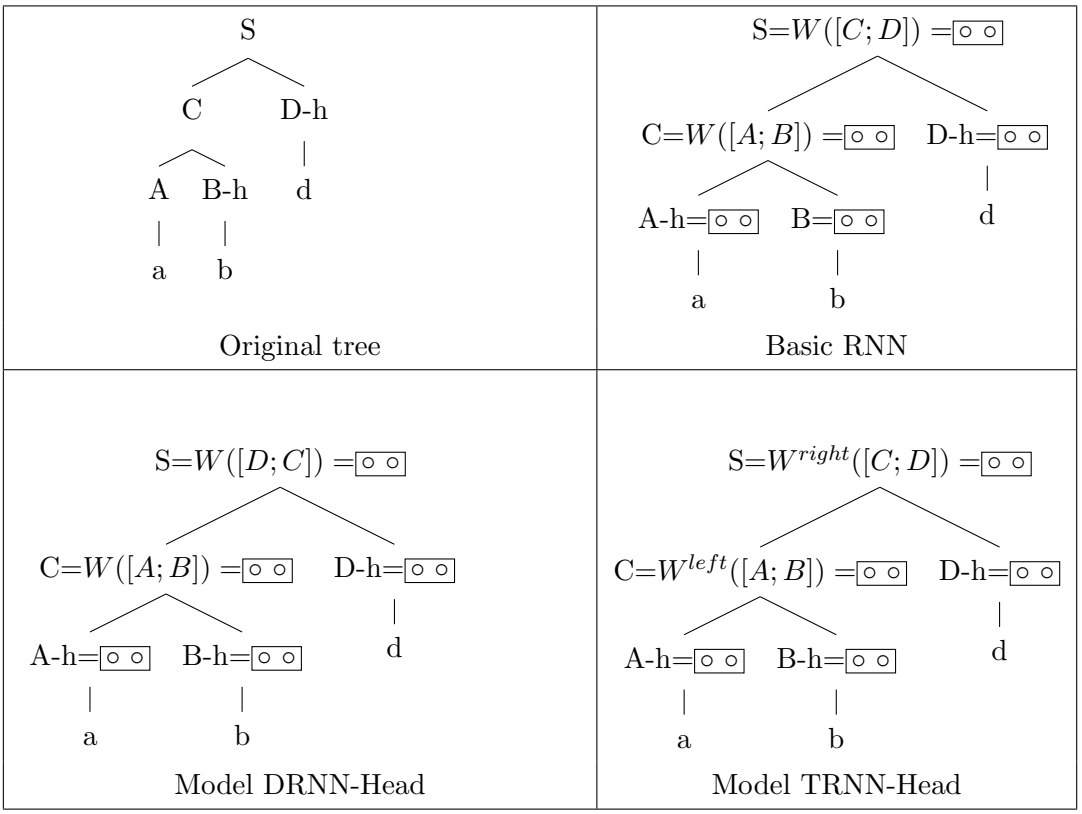


TABLE 5.1: Comparison of head dependent models. The h flag indicates the constituent is the linguistic head. the d flag indicates the constituent as a dependents. The superscript on W indicates the composition function used.

- dependents → dependents Head
- dependents → dependents dependents
- dependents → Head Head

The model uses just three composition types, as I assume that there exists just one head constituent per phrase.

Within table 5.2 I list an overview of all the models proposed, including information regarding the linguistic enrichment and encoding.

		Linguistic types		
		None	Head/Dependent	Head/Argument/Adjunct
Encoding	Position	BRNN	RNN-Head	RNN-HeadArgumentAdjunct
	Head	N/A	DRNN-Head	DRNN-HeadArgumentAdjunct
	Position + Head	N/A	TRNN-Head	

TABLE 5.2: Overview of all RNN based models

5.2.1 Reranking

The use of multiple composition functions increases the search space, making finding the highest scoring tree more time consuming and training more difficult. I optionally employ reranking to reduce the search space; the RNN model select the most likely tree according to equation 4.10 from a list of labelled syntactic trees as provided by an external symbolic parser. As such the semantic strengths of the RNN are combined with the speed of symbolic parsers. Reranking is particularly advantageous in the cases of binarization. When a phrase is binarized, there will be at least two newly-created subtrees. For these subtrees there will be a maximum of one head constituent. The RNN approaches which don't use reranking have no labelled information meaning they have no information regarding binarization and the limit to the number of head constituents, as such the search space is artificially high.

For all tests the generative Charniak (2000) parser was used and provided the top KBest parses. Due to the removal of syntactic tags many duplicates trees are provided which are then discarded; the effective KBest is thus variable. The Charniak parser was chosen as not only does it provide near state of the art performance in a resource-friendly manner, but it has been shown to benefit from discriminative re-ranking (Charniak and Johnson, 2005). I therefore combine the generative symbolic strengths of the Charniak parser with the discriminative semantic approach of the RNN. Beam search was optionally used in the same manner as section 5.1.4, to further restrict the search space explored. As beam search prunes on the cell level of the CYK algorithm, it is possible that no valid trees could be created. In this case a back-off mechanism is employed such that the search begins again with a wider beam. Socher et al. (2013) uses the log-likelihood of the corresponding PCFG rule. However, for connectionist reasons I do not include this information, as I wish minimize the influence a symbolic process has on the non symbolic model. Furthermore, the log-likelihood is likely to be of limited use due to the coarseness of the rules (see table 6.1).

5.2.2 Binarization

Due to the the limitations of the composition functions all trees need to be provided in binarized form. Binarization is of particular importance within this model, as the composition of $f([f([a;b]);c])$ is different to that of $f([a;f([b;c])])$. When two children are composed both representations become *squashed* within one final node. The original approaches to RNN based language models used right binarization. However, right binirzation often leads to the linguistically unjustified composition of dependents and dependents. I again draw from success within symbolic parsing approaches and propose

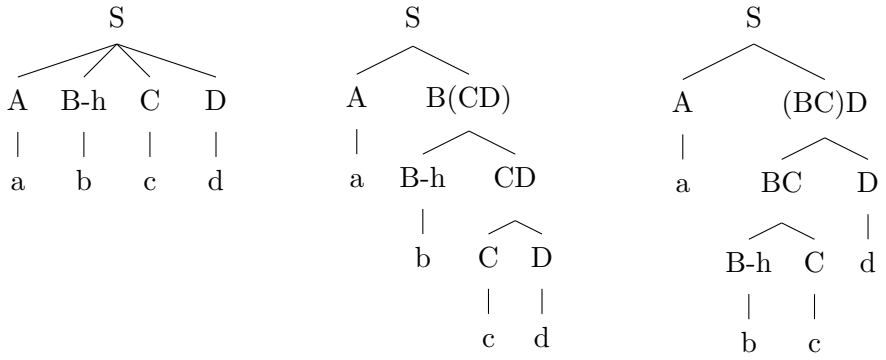


FIGURE 5.4: The leftmost tree is the original tree, which has been right binarized forming the middle tree and head outward binarized on the rightmost tree.

to use head outwards binarization. Head outwards binarization is used such that the head composes with the argument first (figure 5.4). Head-outwards binarization not only simplifies the model but also removes linguistic unjustified composition of dependents and dependents.

Chapter 6

Implementation and Evaluation

6.1 Introduction

In this chapter I will evaluate the syntactic and semantic performance of the RNN based models outlined in the previous chapter. As the RNN acts as a joint syntactic-semantic model the semantic-syntactic interface is implicitly accounted for and does not need to be explicitly tested. The evaluations will be both quantitative and qualitative in nature. A quantitative evaluation metric allows my models to be directly compared to each other and to previous semantic/syntactic approaches. Due to the possibility of over-fitting the quantitative task, a qualitative evaluation is also provided in the hopes of offering a further insight.

I use a parsing task to evaluate the syntactic understanding of my models. Within parsing tasks an utterance is given and the model computes its syntactic structure, which is then compared against the correct syntactic structure. Earlier I discussed that when learning the syntactic structure the RNN implicitly learns semantic information, but it is difficult to evaluate *how much* semantic information has been learnt. This motivates an evaluation on an explicit semantic task. In the past semantic role labelling, sentiment analysis and question answering have been popular approaches for semantic evaluation. However, it is hard to see them as *true* Montagovian semantic tasks rather than pragmatic or information retrieval tasks. I instead take inspiration from model theoretic formal semantics and go back to the issue of truth. Paraphrase detection tasks present a pair of utterances to be classified as paraphrases of each other or not. An utterance is a paraphrase of another utterance if both are semantically equivalent; they are true in the same possible worlds.

Within this chapter I first outline the parsing task and the details of my approach in addressing it. I next show and discuss the results my models achieved. I follow this by

outlining the semantic task as well as providing the results achieved. Finally, I provide a qualitative discussion of both the semantic and syntactic tasks.

6.2 Parsing

Parsing is the task of computing the syntactic structure of an utterance. To evaluate the parsing ability of a model, a corpus of sentences is defined. A corpus is normally divided into three sets, with each set performing a different function. The training set is used to learn the parameters (θ values) of the model. The development set is used to fine-tune the models' and set the models hyperparameters. The test set is then used to evaluate the model. At test time the utterances are given to the model but not their syntactic structures, which are computed and compared against the gold(correct) structures.

To evaluate how *good* the model computed trees are, an evaluation metric must be chosen. One tempting metric would be to use the percentage of computed trees that exactly match the gold trees. However, this would be too coarse, as trees which are *almost* correct would be marked just as incorrect as those which were completely incorrect. While there are several approaches to measuring degrees of correctness, I follow standard parsing convention and take the *span approach*. Within the span approach, a subtree is correct if its leftmost word and rightmost word match the gold trees leftmost and rightmost words. Precision, recall and their harmonic mean (F1) are used to measure the degree of correctness over all subtrees within the entire corpus. Precision is calculated using:

$$\frac{\text{(number of correct constituents)}}{\text{(number of constituents in the computed trees)}} \tag{6.1}$$

Recall:

$$\frac{\text{(number of correct constituents)}}{\text{(number of constituents in the gold trees)}} \tag{6.2}$$

And F1 is their harmonic mean:

$$2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6.3}$$

As the approach taken within this model requires binarization, recall and precision will be equivalent. However, in non-binarized approaches this would not be the case; the number of spans would be determined by the number of children for each subtree.

While the above metrics give an unlabelled score, they can be expanded by considering the labels on the tree. Within the labelled span approach a subtree is correct if the root label of the computed tree matches the root label of the gold tree and leftmost word and rightmost word match the gold tree's leftmost and rightmost words. Within symbolic approaches to parsing the label refers to the node's syntactic category. However, in the case of RNN the label at each node refers to which composition function was used.

While the span approach is popular and the one used in this thesis, it has a downside. The span approach artificially focuses on just one aspect of *incorrectness*. In contrast, an edit-distance-based approach counts the number of different operations required to transform the incorrect tree to the correct tree. Emms (2008) takes this approach developing the syntactic-edit-distance metrics: DICE and JACARD. Where the minimal sets D, I, S, and M containing the nodes that are deleted, inserted, swapped, or matched with the nodes from the gold tree G to the nodes of the computed parsed tree P. These sets are then used for the metrics, as follows:

$$DICE = 1 - \frac{|D| + |I| + |S|}{|G| + |P|} \tag{6.4}$$

$$JACARD = 1 - \frac{|D| + |I| + |S|}{|D| + |I| + |S| + |M|} \tag{6.5}$$

To evaluate my model using these metrics would have require changing the loss function; the margin of error is no longer based upon span error but on edit-distance error. This would require retraining the RNN model, an expensive prospect, hence not taken.

6.3 Setup

The corpus used for the parsing task is the Wall Street Journal Penn treebank II (WSJ) (Marcus et al., 1994), a dataset of syntactic annotation of English language journal articles. The WSJ was chosen because English is considered a morphologically poor language, thus allowing the focus to rest on the syntax and the semantics. While the WSJ is not considered representative of the English language as a whole; it is the most widely used corpus and therefore allows for comparisons to other approaches. For computational cost reasons I restrict the maximum sentence length to 15. Any sentences after preprocessing with greater than 15 words are discarded from both the test, development and training set.

Within the following subsections I will list specific details of the model for this task, including: initialisation approaches, pre-processing of the WSJ and hyperparameter selection.

6.3.1 Implementation

The code for this thesis is split into two main code bases, the model itself and preprocessing[1]. The code for the models is written in Python, based on the tree class found within Natural Language Toolkit (NLTK) (Bird et al., 2009). Pre-processing of the corpus was partially done with code supplied by Van Cranenburgh et al. (2011). Within the model code base the *multiprocessing* library was extensively used allowing the simultaneous use of multiple CPUs. Matrix multiplication is a large part of the algorithm, therefore a dedicated library (Ascher et al., 1999) was chosen.

External resources used include the Treep[2] annotator and the Charniak[3] parser. Treep (Chiang and Bikel (2002)) was used to annotate the trees with head argument adjunct information. The rules used for Treep are based, with slight modification, on Chiang and Bikel (2002) which is based on the work of Collins (1997). A list of these rules can be found in the appendix C. The Charniak parser is used to provide a KBest list of syntactic trees for an utterance within both the training and testing phase.

6.3.2 Pre-processing

As it is important to be able to compare my work to the work of others, I follow standard preprocessing techniques on the corpus. First, I remove all punctuation from the corpus, thus making the text corpus more similar to spoken language. Secondly, I remove traces and null elements from the WSJ. Traces indicate a non-local dependency, such as WH-movement. Null elements typically appear in places where an optional element are missing, for instance when *that* or *who* is missing: "the king said *null* he could go" or "the man *null* I saw". In the future it would be interesting to include these elements, as this would allow the composition to be more in line with linguistic theory (Gabbard et al., 2006).

A particularly problematic issue for the RNN based approach is *unknown words*; words that don't appear within the training set, thus have no vector representation. One possible solution would be to generate a random vector for each unknown word found within the test set. This however is problematic; unlike most approaches symbolic approaches to parsing RNN are not Markovian: the root vector is dependent on all vectors of the word, making the RNN approach particularly sensitive to all incorrect word representations. Furthermore, by generating the vector randomly, the parsing evaluation is no longer deterministic but instead the score will fluctuate dependent on

[1]Code can be found at https://github.com/agent1/RNN-Head
[2]Download link: http://www.isi.edu/ chiang/software/treep/treep.html
[3]Download link: https://github.com/BLLIP/bllip-parser

the random vector's values. I instead add an *unknown* vector to the lexicon: this vector is used and learnt in place of words that appear less than three times within the training set.

6.3.3 Initialisation

Initialisation can have a huge effect on the performance of many NLP models, due to the possibility of getting stuck in local optima. Therefore, care must be taken when initialising the three sets of parameters for the RNN, the W, W^{score} and the lexicon. For both W and W^{score} I use random initialisation; the values are drawn from a uniform distribution between $\frac{-1.0}{\sqrt{(vectorSize)}} \leq x \leq \frac{1.0}{\sqrt{(vectorSize)}}$. While the lexicon could also be initialised in such a manner, it has been shown to be beneficial to use data gathered using flat distributional semantics (Socher, 2014). I therefore use the lexicon from Collobert et al. (2011), who created word vectors using a Neural Network based language model. Since this lexicon was created without explicitly modelling composition I allow the RNN to change the vector value of words during training.

6.3.3.1 Baby steps

An alternative initialisation approach comes from unsupervised parsing, where it has been shown to be beneficial to learn the syntactic structure for *simple* utterances before training on complex cases (Spitkovsky et al., 2009). Due to the complexity of training a RNN I will transfer this approach to a supervised setting. Within Spitkovsky et al. (2009) simple sentences are *short* sentences, as there are fewer possible syntactic trees. Within a supervised setting, the number of possible tree is less problematic because the correct syntactic structure is known. Instead, I define simplicity with respect to the composition function, where the composition of certain linguistic types are simpler than others. I consider the head-argument composition a core semantic process, as every predicate and utterance must specify an argument but adjuncts are optional and do not appear within every utterance.

I will first pre-train on a *simplified* corpus which contains only those trees which do not contain adjuncts constituents. I will then use these weights to initialise a new model which trains over the entire unrestricted corpus. Due to the sensitivity of the weights the learning rate is lowered to ensure that the efforts of the pre-training is not undone.

6.3.4 Cross validation

To adjust the hyperparameters I use cross validation over the development set(WSJ section 22). In table 6.1 I show the hyperparameters I use; within appendix A the results of cross validation can be seen.

Name	Value	Description
K	0.1	Penalty applied to a wrong span
λ	0.0001	Regularization value
Beam size	∞, 1	Either unlimited (global search) or size one (greedy search)
N	50	The dimension of the word vectors
LR	0.5, 0.05	AdaGrad learning rate
M	500	batch size (formula 4.15)
KBest	200	The number of trees Charniak provides to be reranked

TABLE 6.1: RNN hyper-parameters

6.4 Results

6.4.1 Preliminary results

Although a reranking RNN setup achieves better results than the non reranked setup, I still give the results for the non reranking setup. By providing the non reranked results

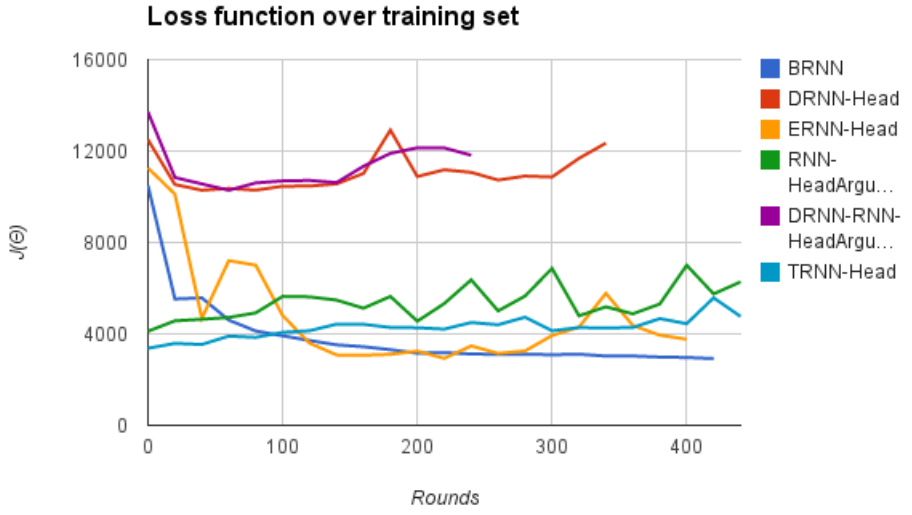


FIGURE 6.1: Loss function over number of training rounds for non-reranked RNN based models

I show a replication of the work of Socher et al. (2010) and provide a motivation for the switch to the reranking setup. Within the results table 6.2 I include in addition to the RNN models, two baselines: *PCFG* and *HAA-PCFG*. *PCFG* is a standard PCFG with no smoothing or vertical/horizontal markovization. *HAA-PCFG* uses a PCFG, however all syntactic categories are replaced with the linguistic types: head, argument or adjunct. The RNN models used for preliminary testing are trained and tested using a greedy search (a beam size of 1). The training rounds were limited due to the computational cost associated with training.

The results in table 6.2 surprisingly show us that the BRNN achieves the highest F1 score, followed by RNN-Head and RNN-HeadArgumentAdjunct. One possible reason that the linguistically-enriched models performed worse than the BRNN is their larger search space. A large search space makes finding the optimal parameters more difficult when training with a limited number of training rounds. Furthermore, a greedy approach may not be appropriate for the linguistically-enriched RNN models, as the head argument relationships are inherently not markovian. All trees must have at least one head element. A solution to both problems is to increase the number of training rounds and increase the beam size (possibly to include a global search). However, this is problematic as the computational cost of training the RNN is already high, leading to the premature termination of training in these cases. While all RNN based models achieved lower scores than the vanilla PCFG this is in part due to the low number of training rounds; the BRNN, with a significant increase in training rounds achieves an F1 score of 77.64, significantly better result than the PCFG.

The results show us that the approaches which use head dependent concatenation order (DRNN-Head and DRNN-HeadArgumentAdjunct) achieve significantly worse F1 scores than the corresponding temporal concatenation order models (RNN-Head and RNN-HeadArgumentAdjunct). While earlier I partially attribute poor performance to

Model	UF1
PCFG	69.79379
HAA-PCFG	22.20
BRNN	67.697266
DRNN-Head	35.03414
DRNN-HeadArgumentAdjunct	27.352049
RNN-Head	66.95751
RNN-HeadArgumentAdjunct	67.58346
TRNN-Head	63.03111

Table 6.2: Preliminary non-reranked results on the WSJ15

a low number of training rounds, figure 6.1 shows us that this is not likely in these cases. We see for both DRNN-Head and DRNN-HeadArgumentAdjunct the loss function did not significantly decrease over time, suggesting that more training rounds would not help the performance. Interestingly the TRNN-Head model which encodes both temporal and head information achieves worse results than the RNN-Head model which just encodes temporal information. In this case the loss function did significantly decrease over time; suggesting that more training rounds would be beneficial.

6.4.2 Results

Model	UF1
PCFG	69.79379
Stanford-PCFG	72.28791
Charniak	90.89176
Charniak-random	71.449
Socher et al. (2013)	91.1[4]
BRNN	81.27
RNN-Head	82.89
RNN-HeadArgumentAdjunct	86.04
DRNN-Head	79.86
DRNN-HeadArgumentAdjunct	82.89
TRNN-Head	78.23
BRNN-baby	86.22

TABLE 6.3: Results for RNN models using right binarization and baseline results

The preliminary non reranked results motivate a switch to a reranking setup, where the top 200 trees from Charniak are used. By reducing the search space a reranking setup allows for a global search (an unlimited beam size) and more training rounds, which improve upon the results of the non-reranked setup. As results of the reranking setup are reliant on a parser, two sets of baselines are given, using either a PCFG parser or the Charniak parser. The PCFG approach includes the the baseline listed

Model	UF1
BRNN	80.93
RNN-HeadArgumentAdjunct	81.93329
DRNN-Head	79.00
DRNN-HeadArgumentAdjunct	78.44792

TABLE 6.4: RNN models results when using head-outwards binarization

within the preliminary results; *PCFG*. Additionally, I add the baseline *Stanford-PCFG*, a PCFG parser which includes probability smoothing, vertical and horizontal markovization (v=3, h=1). The second set of baselines are added due to using the Charniak parser within the reranking setup. *Charniak* is a state of the art parser as proposed by Charniak et al. (1998). *Charniak-random*, randomly chooses one parse tree from the KBest candidate parse trees[5]. When just examining the baselines we see straight away from table 6.3 that *Charniak-random* gives a higher F1 score than both *PCFG* and *HAA-PCFG*, showing the strength of the Charniak parser.

The results seen within table 6.3 offer several insights into the RNN models. Firstly, we see that all the RNN models beat all the PCFG based baselines and the Charniak-random baseline. Scoring higher than the Charniak-random baseline shows that the RNN models are learning syntactic structures and not just scoring highly due to the reliance on the KBest list from Charniak. While the models beat the random baseline, no model reports a higher F1 score than *Charniak*; this is partially due to the strength of Charniak. The results are given for WSJ15; the Charniak F1 score for WSJ of unrestricted length is comparable, suggesting that the RNN based results should carry over to the entire non-length restricted WSJ.

A question this thesis tried to answer was whether a small number of composition functions can achieve comparable state of the art results of an RNN model using significantly more composition functions. From the results we see that none of the models achieved state of the art results of Socher et al. (2013). While both Socher et al. (2013) and *Charniak* show the importance of syntactic categories, we do see an increase in F1 score as linguistic enrichment is added. The results show that an RNN-Head, with two composition functions achieves a higher score than BRNN, with just one composition function. *RNN-HeadArgumentAdjunct* with three composition functions achieves significantly better results than the BRNN model.

One surprising result was that impact baby steps initialisation had on the BRNN model. The baby steps model BRNN-baby achieved significantly higher results than the random initialised BRNN model and the more linguistically enriched RNN-HeadArgumentAdjunct. This highlight not only how head, argument, adjuncts distinctions play an important role within the RNN framework and the importance of initialisation over model enrichments.

While several approaches taken within this thesis increased the performance of the model, there were less successful approaches. Head dependent concatenation gives worse results than standard concatenation order. We can see that DRNN-Head scores

[5]To select a random parse tree the weights within the network are randomly initialised then used to parse the test corpus, this is repeated and the mean F1 score is reported.

lower than RNN-Head and DRNN-HeadArgumentAdjunct scores lower than RNN-HeadArgumentAdjunct. Head dependent concatenation removes temporal encoding in order to encode head linguistic type information. The results suggest that this trade-off was not beneficial and that temporal encoding is more important than the location of the head constituent. The results from TRNN-Head surprisingly are worse than DRNN-Head and RNN-Head results, even though it encodes both word order and head order. This is surprising as RNN-Head can be seen as a special case of TRNN-Head. If the composition functions from TRNN-Head LeftHead-Depedent and RightHead-Depedent were forced to be equal this model would be equivalent to RNN-Head. Suggesting, that it is not the power of the TRNN-Head but the learning which gives the lower score. Where, the low score of TRNN-Head is possibly a result of the composition function Headright-Depedent having fewer training examples.

Other disappointing results were those of head-outward binarization, which gave significantly worse results across the board, as seen in table 6.4. This highlights how dramatic an effect pre-processing can have on the results as well as making binarization a key issue. While right binarization generally fits with the linguistic intuition that English is right branching, this however is not always the case[6] and care must be taken as to how binarization is performed. I partially attribute the poor performance to the RNN behaving as a compression algorithm rather than a composition function. Vectors which must go through multiple compositions becomes diluted by the time it reaches the root node. With head-outward binarization the head constituent now goes through multiple composition functions, which dilutes the constituent which is syntactically/semantically more *important*. This is not the case in Montagovian grammars, where the influence of the head constituent would not be diluted. While it could be argued that the loss is due to the small size of the vectors, which can not contain the semantics of an entire sentence, Socher (2014) experimented with vectors up to 100 in size and saw negligible difference with those of 50 dimensions. Within the qualitative discussion of the models I further explore the notion that the RNN behaves as a compression algorithm.

6.5 Semantics

To evaluate the semantic information learnt by the RNN models I use a paraphrase detection task. The Microsoft Research Paraphrase Corpus (MSRP), as introduced by Dolan et al. (2004), consists of pairs of sentences, which must be either classified as a paraphrase or not. The task is semantic in nature as the words and the structure of the

[6] Slavic languages are one such instance

two utterances can be different yet semantically equivalent. However, the task still incorporates a syntactic element; the syntactic structure for the utterances are not given. Instead, the model must compute the syntactic structure, so that the semantic vectors can be constructed. To focus on the semantic aspect of the task an external state of the art parser is used to calculate the most likely syntactic structure for the utterance. Charniak was chosen to compute the syntactic structures and from the parsing results within table 6.2 we see that Charniak scores higher than the RNN approaches. Furthermore, the RNN approaches have been only trained on utterances of length 15 and under whereas the MSRP contains utterances with many more words that this. The RNN models will then use this syntactic structure to generate the vectorial representation of the utterance.

The paraphrase detection task is supervised in nature; the training corpus consists 1,076 sentence pairs that are marked as either a paraphrase or not and a test set of 1,725 sentences which the model must label as a paraphrase or not. The task is made particularly difficult due to the care taken in constructing the corpus, where word choices were similar in each pair of utterances, such that paraphrases could not be detected using word unigrams alone. While it would be possible to learn the RNN θ parameters directly on the training set, I avoid this option for two reasons: (1) the limited number of sentences, making training difficult; (2) rather than determining how much semantic information the RNN model can capture I am interested in how much semantic information the RNN model has learnt from the parsing task. Instead, I use an off-the-shelf supervised linear support vector classifier (LSVC). I use two set of features as inspired by Blacoe and Lapata (2012). The first set of features is the concatenation of the root vector for both sentences. The second set of features consists of the cosine distance between the root vectors of the sentences, the length of both sentences and the unigram overlap between the two sentences, where, the unigram overlap is the the number of words which occur in the union of both sentences. By using a minimal number of features the focus of the evaluation can be on the root vectors.

In addition to the results of the LSVC models I will give the results of a baseline (AlwaysYes) which always predicts that the pair of sentences is a paraphrase. I also include the results from three approaches found within Blacoe and Lapata (2012) which use the same two sets of features[7]. The difference between my models and the Blacoe and Lapata (2012) is the choice of composition function. The first model $+$, uses vector addition to compose new items. The second model $\odot$, uses component-wise multiplicative vector composition. The third model, RAE uses a recursive auto encoder (RAE) as

[7]Note for the addition model instead of using the concatenation of the vectors as the input to the SVC, the vectors were substituted from one and other.

based on Socher et al. (2011). For completeness I give the results of several state of the art approaches.

6.5.1 Results

Model	F1	Accuracy	Parsing F1
AlwaysYes	79.9	66.5	
DRNN-HeadArgumentAdjunct	81.06	68.80	82.89
BRNN-baby	81.19	69.30	86.22
RNN-Head	81.01	69.94	82.89
DRNN-Head	81.20	70.30	79.86
TRNN-Head	81.37	70.87	78.23
RNN-HeadArgumentAdjunct	81.42	71.53	86.04
BRNN	81.91	72.36	81.27
RAE	81.28	70.26	
$\odot$	82.33	73.04	
$+$	82.16	73.51	
Mihalcea et al. (2006)	81.3	70.3	
Rus et al. (2008)	80.5	70.6	
Qiu et al. (2006)	81.6	72.0	
Islam et al. (2007)	81.3	72.6	
Blacoe and Lapata (2012)	82.3	73.0	
Socher et al. (2011)	83.6	76.8	
Madnani et al. (2012)	84.1	77.4	

TABLE 6.5: Paraphrase detection results.

The results from table 6.5 show that all the RNN models achieve a higher F1 and accuracy score than the baseline *AlwaysYes*. Due to the high F1 score of AlwaysYes, I pay particular attention to the accuracy of the model. The BRNN achieves the highest F1 and accuracy score beating several previous state of the art results. However, it still falls short of achieving current state of the art results. We see that the additive and multiplicative composition functions achieve higher F1 scores than the RNN based approaches, further reinforcing the notion that simpler composition functions are better for semantics. As RAE is considered to capture more semantic information than RNN approaches, it is surprising that the RNN based approaches achieved comparable results. Although, this could be due to the slight differences in setup between my approach and Blacoe and Lapata (2012).

In addition to evaluating the RNN based models against state of the art results, I wished to see if the Max-Margin parsing loss function was suitable for capturing semantic information. The results show that this is not the case; the RNN models show no strong correlation between syntactic performance and semantic performance. Furthermore the BRNN achieves better semantic results than BRNN-baby, even though they are equally expressive models, with BRNN-baby achieving better syntactic results.

6.6 Exploration

In addition to the results given within the semantic and syntactic tasks I will analyse the trees within the corpus and give an analysis of the composition functions used. I first focus on the composition functions found within RNN-HeadArgumentAdjunct, as this is the most linguistically rich model. An assumption of this thesis was that differently typed linguistic items compose differently. To test this assumption I compare the composition functions by measuring the Frobenius norm of the subtraction of the composition W matrices. Table 6.6 shows us that the two most similar composition functions are Dependent-Dependent and Head-Argument. This indicates that linguistic constituents which are dependents and heads are closer to each other than dependents and adjuncts; and adjuncts and heads. Therefore, a better split in composition functions may be Head-Adjunct and Other-Other, where other would be either head or argument types.

Previously I stated that the composition function of dependent-dependent would act more as a compression algorithm than composition function. To test this I measure the distance of the matrices to the identity matrix as well as comparing how similar the vectors composed to the matrix are to the mean of the addition of the two vectors. Both these metrics measure if one argument is more influential; if this is not the case I would consider the composition to be compression-like in nature. From the Frobenius norm within table 6.7 we see that dependent-dependent is in fact the closest to the identity matrix and those constituents composed by it are closest to the mean addition.

Composition function	Distance
Head-Adjunct	28.1150852527
Head-Argument	26.7589556579
Dependent-Dependent	23.9236758953

Table 6.6: Distance to identity matrix

Within appendix D I list the trees about which the RNN models have not captured correctly; tree which the RNN assigns a high score yet these trees receive a low F1

Composition function	Head-Argument	Head-Adjunct	Dependent-Dependent
Head-Argument	0	35.0250278764	32.1754451601
Head-Adjunct	35.0250278764	0	32.5776586789
Dependent-Dependent	32.1754451601	32.5776586789	0

TABLE 6.7: Distance between composition functions

score. These trees show several cases of topicalization not receiving the correct structure. Topicalization is an example of discontinuity, a linguistic phenomenon which is not capturable within a CFG. While the RNN is considered to be context sensitive, these results raise questions over this consideration. Alternatively it show that more care in training is needed to better capture context sensitive phenomena.

Chapter 7

Conclusion

7.1 Closing remarks

I have used a Recursive Neural Network to capture syntax and semantics of natural language. I started by replicating the work of Socher et al. (2010). I then expanded upon this approach enriching the model with core semantic syntactic types. This enrichment came in two forms: multiple composition functions and initialisation. Like Socher et al. (2013) I believe that different linguistic items compose differently and therefore merit their own composition functions. However, unlike Socher et al. (2013) I chose to explore a small number of both semantic and syntactic core types (head, argument and adjunct) resulting in a small number of composition functions. The second enrichment came in the form of initialisation. Where, these linguistic types are used to define a core composition function used within baby steps initialisation procedure. The results from evaluating both types of approaches shows the success of enrichment as well as raising several interesting questions, which I will now give.

The results from chapter six show that the core types chosen offer a suitable way to split linguistic constituents, where models with further refined constituents gave better results. Although the models failed to achieve state of the art results, it begs the question of whether it was due to the small number of composition functions or the choice of composition functions. To find the ideal choice of composition functions there are several approaches, one possible future approach would be to test several different linguistic theories as the inspiration for core compositional types. This however is time consuming and there is no guarantee that any linguistic theory would offer the optimal splitting of the composition functions. Instead, future work could be done using the data to come up with the splitting composition functions. Previously Petrov and Charniak (2011) has used hierarchical spiting to refine symbolic grammars, for instance NP categories are

split into NP_1 and NP_2 based on their occurrences within the treebank. It would be interesting in using this approach to select different composition functions.

The RNN based approach to the paraphrase tasks shows the weakness of both the loss function and the training itself, as there seems to be limited correlation between parsing results and semantic results. In the future it would be interesting to explore how different loss functions capture both syntax and semantics. Further difficulties for the paraphrase task surface from manually inspecting the MSRP corpus; we see that the sentences are not *truly* semantically equivalent. Instead, there are cases of ambiguous pronouns, inference and pragmatic world knowledge being required. As such I suggest extending the model with contextual knowledge, which would be better suited to deal with these phenomena. Contextual knowledge can be gained through the understanding of documents as a whole not via individual isolated utterances. The RNN model should be given document representation; this could be seen as an extension of the work done on distributional semantics within a compositional setup (Le and Mikolov, 2014). Within the Le and Zuidema (2014a) framework the root outside vector could be used to represent the context of previous utterances within the document.

While formal semantics heavily influenced the model and the evaluation, this influence can still be strengthened. Within Montague grammar we see the extensive uses of types, however no such type information is explicitly found within RNN. While it is possible that type information is implicitly encoded within the vectors it is difficult to determine whether this is actually the case. Instead, this implicit information could be strengthened at training time by adding type information to the gold tree. The RNN would then be trained to predict the correct logical type. Logical type information could also be explicitly encoded into the model in a similar way as the tensor approach, where the type information represents the dimensionality of the vector.

In conclusion I have shown how the RNN can be used to model both syntax and semantics. I show that further linguistic enrichment is justified and improves the results on syntactic tasks. Although, the approach fails to achieve state of the art results, it gives improvements over the original RNN model and raised several questions for further study.

Appendix A

Cross validation

Below I list some key cross validation results. I start with Figure A.1, which shows the affect of regularization values on loss function on the development set. The results show that a high regularization value leads an increase in the loss function. Next I examine two different sources of vectors Turian et al. (2010) and Collobert et al. (2011). The results show that Collobert et al. (2011) gives better performance. Furthermore, using vectors of size 50 was not significantly slower than vectors of size 25. Table A.1 shows the results of a reranking setup using 100 trees to rerank. The results again show that head concatenation offers worse results than temporal concatenation. However, the results between levels of linguistic enrichment is less clear. This could be due to the small number of training examples the models are given.

Model	F1 score
BRNN	84.049774
DRNN-Head	81.16
RNN-Head	83.85
RNN-HeadAdjunct	83.58
DRNN-HeadAdjunct	81.01
TRNN	79.49

Table A.1: F1 scores for reranking setup using top 100 parse trees

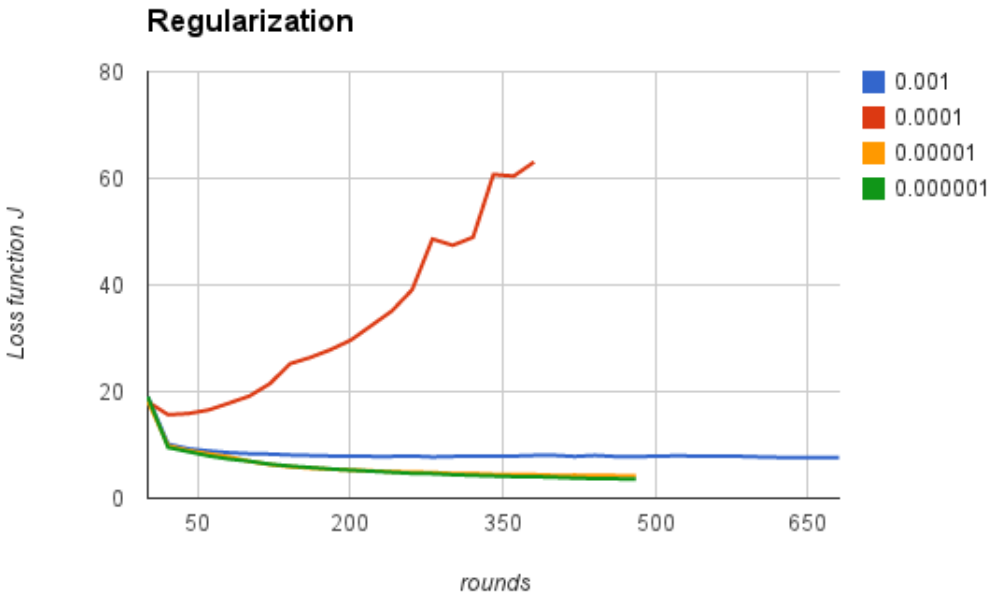


FIGURE A.1: Cross validation for Regularization values

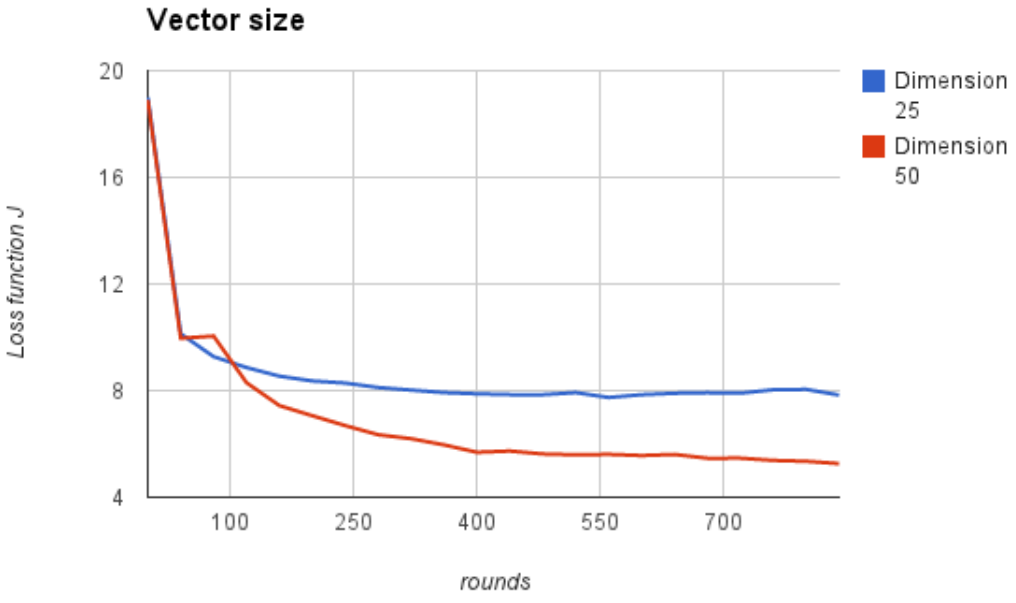


FIGURE A.2: Cross validation for vector size

Appendix B

Overview of alternative RNN models

In this section I briefly outline alternative approaches to capture syntax and semantics based on the RNN model. I start first by discussing the modifications suggested within the original RNN Socher et al. (2010). I then introduce Recursive Matrix-Vector Spaces and finally provide details of the inside outside approach to RNNs.

B.1 Context-aware RNN

Socher et al. (2010) proposed a contextual element be added to the RNN. in addition to the two constituents being composed, contextual words are added to the composition function. The new composition is now defined using:

$$p = f(W[c_{i-1}; c_i; c_j; c_{j+1}] + b) \tag{B.1}$$

Where a context of word to the left and the right are appended to the vector. An example can be seen in figure B.1.

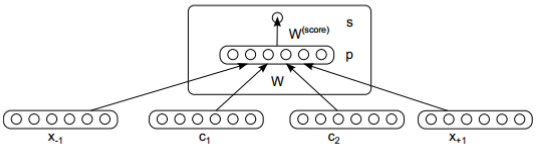


FIGURE B.1: Context-aware RNN

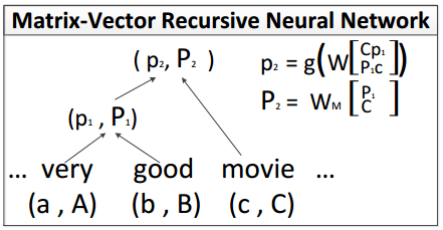


FIGURE B.2: MV-RNN

B.2 Category Classifier

Further modifications to Socher et al. (2010) including changing the loss function from the Max-Margin framework to predicting the syntactic category from the gold tree. The approach becomes generative; a softmax layer is added to each node on the gold tree; the objective functions minimizing the cross-entropy error of this softmax layer. The error will backpropagates and influence the θ values.

B.3 Semantic Constitutionality through Recursive Matrix-Vector Spaces

In contrast to the standard approach with RNN where words are represent by a vector within MV-RNN, words are represented by both a vector and a matrix. The vector represents the semantic of the word and the matrix the combinatorial procedure. An example can be seen in figure b.2. The composition models for two words is defined using the following:

$$p = F_{A,B}(a, b) = f(Ba, Ab) = f\left(W \begin{bmatrix} Ba \\ Ab \end{bmatrix}\right) \tag{B.2}$$

where B and A are matrices. For all internal nodes a single W_m matrix is learnt.

B.4 Inside Outside

An alternative approach to context was developed by Le and Zuidema (2014a); every node in the tree is represented by two vectors an inside vector like the standard RNN and an outside vector representing the context of the node, which is caculated in a top down manner. An example can be seen in figure B.3

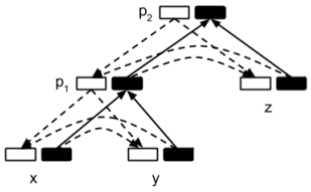


FIGURE B.3: Inside outside, diagram as adapted by Le and Zuidema (2014a)

This results in two composition functions: the inside representation is encoded within formula B.3

$$f(W_1C_1 + W_2 + C_2 + b) \tag{B.3}$$

and the outside representation is encoded within formula B.4

$$f(W_1O_1 + W_2 + O_2 + b) \tag{B.4}$$

Appendix C

Collins rules

Below I include the Treep rule based Chiang and Bikel (2002) which was used to add head annotation to a syntactic tree:

```
punc := ".",",",":";
pos := CC,CD,DT,EX,FW,IN,JJ,JJR,JJS,LS,MD,
       NN,NNS,NNP,NNPS,PDT,POS,PRP,"PRP$",RB,RBR,
       RBS,RP,SYM,TO,UH,VB,VBD,VBG,VBN,VBP,
       VBZ,WDT,WP,"WP$",WRB,AUX,AUXG,
       punc,"$","#","-LRB-","-RRB-";
head := "-HD";

ADJP   -> .* < ((NNS|QP|NN|"$"|ADVP|JJ|VBN|VBG|ADJP|JJR|
                 NP|JJS|DT|FW|RBR|RBS|SBAR|RB|.)+head
                > .*);

ADVP   -> (.* < (RB|RBR|RBS|FW|ADVP|TO|CD|JJR|JJ|IN|NP|JJS|NN|.)+head) > .*;

CONJP  -> (.* < (CC|RB|IN|.)+head) > .*;

FRAG   -> .* .+head;

INTJ   -> .+head .*;

LST    -> (.* < (LS|":"|.)+head) > .*;

NAC    -> .* < ((NN|NNS|NNP|NNPS|NP|NAC|EX|"$"|CD|QP|
                 PRP|VBG|JJ|JJS|JJR|ADJP|FW|.)+head
                > .*);

# Special rule for NP
NP     -> .* POS+head
        | .* > (NN|NNP|NNPS|NNS|NX|POS|JJR)+head > .*
        | .* < NP+head < .*
        | .* > ("$"|ADJP|PRN)+head > .*
        | .* > CD+head > .*
        | .* > (JJ|JJS|RB|QP)+head > .*
        | .* .+head;
```

```
PP   -> (.* < (IN|TO|VBG|VBN|RP|FW|.)+head) > .*;

PRN    -> .+head .*;

PRT    -> (.* < (RP|.)+head) > .*;

QP     -> .* < (("$"|IN|NNS|NN|JJ|RB|DT|CD|NCD|QP|JJR|JJS|.)+head > .*);

RRC    -> (.* < (VP|NP|ADVP|ADJP|PP|.)+head) > .*;

S    -> .* < ((TO|IN|VP|S|SBAR|ADJP|UCP|NP|.)+head > .*);

SBAR -> .* < ((WHNP|WHPP|WHADVP|WHADJP|IN|DT|S|SQ|SINV|SBAR|FRAG|.)+head > .*);

# This rule is found in the thesis only
SBARQ  -> .* < ((SQ|S|SINV|SBARQ|FRAG|.)+head > .*);

SINV   -> .* < ((VBZ|VBD|VBP|VB|MD|VP|S|SINV|ADJP|NP|.)+head > .*);

SQ     -> .* < ((AUX|AUXG|VBZ|VBD|VBP|VB|MD|VP|SQ|.)+head > .*);

UCP    -> .* .+head;

VP   -> .* < ((TO|AUX|AUXG|VBD|VBN|MD|VBZ|VB|VBG|VBP|VP|ADJP|NN|NNS|NP|.)+head >
    ↪ .*);

WHADJP -> .* < ((CC|WRB|JJ|ADJP|.)+head > .*);

WHADVP -> (.* < (CC|WRB|.)+head) > .*;

WHNP   -> .* < ((WDT|WP|"WP$"|WHADJP|WHPP|WHNP|.)+head > .*);

WHPP   -> (.* < (IN|TO|FW|.)+head) > .*;

# These two are not specified in Collins' head table; Collins' parser
# defaults to choosing the leftmost child
NX,X -> .+head .*;

# Top node
TOP,"" -> .+head;

# Parts of speech
pos -> .;

# Deleted nodes
"‘‘","’’" -> .;

# Empty nodes
"-NONE-" -> .;
```

Once the head information is applied, argument adjunct distinctions can be made with the following treep rules:

```
punc := ".",",",":";
pos := CC,CD,DT,EX,FW,IN,JJ,JJR,JJS,LS,MD,
       NN,NNS,NNP,NNPS,PDT,POS,PRP,"PRP$",RB,RBR,
       RBS,RP,SYM,TO,UH,VB,VBD,VBG,VBN,VBP,
       VBZ,WDT,WP,"WP$",WRB,AUX,AUXG,
       punc,"$","#","-LRB-","-RRB-";
adj := "-ADV","-VOC","-BNF","-DIR","-EXT","-LOC","-MNR","-TMP","-CLR","-PRP";
head := "-HD";
arg := "-A";

# Coordination rules

# These rules take priority over the argument rules, because sisters
# of coordination constructions can never be arguments.

# Even a conjunction not immediately following the head is treated specially
#  (what if there is more than one?)

. -> .* head < .* < CC . .*
   | .* .+head < punc* < CC head-head .*;

# Argument rules

# If the parent and head have the same category, no arguments are
# marked.

# First child after head which is neither PRN nor a preterminal is the argument

PP -> .* < PP&head .*
    | .* < head .* < !pos&!PRN+arg .*;

# Look for arguments anywhere to the left or right

S -> .* < S&head .*
   | ((NP,SBAR,S)&!adj+arg|.)*;

SBAR -> .* < SBAR&head .*
   | (S&!adj+arg|.)>*;

VP -> .* < VP&head .*
   | ((NP,SBAR,S,VP)&!adj+arg|.)*;

# Catch-all rule

. -> .*;
```

Appendix D

Treebank sample

I list the trees about which the RNN models are confident about and yet these trees receive a low F1 score. I first list the gold tree then the RNN models computed tree. Where, the labels refer to the composition function used at each node.

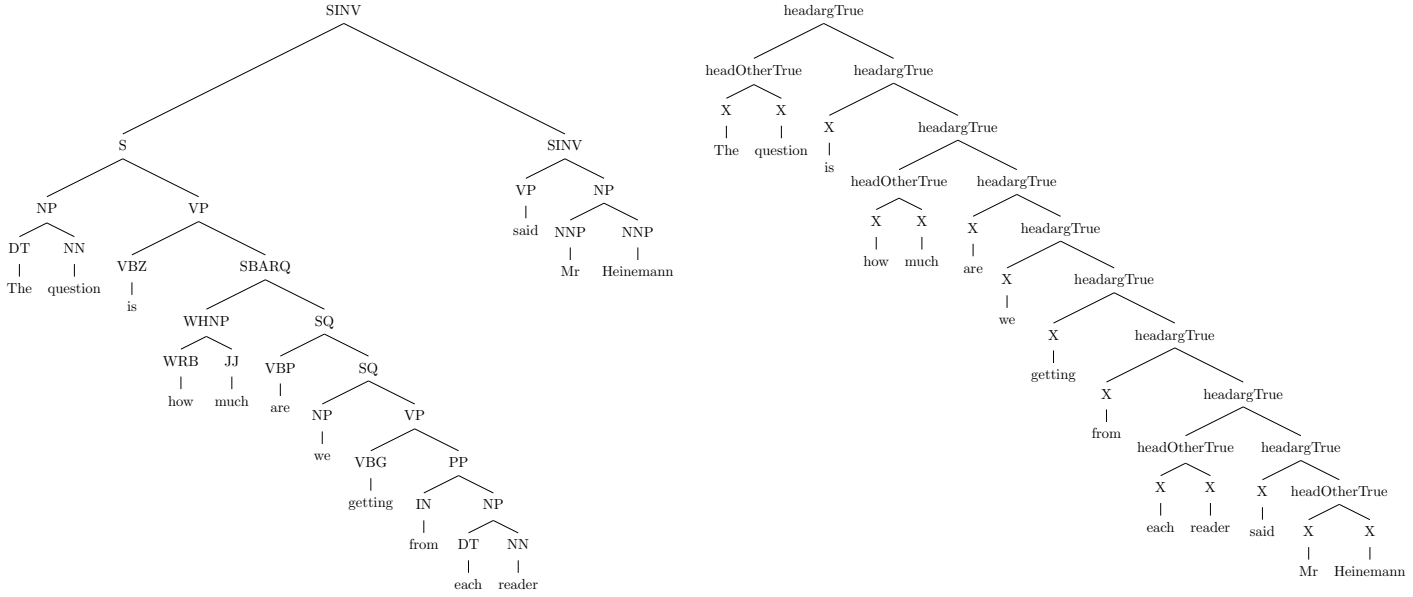


TABLE D.1: The question is how much are we getting from each reader said Mr Heinemann

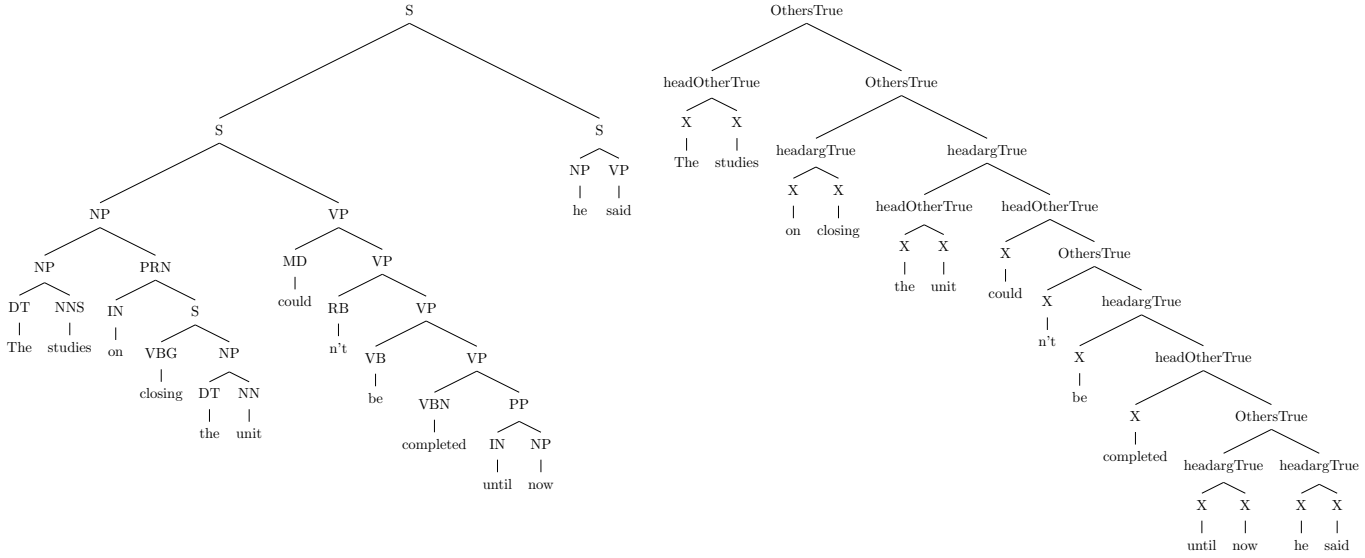


TABLE D.2: The studies on closing the unit couldnt be completed until now he said

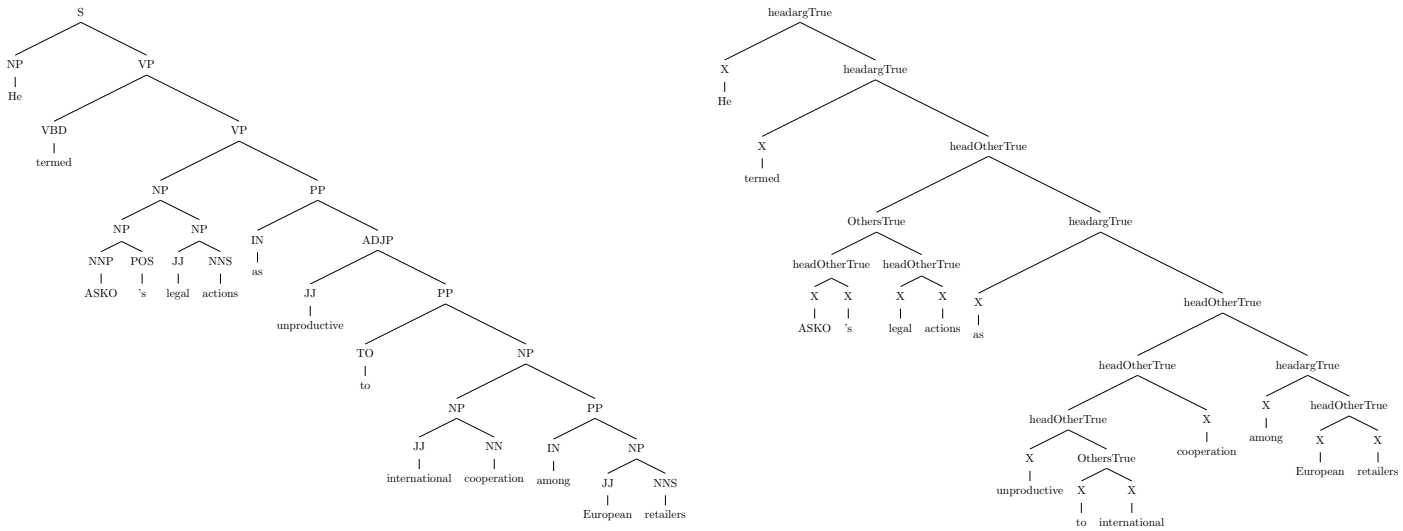
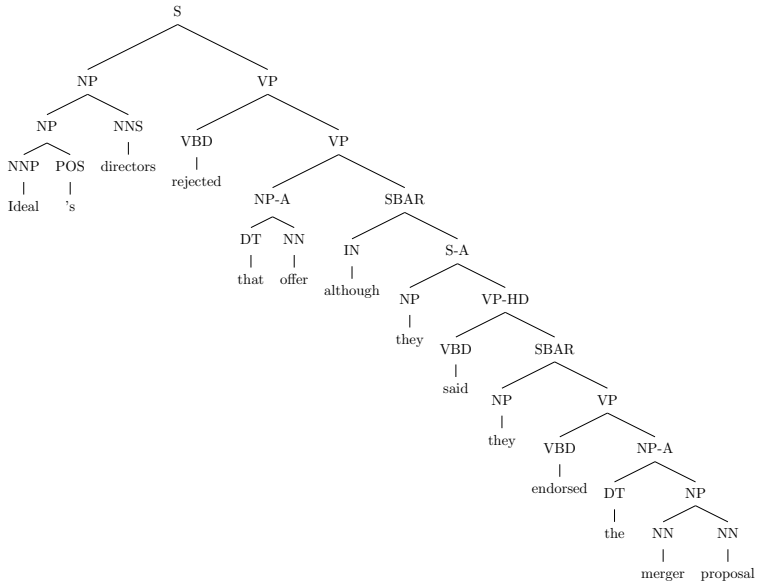


TABLE D.3: He termed legal actions unproductive to international cooperation among European retailers



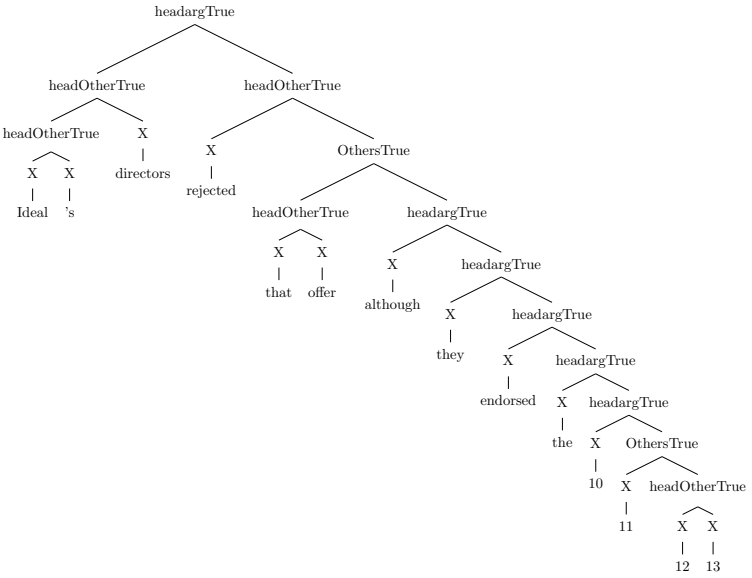
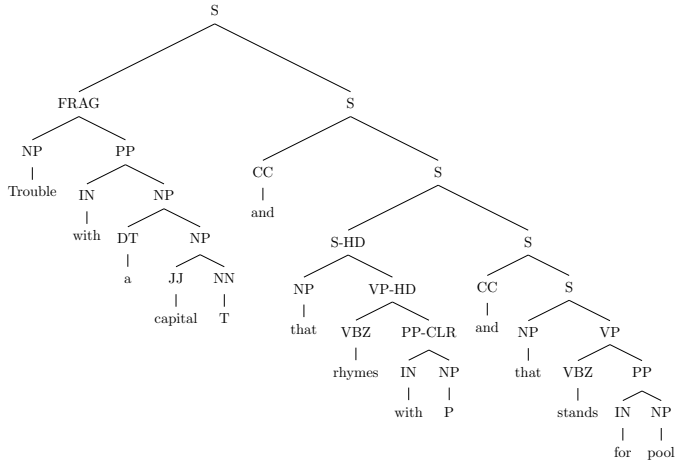


TABLE D.4: Ideals directors rejected that offer although they said they endorsed the merger proposal



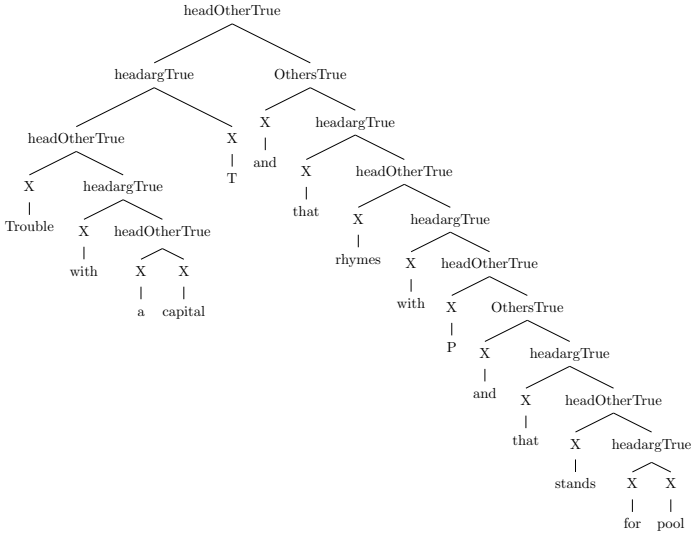


TABLE D.5: Trouble with a capital and that rhymes with p and that stands for pool

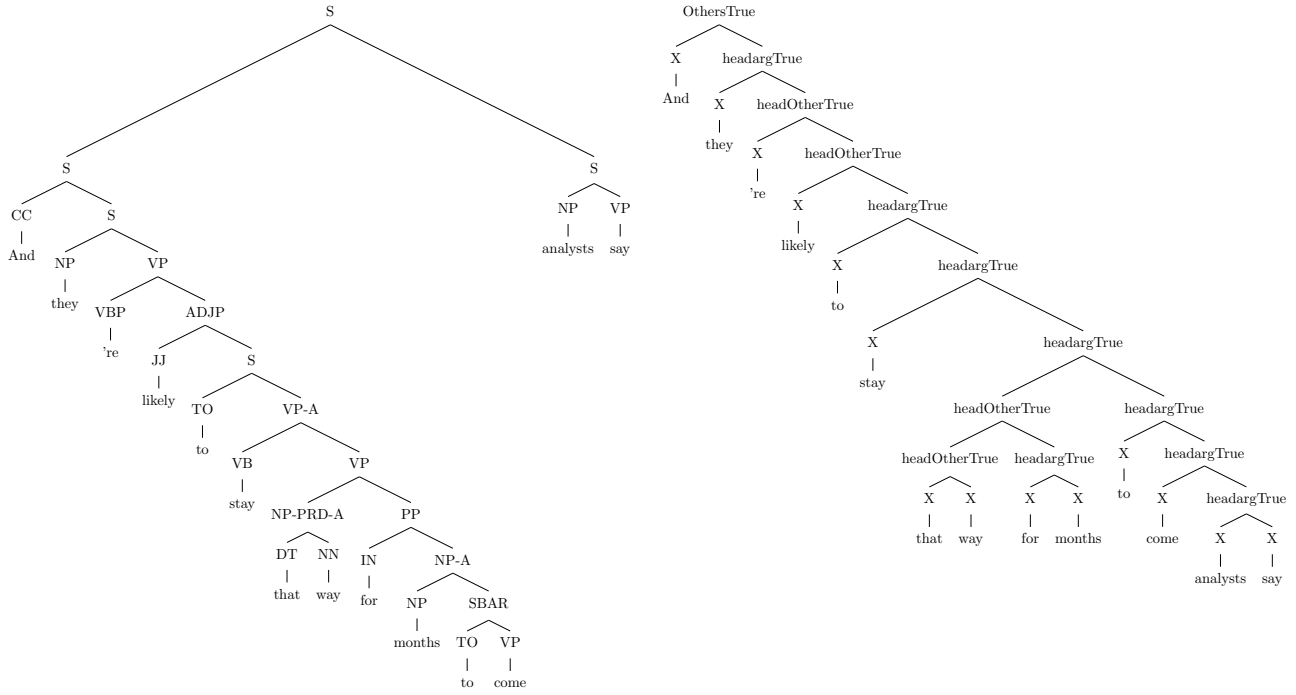


TABLE D.6: And theyre likely to stay that way for months to come analysts say

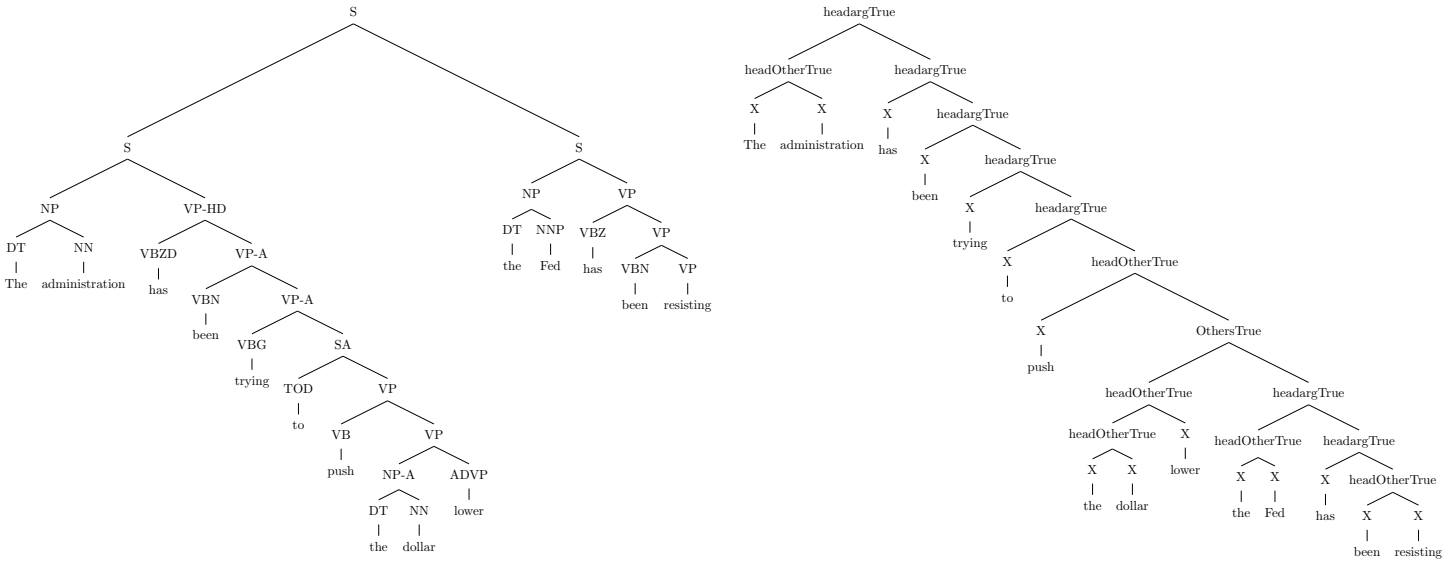


TABLE D.7: The administration has been try to push the dollar lower fed has been resisting

Bibliography

Alkhouli, T., A. Guta, and H. Ney (2014). Vector space models for phrase-based machine translation. *Syntax, Semantics and Structure in Statistical Translation*, 1.

Ascher, D., P. F. Dubois, K. Hinsen, J. Hugunin, and T. Oliphant (1999). *Numerical Python* (UCRL-MA-128569 ed.). Livermore, CA: Lawrence Livermore National Laboratory.

Baker, L. D. and A. K. McCallum (1998). Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 96–103. ACM.

Baldi, P. and G. Pollastri (2003). The principled design of large-scale recursive neural network architectures–dag-rnns and the protein structure prediction problem. *The Journal of Machine Learning Research 4*, 575–602.

Baroni, M., R. Bernardi, and R. Zamparelli (2014a). Frege in space: A program of compositional distributional semantics. *Linguistic Issues in Language Technology 9*.

Baroni, M., R. Bernardi, and R. Zamparelli (2014b). Frege in space: A program of compositional distributional semantics. *Linguistic Issues in Language Technology 9*.

Bird, S., E. Klein, and E. Loper (2009). *Natural language processing with Python*. " O'Reilly Media, Inc.".

Blacoe, W. and M. Lapata (2012). A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 546–556. Association for Computational Linguistics.

Boleda, G., S. Padó, and J. Utt (2012). Regular polysemy: A distributional model. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pp. 151–160. Association for Computational Linguistics.

Bruni, E., G. Boleda, M. Baroni, and N.-K. Tran (2012). Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pp. 136–145. Association for Computational Linguistics.

Budanitsky, A. and G. Hirst (2006). Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics 32*(1), 13–47.

Carlson, G. N. (1984). Thematic roles and their role in semantic interpretation. *Linguistics 22*(3), 259–280.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pp. 132–139. Association for Computational Linguistics.

Charniak, E. and G. Carroll (1994). Context-sensitive statistics for improved grammatical language models. In *AAAI*, pp. 728–733.

Charniak, E., S. Goldwater, and M. Johnson (1998). Edge-based best-first chart parsing. In *Proceedings of the sixth workshop on very large corpora*, pp. 127–133. Citeseer.

Charniak, E. and M. Johnson (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 173–180. Association for Computational Linguistics.

Chiang, D. and D. M. Bikel (2002). Recovering latent information in treebanks. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pp. 1–7. Association for Computational Linguistics.

Chiang, D., K. Knight, and W. Wang (2009). 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 218–226. Association for Computational Linguistics.

Choi, J. D. and M. Palmer (2011). Transition-based semantic role labeling using predicate argument clustering. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pp. 37–45. Association for Computational Linguistics.

Chomsky, N. (1988). *Aspects of the Theory of Syntax*, Volume 11. MIT press.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 16–23. Association for Computational Linguistics.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research 12*, 2493–2537.

Corbett, G. G., N. M. Fraser, and S. McGlashan (1993). *Heads in grammatical theory*. Cambridge University Press.

Curran, J. R., S. Clark, and J. Bos (2007). Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pp. 33–36. Association for Computational Linguistics.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems 2*(4), 303–314.

Dolan, B., C. Quirk, and C. Brockett (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, pp. 350. Association for Computational Linguistics.

Dowty, D. (1991). Thematic proto-roles and argument selection. *language*, 547–619.

Dowty, D. R. (1979). *Word meaning and Montague grammar: The semantics of verbs and times in generative semantics and in Montague's PTQ*, Volume 7. Springer.

Duchi, J., E. Hazan, and Y. Singer (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research 12*, 2121–2159.

Elman, J. L. (1990). Finding structure in time. *Cognitive science 14*(2), 179–211.

Emms, M. (2008). Tree distance and some other variants of evalb. In *LREC*.

Evert, S. (2010). Distributional semantic models. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Los Angeles*.

Fellbaum, C. (1998). *WordNet*. Wiley Online Library.

Gabbard, R., M. Marcus, and S. Kulick (2006). Fully parsing the penn treebank. In *Proceedings of the main conference on human language technology conference of the North American chapter of the association of computational linguistics*, pp. 184–191. Association for Computational Linguistics.

Giménez, J. and L. Marquez (2004). Svmtool: A general pos tagger generator based on support vector machines. In *In Proceedings of the 4th International Conference on Language Resources and Evaluation*. Citeseer.

Harris, Z. S. (1954). Distributional structure. *Word*, 146–162.

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural networks 2*(5), 359–366.

Irsoy, O. and C. Cardie (2014). Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pp. 2096–2104.

Islam, M. A., D. Inkpen, and I. Kiringa (2007). A generalized approach to word segmentation using maximum length descending frequency and entropy rate. In *Computational Linguistics and Intelligent Text Processing*, pp. 175–185. Springer.

Jackendorff, R. (1977). X-bar-syntax: A study of phrase structure. *Linguistic Inquiry Monograph 2*.

Kay, P. (2005). Argument structure constructions and the argument-adjunct distinction. *Grammatical constructions: Back to the roots* (4), 71–98.

Klein, D. and C. D. Manning (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pp. 423–430. Association for Computational Linguistics.

Le, P. and W. Zuidema (2014a). The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 729–739.

Le, P. and W. Zuidema (2014b). Inside-outside semantics: A framework for neural models of semantic composition.

Le, Q. V. and T. Mikolov (2014). Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Levin, B. (1993). *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Lewis, M. and M. Steedman (2013). Combined distributional and logical semantics. *TACL 1*, 179–192.

Lin, D. and P. Pantel (2001). Discovery of inference rules for question-answering. *Natural Language Engineering 7*(04), 343–360.

Lund, K., C. Burgess, and R. A. Atchley (1995). Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th annual conference of the Cognitive Science Society*, Volume 17, pp. 660–665.

Madnani, N., J. Tetreault, and M. Chodorow (2012). Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 182–190. Association for Computational Linguistics.

Marcus, M., G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger (1994). The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pp. 114–119. Association for Computational Linguistics.

Massé, A. B., G. Chicoisne, Y. Gargouri, S. Harnad, O. Picard, and O. Marcotte (2008). How is meaning grounded in dictionary definitions? In *Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing*, pp. 17–24. Association for Computational Linguistics.

McClelland, J. L., M. M. Botvinick, D. C. Noelle, D. C. Plaut, T. T. Rogers, M. S. Seidenberg, and L. B. Smith (2010). Letting structure emerge: connectionist and dynamical systems approaches to cognition. *Trends in cognitive sciences 14*(8), 348–356.

Menzel, C. (2014). Possible worlds. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Winter 2014 ed.).

Mihalcea, R., C. Corley, and C. Strapparava (2006). Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, Volume 6, pp. 775–780.

Mitchell, J. and M. Lapata (2008). Vector-based models of semantic composition. In *ACL*, pp. 236–244. Citeseer.

Mitchell, J. and M. Lapata (2010). Composition in distributional models of semantics. *Cognitive science 34*(8), 1388–1429.

Montague, R. (1970). Universal grammar. *Theoria 36*(3), 373–398.

Nivre, J. (2005). Dependency grammar and dependency parsing. *MSI report 5133*(1959), 1–32.

Pantel, P. and D. Lin (2002). Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 613–619. ACM.

Petrov, S. and E. Charniak (2011). *Coarse-to-fine natural language processing*. Springer.

Pinker, S. (1999). *Words and rules: The ingredients of language.* Basic Books.

Ponzetto, S. P. and M. Strube (2006). Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pp. 192–199. Association for Computational Linguistics.

Pustejovsky, J. (1991). The generative lexicon. *Computational linguistics 17*(4), 409–441.

Qiu, L., M.-Y. Kan, and T.-S. Chua (2006). Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pp. 18–26. Association for Computational Linguistics.

Rapp, R. (2004). A freely available automatically generated thesaurus of related words. In *LREC*.

Ratliff, N. D., J. A. Bagnell, and M. A. Zinkevich (2007). (online) subgradient methods for structured prediction.

Rus, V., P. M. McCarthy, M. C. Lintean, D. S. McNamara, and A. C. Graesser (2008). Paraphrase identification with lexico-syntactic graph subsumption. In *FLAIRS conference*, pp. 201–206.

Sahlgren, M. (2008). The distributional hypothesis. *Italian Journal of Linguistics 20*(1), 33–54.

Šarić, F., G. Glavaš, M. Karan, J. Šnajder, and B. D. Bašić (2012). Takelab: Systems for measuring semantic text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pp. 441–448. Association for Computational Linguistics.

Schubert, L. (2014). Computational linguistics. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Spring 2014 ed.).

Socher, R. (2014). *RECURSIVE DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING AND COMPUTER VISION*. Ph. D. thesis, STANFORD UNIVERSITY.

Socher, R., J. Bauer, C. D. Manning, and A. Y. Ng (2013). Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Socher, R., Y. Bengio, and C. D. Manning (2012). Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012*, pp. 5–5. Association for Computational Linguistics.

Socher, R., E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pp. 801–809.

Socher, R., B. Huval, C. D. Manning, and A. Y. Ng (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1201–1211. Association for Computational Linguistics.

Socher, R., C. C. Lin, C. Manning, and A. Y. Ng (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 129–136.

Socher, R., C. D. Manning, and A. Y. Ng (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pp. 1–9.

Speaks, J. (2014). Theories of meaning. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Fall 2014 ed.).

Specia, L., S. K. Jauhar, and R. Mihalcea (2012). Semeval-2012 task 1: English lexical simplification. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pp. 347–355. Association for Computational Linguistics.

Spitkovsky, V. I., H. Alshawi, and D. Jurafsky (2009). Baby steps: How less is more in unsupervised dependency parsing. *NIPS: Grammar Induction, Representation of Language and Language Learning*, 1–10.

Surdeanu, M. and J. Turmo (2005). Semantic role labeling using complete syntactic analysis. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pp. 221–224. Association for Computational Linguistics.

Taskar, B., D. Klein, M. Collins, D. Koller, and C. D. Manning (2004). Max-margin parsing. In *EMNLP*, Volume 1, pp. 3. Citeseer.

Turian, J., L. Ratinov, and Y. Bengio (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the*

association for computational linguistics, pp. 384–394. Association for Computational Linguistics.

Twain, M. (1988). *Adventures of Huckleberry Finn*, Volume 8. Univ of California Press.

Van Cranenburgh, A., R. Scha, and F. Sangati (2011). Discontinuous data-oriented parsing: A mildly context-sensitive all-fragments grammar. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pp. 34–44. Association for Computational Linguistics.

Yamada, K. and K. Knight (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pp. 523–530. Association for Computational Linguistics.

Zeevat, H. (2014). *Language Production and Interpretation: Linguistics meets Cognition*. Brill.