Analysis and Prediction of Dutch-English Code-switching in Dutch Social Media Messages

MSc Thesis *(Afstudeerscriptie)*

written by

Nina Dongen
(born 01-02-1981 in Amsterdam)

under the supervision of **Dr. Raquel Fernández Rovira**, 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:	**Members of the Thesis Committee:**
24-02-2017	Dr. Paul Dekker
	Dr. Raquel Fernández Rovira
	Dr. Diego Marcheggiani
	Prof. dr. Ronald de Wolf (Chair)

Institute for Logic, Language and Computation

Abstract

Multi-lingual phenomena as code-switching disturb widely used language interpretation tools, while the demand for such tools is rising due to the expanding worldwide popularity of online applications. This study explores code-switching between the lexically strong related languages Dutch and English in Twitter messages. Contrary to similar studies on code-switching, the focus is centred on the occurrence of English words in everyday Dutch, instead of a specific bilingual community. This research covers five main stages. First, a new Twitter corpus is collected of which a subset is manually annotated. Second, linguistic analysis of Dutch-English code-switching is performed. Third, several models are explored to perform a language identification task at word level. Fourth, several models are explored to perform automatic prediction of code-switching at word level. Finally, the best models for both tasks are combined and tested. Results show that multi-language data remain a challenge for computational approaches.

Acknowledgements

First, I would like to thank my supervisor Dr. Raquel Fernández Rovira, for her excellent guidance throughout the whole project. Our countless meetings and her insightful comments were extremely valuable. Besides, it was a pleasure to work with her.

Additionally, I would like to acknowledge the members of the committee Prof. dr. Ronald de Wolf, Dr. Paul Dekker, Dr. Diego Marcheggiani and Dr. Raquel Fernández Rovira, for taking the time to critically read the thesis and prepare the defence.

I would like to thank my friend Julian Jansen, the second annotator, for his patience and precise work. Also, I would like to thank my friends Thom van Gessel and Tom Schoonen, who provided many helpful comments in the last phase of my thesis.

Moreover, I would like to thank my parents and unofficial parents-in-law, for their mental and financial support.

Finally, special thanks go to Jesse Boom, the love of my life, who dedicated so much of his time to care for me. Without him, I would not have been able to carry out this project.

Contents

Chapter 1

Introduction

1.1 Motivation

Imagine the following: You are writing an informal message on your smartphone or tablet to a close friend. The message is in your mother tongue; the same holds for your friend. Within the sentence you are concocting, you decide to switch over to a second language, say English, mastered by the both of you; just because English seems to catch the intended meaning better in this particular case. But now, assuming you make use of automatic correction as provided by the used program, the English word is automatically changed into a word in the main language, a word most similar to the English original. Frustrated, you correct the word back to English or even disable the automatic correction function altogether.

This frustration, often recognized by users of social media, is the main instigator of this study. How can it be that in the "smartphone era" mixing two languages in one sentence poses such a problem? Why does a program not automatically recognize the language switch? The described inconvenience appears to be the result of a much larger language problem. Let us specify the issue by looking at the background more closely.

The phenomenon of mixed language within the same text or conversation is known as *code-switching*[1] and sometimes abbreviated here as CS. In the late 70s code-switching was picked up to study by a large group of sociolinguists, for example pioneers as Lipski (1978) or Poplack (1980); and more recently Romaine (1995), Myers-Scotton (1997) and Broersma (2009). Based on their findings we can now discern three kinds of CS: firstly, a language switch may occur between sentences at an inter-sentential level (Example 1a); secondly, a language switch may be encountered within a sentence at an intra-sentential level (Example 1b); and finally, a switch may take place within a word at the level of morphemes (Example 1c).

[1] Several terms are used such as code-mixing, language-mixing or language-switching; there is no consensus on the terminology.

(1) a. Yesssss, eindelijk de #seizoensfinale van #Familie! **Let's kill June** @name1 @name2
Yesssss, finally the #seasonfinale of #Family! Let's kill June @name1 @name2
b. @name1 beschouwend en in the **line of fire**?
@name1 considered and in the line of fire?
c. Er wordt weer lustig er op los **geframed** door de NOS over #brexit.
Again at the NOS they are freely framing about #brexit.

Although CS in general is quite simple to describe, studying it is actually rather complicated. We should realize that in theory every language could be mixed with every other language. Therefore, in order to study the matter, scientists pick out one or a few language combinations. This study will focus on the combination of English and Dutch.

Recently the topic is also studied by computational linguists (e.g. Solorio & Liu (2008), Das & Gambäck (2015) and Papalexakis et al. (2014)), for code-switching poses to be problematic for natural language processing (NLP). The reason for this is that many language processing tools assume a monolingual input text. Without proper language identification and prediction these language technologies will not perform well or even fail (Nguyen et al., 2015). CS therefore disturbs widely used language processing techniques such as sentence parsing, machine translation, automatic speech recognition and so on. With the expanding presence of digital possibilities this problem becomes more pressing.

Recognition and prediction of CS is closely intertwined with language identification. Without identification of the used language(s) it is impossible to recognize the occurrence of code-switches. To this date, document size matters when language identification is concerned. Automatic language recognition on a document scale is quite reliable and the same holds for longer sentences (i.e. >16 words). Grefenstette (1995) showed that a close to perfect accuracy can be reached in the classification of news articles with a simple model either based on most common words or trigrams in a language. But, language identification on a smaller scale still poses a challenge, especially the classification at the level of individual words.

Code-switching is typically found in informal conversation (Broersma, 2009), such as on social media. But, language use at social media is noisy; additives such as emoticons, abbreviations, exclamations, clerical errors etc. add to the complexity of the data. As a result, messages on social media are known to pose extra complexity to automatic interpretation. Though, with the increasing worldwide popularity of social media applications, the demand for better interpretation tools is rising.

1.2 Goals

The main goal of this thesis is to analyse, identify and predict CSs from and to English in everyday Dutch as used on social media. I will analyse Dutch-English

CSs as they appear in everyday Dutch language, on Twitter. My hypothesis is that Dutch-English code-switches occur on a regular basis within ordinary Dutch. However, the percentage of words involved may be low.

In addition to this hypothesis, I have five aims to accomplish my final goal. First, I want to collect a Twitter corpus, to study Dutch-English CS in everyday Dutch. Second, I want to analyse the encountered CSs. Third, I want to automate language identification of English in a predominantly Dutch corpus. Fourth, I want to automate CS prediction of English-Dutch CSs in a predominantly Dutch corpus. Finally, I want to combine automatic language identification with CS prediction, in order to automate CS prediction from start to end.

1.3 Related work

1.3.1 Linguistics on CS

In an early study Lipski (1978) studied the occurrence of CS in a bilingual English-Spanish corpus. He found that the swapping between languages seems to be restricted by the syntactical structures of languages involved. He hypothesised that prior to a switch the involved languages may contain divergent elements, but should be syntactically identical after a switch. However, not all encountered code-switches follow this constraint perfectly: post-switch sections of the sentence may not be fully congruent in the concerned languages. Nevertheless, Lipsky shows that incompatible syntactic structures are usually rejected as nonsense when presented to native speakers in an experimental setting.

The support of CS constraints is shared with other researchers. For example, Joshi (1982), who developed a formal framework in order to model code-switches. His paper shows that a large number of the constraints can be derived from one general constraint concerned with non-switchability of closed class items. In short, closed classes are grammatical word classes with a limited amount of members; new items are added seldom. In both English and Dutch closed classes include pronouns, determiners, conjunctions, preposition and auxiliary verbs. In contrast, there are open classes, which are mostly large in size and usually allow the addition of new items; think of nouns, verbs minus auxiliary verbs, adjectives, adverbs and interjections. According to Joshi, closed class items are generally not switched between languages, only items that belong to open classes.

In this study it is similarly assumed that code-switching is subject to constraints. Nevertheless, there are researchers who reject this idea, such as Thomason (2001). She claims that in theory, given the appropriate social conditions, no linguistic constraints exist and any linguistic feature can be transferred to any language.

Poplack (1980) is another proponent of the existence of constraints on code-switching. She concluded, after studying an English-Spanish dataset collected from a bilingual community, that a code-switch happens when the grammati-

cal structures of the first and second language overlap. Or, as she would call it, when the *equivalence constraint* is respected. She suggests that the equivalence constraint may be used to measure the degree of bilingual ability. First, she discerns two extremes: on the one hand 'risky' complex intra-sentential code-switches or *Intimate CS*, on the other hand less complex code-switches, characterized by relatively many tag[2] and single noun switches plus many inter-sentential switches, or *Emblematic CS*. Poplack's results indicate that non-fluent bilinguals do not violate the equivalence constraint by use of ungrammatical combinations, as might be expected. Instead, they make sure to avoid tricky intimate switch points. On the other hand, fluent bilingual speakers do use intimate code-switches. Moreover, the fact that even non-fluent bilingual speakers do not violate the equivalence constraints, strengthens the theory that such constraints do exist.

1.3.2 Research on Dutch CS

There are several studies about CS in which Dutch plays a role (Broersma (2009), Nguyen & Doğruöz (2013), Papalexakis et al. (2014), Yılmaz et al. (2016)). One of these, by Broersma (2009), is particularly interesting; not only because the study is specified at Dutch-English code-switching, but also because it tries to explain the mechanism behind unconscious code-switching. She finds that in natural speech of a Dutch-English bilingual, CSs occur more frequently when adjoined to a *trigger word*. In this setting, trigger words are cognates, i.e. members of a pair of words that have a common etymological origin. Examples of trigger pairs are: *ik - I*, *hij - he*, *was - was*, *goed - good* and *denk - think*. Since English and Dutch are both West Germanic languages, they are lexically strongly related and therefore share many trigger words. To overcome the difficulty of identifying the triggers, a total of six human judges manually annotated the corpus. The data not only shows that trigger words correlate with code-switching, but also that CS occurs more often between strongly related languages as Dutch and English compared to less related languages as Moroccan Arabic and Dutch.

1.3.3 Automatic language identification

The majority of tools currently developed in NLP are directed at monolingual texts. Automatic identification of language is usually the first step to deal with multiple languages in a system (Nguyen et al., 2015). Early studies on language identification were foremost directed at the recognition of a language on document level (Baldwin & Lui, 2010). By now, a number of more fine-grained approaches have been studied, at both sentence (Elfardy & Diab, 2013) and word level (Nguyen & Doğruöz, 2013; Das & Gambäck, 2015).

[2]By a tag Poplack means a filler or a tag question. A filler is a signal word indicating the utterer may pause but does not finish her turn yet, for example "I mean" or "you know". A tag question is an interrogative fragment such as "right?" or "isn't it?".

Nguyen & Doğruöz (2013) focus on automatic language identification on word level in Turkish-Dutch bilingual online communication. They take language identification as a classification task with the labels Dutch and Turkish. As baseline Nguyen & Doğruöz use an off-the-shelf tool meant for language identification on document level. For their main approach they use (combinations of) language models and dictionaries to tag word language. Later they improve their strategy by inclusion of context features based on the surrounding tokens, by use of logistic regression, and conditional random fields (CRF). For evaluation they look at performance of language identification on both word level and post level. Their results show that the off-the-shelf baseline does not perform well. Their best model, the CRF, makes use of a language model combined with context features.

Das & Gambäck (2015) study the characteristics of code-mixing in social media (Facebook). They present a system to automatically detect language boundaries in mixed English-Bengali and English-Hindi messages. Their main focus is on intra-sentential word level language identification. They use Support Vector Machines (SVM) to classify the words. As baseline they used a simple dictionary-based method. Performance of the best SVM system reached F_1-scores of 75-80%.

1.3.4 Predicting code-switches

A further step in dealing with multiple languages in NLP is automatic prediction of CSs in a text. CS forecasting builds on automatic language recognition. Without such identification, it is not possible to train features to predict CSs on a larger data. Where automatic language identification predicts the language of an utterance, predicting code-switches involves forecasting the language of a word to come without having access to that word.

In (2008) Solorio & Liu were the first to predict code-switches at word level. They used a small English-Spanish bilingual spoken corpus that they transcribed and annotated. Their prediction of potential CS points in a sentence is based on both syntactic and lexical features. Two learning algorithms, i.e. Naive Bayes (NB) and Value Feature Interval (VFI), are tested using two criteria: 1) the combination of precision, recall and F_1-score; 2) manually rated naturalness of generated switches. Used features involve previous token, language id and various Part-of-Speech (PoS) tags for the previous word and the position of a previous word within a phrase (e.g. verb phrase). NB outperforms VFI in most of the tested feature configurations. The highest F_1-score of NB was 28%, VFI 24%; still far from what is required in a real-life setting. Also, when naturalness of generated CS sentences was tested (scale 1-5), NB (3.33) scored higher than VFI (2.50) (VFI scores even lower than random).

Papalexakis et al. (2014) predict code-switches in a large dataset (4.5 million posts) collected from an online Turkish-Dutch discussion forum. Their goal is to automatically predict CSs within a post at word level. Besides expected features based on language identification tags, they also involve features covering emoticons and multi-word expressions. They use a system for automatic

language identification at word level created in a previous study (Nguyen & Doğruöz (2013)) discussed above. Their best F_1-score is 78%, using only three language identification features.

Difference in performance between Solorio & Liu and Papalexakis et al. has two main reasons. First, Papalexakis et al. base their CS predictions on language identification, which assumes to have information about the whole message, while Solorio & Liu does not. This choice is made by Solorio & Liu to account for a real-time structure throughout the process. As a downside, performance of CS prediction is expected to be lower. Second, contrary to Solorio & Liu, Papalexakis et al. use a sampled set for testing and training in order to overcome the challenges of an imbalanced data set; which is likely to have a positive influence on the outcomes.

1.4 Contributions and overview

In this thesis I will study Dutch-English code-switches as they appear in Dutch social media. I do not use material from a specific bilingual community, but analyse the use of English within ordinary Dutch messages as found on Twitter (Dutch is taken to be the dominant language). To my knowledge this is the first study on Dutch-English CS within an environment that is not explicitly bilingual. This study contributes to the investigation if code-switching in the following ways:

- A large **corpus** of roughly 95,000 Dutch tweets was collected on Twitter and made freely available.[3] This includes a section of 1,300 tweets, which were **manually annotated** on word level (Chapter 2). In addition to the manually annotated corpus, a set of annotation guidelines is provided (Appendix A). Furthermore, a simple command-line **annotation tool** was developed. This tool is also made available,[4] together with a user manual (Appendix B).

- Dutch-English code-switches, as they appear in the annotated corpus, were analysed. The focus of **analysis** lies on two kinds of code-switching: intra-sentential code-switches and morphological code-switches. This analysis provides information for model development (Chapter 3).

- Several supervised machine learning models, were **developed for the task of CS prediction** on word level in a real-time setting. For the execution of this task, a total of 10 features was analysed and ranked. Training of the features was based on the manually annotated set (Chapter 4).

- Several models, both probabilistic and non-probabilistic, were **developed for the task of language identification** on word level in a real-time

[3] http://illc.uva.nl/∼raquel/data/CS_prediction.zip
[4] http://illc.uva.nl/∼raquel/data/CS_prediction.zip

environment. For the execution of this task, a total of 30 features was analysed and ranked (Chapter 5).

- The model performing best on CS prediction was **combined** with the model performing best on language identification. This combination enables to fully automate the total process of CS prediction from start to end (Chapter 5).

Chapter 2

Corpus development

2.1 Introduction

The goal of this thesis is to analyse, identify and predict CSs from and to English in everyday Dutch as used on social media. For these purposes a data set of Twitter messages is collected. Of this set a subset is manually annotated for analysis (Chapter 3) and to serve as gold standard for model development and testing (Chapters 4 and 5).

In Section 2, I explicate the data collection with detailed information on how the corpus was gathered and which information was filtered out. Moreover, I provide some basic statistics. Section 3 is about the annotated corpus. It shows the used annotation scheme, the statistics derived from the annotation process and information about inter-annotator agreement. Also there are some final remarks on how the collected data sets are used.

2.2 Data collection

I used the TwitterSearch data collecting toolkit provided by Koepp (2016). The library is available at Github[1] and makes use of the Twitter Search API.

Twitter messages were collected by search for Dutch tweets worldwide. These were all automatically tagged 'nl' by the Twitter search API. Automatic language identification by Twitter is crude in the sense that a message is identified with one language only. A tweet tagged as 'nl' contains a majority of Dutch words, or more specifically, has Dutch as its predominant language. As a result, tweets identified as Dutch may additionally contain words of other languages, such as English. Language mixing within tweets can only be identified with the development of a more fine-grained language identification tagger. A downside of collecting 'nl' messages only is that we potentially miss tweets tagged as 'en' (for English) that may include language mixing with Dutch; which ideally

[1] https://github.com/ckoepp/TwitterSearch/

should also be part of our dataset.[2]

Location was not taken as a mandatory condition since nowadays most users disable the function to provide information concerning their whereabouts (analysis of the data showed only 5% of the users do provide such information, so it would have taken much longer to collect a comparable sized corpus).

2.2.1 Collection

I collected the Twitter Corpus between June 28^{th} and July 4^{th} 2016 at several moments during the week. One (or more) keywords have to be provided in order to extract tweets: I decided to use one keyword per trial. All keywords are used frequently in Dutch. To my knowledge there is no frequency list available containing Dutch words most used on social media. Therefore, I chose keywords based on word frequency as announced by *Genootschap OnzeTaal*.[3] To confirm the selected keyword is indeed used often on social media, I decided it should yield a minimum of 2000 tweets over two trials.

The top 15 frequency list as provided by *Genootschap OnzeTaal* originally consists of the words *ja, dat, de, en, uh, ik, een, is, die, van, 't, maar, in, niet* and *je*. For practical purposes the original list was slightly adjusted in order to form a selection suitable for searching. Firstly, since *uh* has many different spellings (e.g. *eh, uhhh, uhm, etc.*) I decided to leave it out. Secondly, I changed *'t* as originally in the list to *het*, since more often used within Twitter messages. Thirdly, the word *dat* was omitted, because it did not yield the threshold of 2000 tweets. The final list of keywords can be found in Table 2.1, which also includes the amount of yielded tweets per word.

2.2.2 Filtering

Retweets are filtered out. However, this is not enough to remove all duplicate tweets. The main instigator of double tweets is the option for a user to create an automated message on sites as YouTube or Facebook. For example, one can share the liking of a video on YouTube through Twitter; this results in tweeting a standard sentence of the form "`STANDARD SENTENCE: URL` and `VIDEO TITLE`":

(1) Ik vind een @YouTube-video leuk: https://t.co/xxx Horrific Horses falls #1.
I like this @YouTube-video: https://t.co/xxx Horrific Horses falls #1.

Several variants of similar messages are in use. Since these standard sentences do not exemplify the personal language use of the user, these tweets were removed. Another reason is the use of multiple accounts posting the same message at

[2] Although it is unclear how Twitter actually performs automatic language identification, they did post an insightful article online on forming their gold standard at https://blog.twitter.com/2015/evaluating-language-identification-performance

[3] https://onzetaal.nl/taaladvies/advies/woordfrequentie

Keyword	English	Tweets
een	*a, an*	18302
ja	*yes*	16757
ik	*I*	12814
de	*the*	9566
die	*that*	9312
het	*the*	8533
je	*you*	8350
en	*and*	4516
van	*of*	4184
niet	*not*	3057
in	*in*	2865
is	*is*	2736
maar	*but*	2571

Table 2.1: Amount of tweets collected per keyword

once or one user tweeting the same text combined with different URLs. In these cases only the first occurrence is stored.

2.2.3 Statistics

The original unprocessed corpus consists of 100,000 tweets. After removing all duplicates, 95,126 tweets remain; from now on I will call this the Twitter Corpus. In Table 2.2 we can find some basic statistic information about this Twitter Corpus. First we notice that tweets are short, restricted by Twitter to a maximum of 140 characters; this results in an average length of roughly 14.5 tokens per tweet. Secondly, as expected, a part of the users posted more than one tweet in the time frame of tweet collection; therefore the number of users is lower than the volume of tweets.

TC Corpus	Tokens	Tokens p. tweet	Users	Tweets
Total	1374094	14.45	46090	95126

Table 2.2: Basic statistics Twitter Corpus

2.3 Annotated corpus

2.3.1 Annotation scheme

From the Twitter corpus a total of 1,300 tweets, a hundred per keyword, were extracted for manual annotation. This data set is called the Annotated Cor-

pus (AC). Every token (total 19,464) was assigned one of six different labels: *Dutch, English, Mixed, Social Media Term, Other* and *Unclear*. In the next paragraphs, every class is described shortly accompanied by one example (the label concerned is printed boldface). The exact rules for classification can be found in Appendix A.

Dutch (NL) All Dutch words, the majority, are tagged *Dutch* (NL); according to the digital word list provided by OpenTaal.[4] Special attention is given to formerly English words incorporated in Dutch. For example, the words 'chick', 'shoppen', 'happy' and 'pack' are all labelled Dutch.

(2) **Ik ben wakker en ik leef nog**
I'm awake and still alive

English (EN) English words are labelled *English* (EN) in accordance to the Hunspell dictionary.[5]

(3) ...die ijslandrs gooiden letterlijk met die bal egt **im cryin.....**
...those icelandics literally threw that ball really im cryin.....

Words that are both English and Dutch (e.g. 'man' or 'is'), are labelled according to their corresponding context language. If the context does not provide enough information, the annotator decides which label has to be chosen.

Dutch-English mixed word (MIX) The label of a Dutch-English mixed word encompasses a very specific group of words, namely a word containing a code-switch at the level of morphemes. These words do not occur in either English or Dutch dictionary.

(4) @name1 ja dat is zo kapot irritant, moet je de game weer **restarten** enzo
@name1 that's so incredibly irritating, restart the game again

Social media term (SMT) Under the label of *Social Media terms* or SMTs ranges a collection of words involving URLs, @names, #hashtags, emoticons (e.g. #-#, :s), emoji's (e.g. ☺),[6] words and abbreviations specifically used on social media (e.g. LOL, tbh, tweet) and onomatopoeia (e.g. hahaaaaa, pfff).

(5) **@name1 lol** ja. en zijn paard hoe heet ie weer.
@name1 lol yes. and his horse what's its name.

Other language (OTH) Words of any other language, besides Dutch or English, are labelled as *Other* (OTH).

[4] See http://www.opentaal.org/
[5] http://wordlist.aspell.net/dicts/
[6] An emoticon exist of punctuation, e.g. :-) an emoji is an actual picture, e.g. ☺.

(6) Strijdlied **du jour:** Aan de strijders:...
Battle song du jour: To all warriors:...

Unclear (UNC) Any word that does not seem to fall under any of the mentioned categories is labelled *Unclear*. In general this means that the word does not seem to be a term (often) used on social media and meaning and/or language is not clear.

(7) ...Net voor de start heeft Lotto-name1 nog eens de **v...** ...
...Just before start Lotto-name1 has again v.. ...

2.3.2 Basic statistics

The Annotated Corpus (AC) with a total of a 1,300 tweets consists of 19,464 tokens. In Table 2.3 we can see the distribution of labels (NL, EN, MIX, SMT, OTH and UNC as mentioned in the previous section) over the sets of tokens. We see that the absolute majority of 85.8% is labelled Dutch. The second largest group with a percentage of 11.5%, is annotated with the SMT label. Third runner up is the English tagged token, with 1.4%. The other labels MIX, OTH and UNC, all make up for less than 1 percent of the tokens.

AC	**Total**
NL	16699 (85.8%)
EN	280 (1.4%)
MIX	9 (< 0.1%)
SMT	2246 (11.5%)
OTH	131 (0.7%)
UNC	99 (0.5%)
Total	19464 (100%)

Table 2.3: Annotated Corpus: Tokens per label.

In Table 2.4 we can find the distribution of labels over tweets. Note that this time the given percentages do not add up to 100, since a twitter message might contain several labels. Again, we can see that at large the labels NL, SMT and EN appear the most. Roughly 99% of the tweets contains NL tokens and about 86% an SMT label. The third runner up, the English tokens, cover 8.5% of the Twitter messages.

The main focus in this thesis is on the occurrence of English words in Dutch Twitter messages. Based on the results in Table 2.4, we might say that the ratio of English to Dutch tokens, namely 280 (i.e. 1.4%) compared to a 16,699 (85.8%) respectively, is relatively low. On the other hand, in Table 2.4 we see that the percentage of tweets containing English words is notably larger, i.e. 8.5% in the total set of 1,300 tweets. Therefore we can conclude that while the ratio of English words within Dutch language as used on Twitter, is quite low (namely only 1.4%), the set of affected tweets is considerable (namely 8.5%).

AC	Total
NL	1285 (98.8%)
EN	110 (8.5%)
MIX	9 (0.7%)
SMT	1123 (86.4%)
OTH	21 (1.6%)
UNC	71 (5.5%)
Total	1300

Table 2.4: Annotated Corpus: Tweets per label

2.3.3 Inter-annotator agreement

Both annotators are Dutch natives and fluent in English. The first annotator annotated 1,300 tweets (i.e. the Annotated Corpus), the second annotated the first 100 tweets containing 1666 tokens. Calculation of inter-annotator agreement yielded a Cohen's kappa of 0.94. The corresponding confusion matrix is given in Table 2.5; here we can see that there is some small disagreement in labelling EN, NL and SMT, also we see quite some disagreement in the classification of Dutch and social media terms.

	NL	EN	MIX	SMT	OTH	UNC	Total
NL	**1447**	1	0	3	0	0	1451
EN	1	**17**	0	0	0	0	18
MIX	0	0	**0**	0	0	0	0
SMT	14	1	0	**179**	0	0	194
OTH	0	0	0	0	**0**	0	0
UNC	1	0	0	0	0	**2**	3
Total	1463	19	0	182	0	2	**1666**

Table 2.5: Confusion matrix of inter-annotated data

The disagreements in which an English label is involved can all be found in one tweet. This tweet is shown in (8), where (a) is the first annotator and (b) the second. For clarity all English labels are printed bold:

(8) a. ⟨SMT⟩ @name1 Sup ⟨/SMT⟩ ⟨NL⟩ man, ik ben hier voor de Trilogy ⟨/NL⟩ ⟨EN⟩ **RC.** ⟨/EN⟩ ⟨NL⟩ ik ga voor ⟨/NL⟩ ⟨EN⟩ **player** ⟨/EN⟩ ⟨NL⟩ en ⟨/NL⟩ ⟨EN⟩ **content creator.** ⟨/EN⟩ ⟨NL⟩ hoe zit het met de ⟨/NL⟩ ⟨EN⟩ **conten creator part?** ⟨/EN⟩

b. ⟨SMT⟩ @name1 ⟨/SMT⟩ ⟨EN⟩ **Sup man,** ⟨/EN⟩ ⟨NL⟩ ik ben hier voor de Trilogy RC. ik ga voor ⟨/NL⟩ ⟨EN⟩ **player** ⟨/EN⟩ ⟨NL⟩ en ⟨/NL⟩ ⟨EN⟩ **content creator.** ⟨/EN⟩ ⟨NL⟩ hoe zit het met de ⟨/NL⟩ ⟨EN⟩ **conten creator part?** ⟨/EN⟩
@name1 Sup man, I'm here for the Trilogy RC. I'll go for player and content creator. what's up with the conten creator part?

We can see that in (8a) "Sup" is classified SMT and "man" Dutch, while in (8b) "Sup man" is labelled English as a whole. So, in the first case "Sup" is taken to be an abbreviation for "What's up" used at social media and "man" interpreted as being Dutch. The second annotator took both "Sup" and "man" to be English. "Sup" does occur in the English dictionary, but as a verb meaning 'to drink or eat'; "man" can be either Dutch or English bearing a similar meaning. So, in this combination I the classification according to the first annotator is preferred.

Another difference that can be found is that in (8a) "RC" is classified as English, but in (8b) as Dutch. "RC" can be found in the English dictionary, but then it means 'Roman Catholic'; it is not feasible that this should be the intended meaning, though it is not totally clear what the intended meaning exactly is. The interpretation of (8b) should therefore be preferred, in that case it is taken to be a name, as part of "Trilogy RC", and in that sense neglected.

Both examples show one of the major problems that will arise when language identification is automated, all words "sup", "man", "Trilogy" and "RC" exist in the English dictionary, but should not always be classified as such.

As we can see in the Confusion Matrix (Table 2.5), there exists quite some disagreement about whether a token should be labelled as social media term or as Dutch. Some are clearly mistakes, for example (everything with an SMT label is printed bold):

(9) a. ⟨SMT⟩ **@name1** ⟨/SMT⟩ ⟨NL⟩ en waar zei ik dat jatten wel ok is? ⟨/NL⟩ ⟨SMT⟩ **@name2 @name3 @name4 @name5 @name6** ⟨/SMT⟩

b. ⟨NL⟩ @name1 ⟨/NL⟩ ⟨NL⟩ en waar zei ik dat jatten wel ok is? ⟨/NL⟩ ⟨SMT⟩ **@name2 @name3 @name4 @name5 @name6** ⟨/SMT⟩ *@name1 and where did I say that it is ok to snitch? @name2 @name3 @name4 @name5 @name6*

In (9b) "@name1" is separately labelled with NL, but it should have been SMT because it starts with "@".

Most other disagreements are about the classification of exclamations, as we can see for example in (10):

(10) a. ⟨NL⟩ Dat ik terug moet werken. ⟨/NL⟩ ⟨SMT⟩ **Nah!** ⟨/SMT⟩

b. ⟨NL⟩ Dat ik terug moet werken. Nah! ⟨/NL⟩ *That I have to go back to work. Nah!*

Since "Nah" does not exist in the Dutch dictionary, it should be labelled SMT, hence (10a) is the better choice.

2.3.4 Corpus usage

Note that in the remaining of this thesis only the annotated corpus is used. Though, since the entire Twitter corpus may be useful to other researchers, it

is made freely available.[7]

2.4 Conclusion

A Twitter Corpus composed of Dutch messages was collected (roughly 95,000 tweets). A subset of 1,300 tweets, called the Annotated Corpus, was manually annotated. Six language labels were used per word: Dutch, English, mixed, social media term, other and unclear. Of all 19,464 tokens the majority was Dutch (85.8%), followed by social media terms (11.5%) and English (1.4%). Of all 1,300 tweets, 98.8% contain Dutch tokens, 86.4% social media terms and 8.5% English tokens. Therefore, I conclude that the ratio of English words used in Dutch Twitter messages is quite low (1.4%), but the number of tweets containing English words is considerable (8.5%).

Reliability of annotation was measured by inter-annotator agreement. Both annotators are Dutch natives and fluent in English. One person annotated the total Annotated Corpus, the other a 100. Calculation of inter-annotator agreement yielded a Cohen's kappa of 0.94, which indicates annotation to be highly reliable.

[7] http://illc.uva.nl/∼raquel/data/CS_prediction.zip

Chapter 3

Analysis of code-switches

3.1 Introduction

A first step towards predicting switches from and to English within Dutch tweets is to analyse the nature of its occurrences. There are some studies that analyse Dutch code-switches, take for example Broersma (2009), Papalexakis et al. (2014) or Yılmaz et al. (2016). However, these studies focus solely on bilinguals, while I do not use a specific bilingual sample. Characteristics of this specific problem are not yet described. Analysis might not only expose the issue at hand, but also provide insights that help to automate both CS prediction and language identification.

The Annotated Corpus was used to analyse Dutch-English code-switching. Before going further it is important to remember the great imbalance between Dutch and English tokens in the data set. Of the 19,464 tokens 1.4% is labelled EN and 0.05% as MIX, while the NL annotated tokens have a share of 85.8% (see Section 2.3.2). Dutch clearly is the most prevalent language; it demarcates the constraints for code-switching. Therefore Dutch is taken to be the dominant language.

Although a total of six labels is used in the manual annotation task, in this Chapter, I will only be concerned with English tokens (labels EN and MIX) and Dutch tokens (label NL). The other labels (SMT, OTH and UNC) are taken to be neutral and not mentioned here.

The following Section 2 is reserved for the explanation of code-switches in general. Also, a description of the particular switches analysed in this study (intra-sentential and intra-morpheme CS) is given. The third Section zooms in on intra-sentential code-switching and the fourth on intra-morpheme code-switching. Section 5 describes the distribution of Part-of-Speech tags over the encountered switches. The Chapter ends with a short conclusion.

3.2 Code-switches explained

Code-switching can be defined as the use of two or more languages by one speaker within a single conversation in which the other participant (one or more) has comparable understanding of the used languages. In Twitter messages, information is mostly directed at a group of persons who are free to join. It is assumed that the person using a code-switch expects its (main) audience to understand the foreign word(s).

Code-switches can be subdivided into three topics: *inter-sentential*; *intra-sentential* and *intra-morpheme* switches. First, the switch point of inter-sentential code-switches lies between two sentences, for example:

(1) Yesssss, eindelijk de #seizoensfinale van #Familie! **Let's kill June** @name1 @name2
Yesssss, finally the #seasonfinale of #Family! Let's kill June @name1 @name2

Second, intra-sentential switches take place within a sentence. A sentence can contain more than one switch point as we encounter in (2):

(2) **Basic** kan dus heel **stylish** zijn.
So basic can be very stylish.

Finally, there exist language switches at the level of morphemes, i.e. within a single token, called intra-morpheme language switches:

(3) Er wordt weer lustig er op los **geframed** door de NOS over #brexit.
Again at the NOS they are freely framing about #brexit.

In this study only the last two forms of switching are thoroughly investigated; this has mainly a practical reason, since the notion of inter-sentential switching is not apparent within separate tweets. Although, a tweet may contain more than one sentence, the majority does not (or not clearly, because of noisy punctuation). Accordingly, it is assumed that most inter-sentential switches are not in the corpus, since these will be labelled as English by the Twitter API and therefore excluded when collected. As a result of this choice, a small minority (i.e. 8 cases) of clearly recognizable inter-sentential switches is incorporated in the set of intra-sentential switches.

Due to the noisy data, as expected from Twitter messages, English words within quotation marks are not excluded as code-switch. The informal setting leads to non-standard use of quotation marks and other punctuation symbols. As such it is untrustworthy to use these as a clear sign for literal word reproduction disconnected from personal word selection.

Sometimes a difference is made between borrowing and code-switching. Although various definitions are known, according to Romaine (1995), the concept of borrowing stands for a word from another language that is often taken up and is partially or totally naturalized. Such a division, between borrowing and CS will not be made here. The main reason is that it is extremely difficult to

decide on what grounds a word is partially incorporated, let alone deciding what "often" exactly stands for. Therefore a word is taken to be naturalized when it appears in the Dutch dictionary; if not it is a code-switch.

3.3 Analysis: intra-sentential code-switching

In the annotated set a total of 110 tweets contain one or more intra-sentential code-switches. The total of segments, i.e. sequences of one or more English words, is 136, which in turn are comprised of 280 words.

About 26% of the CS segments have a length of three or more words. This group is on the one hand characterized with multi-word expressions such as idioms, tags and collocations as is shown in examples (4), (5) and (6). On the other hand, these segments correspond to names, e.g. a film title (7) or the name of a computer game (8). Often these segments are not totally integrated with the rest of the message, especially the multi-word expressions; in 77%[1] of the cases they appear at the borders, begin or end, of the tweet.

(4) @name1 oh echt super lief liggen ze samen, en je neefje is groot aan het worden :-) **god bless him**
@name1 oh really it's so sweet how they lay together, and your nephew is growing up :-) god bless him

(5) **We will be back** .... maar dan liever 2017 dan 2018
We will be back .... better 2017 than 2018

(6) @name1 slapende mensen zijn altijd zo mooi, volledig ontspannen is echt natuurlijke schoonheid **at it's finest**
@name1 people are so beautiful when asleep, totally relaxed is natural beauty at it's finest

(7) Jawel * De 430 premirekaartjes **The Boys Are Back In Town** @name1 zijn al uitverkocht https://t.co/xxx https://t.co/xxx
*Oh yes * The 430 premiere tickets The Boys Are Back In Town @name1 are already sold out https://t.co/xxx https://t.co/xxx*

(8) Om 18:00 ga ik live CS:GO spelen en de nieuwe game **"Dead by Daylight"** spelen! Mis het niet !
At 18:00 I will play CS:GO the new game "Dead by Daylight"! Don't miss it !

The group of English segments of length 2 cover 21%, but the largest group of 53% consists of just one word. This last group is generally more embedded within the message, compared to the segments of length 3. The integration of words does not only appear at a semantic level, but also at a syntactic level. The latter meaning that the English words are inserted into the Dutch framework. Several examples of this phenomena are shown in sentences (9), (10) and (11).

[1]This number is based on start and end point of the text segment that exemplifies either English or Dutch, meaning social media terms are taken to be neutral, i.e. neglected.

Still, not all single word code-switches are totally integrated, some appear as discourse markers as in example (12).

(9) @name1 verpest m'n momentje van **funny** zijn niet
@name1 don't spoil my moment of being funny

(10) @name1 Goede muziek is goede muziek. Of de serieuze 'kenner' het **dismissed** als fout of gewoontjes: swah.
@name1 Good music is good music. Even if the serious 'expert' dismisses it as wrong or bland: swah.

(11) **Basic** kan dus heel **stylish** zijn.
So basic can be very stylish.

(12) **Damn!** Ik heb een identiteitscrisis.
Damn! I have an identity crisis.

Returning to the segments containing two words; these seem to fall a bit in the middle. Part can be characterized as fully embedded (as in (13) and (14)), while another part shows more similarity with the more separated multi-word expressions (15).

(13) Ik zie mijn **best friend** voor de eerste keer deze zomer al 8 uren lang niet en ik mis die **fucking hard**
For the first time this summer I have to do without my best friend for 8 hours now and I miss him fucking hard

(14) @name1 @name2 @name3 Ik ben geen wetenschapper maar **gut feeling** zegt dat 100'den km2 met rust laten heel goed gaat blijken
@name1 @name2 @name3 I'm not a scientist but gut feeling says that leaving alone more than 100 km2 will show to be good

(15) @name1 zoektocht **it is**
@name1 quest it is

3.4 Analysis: intra-morpheme code-switching

Words labelled as MIX embody the intra-morpheme switches. These appear rarely, only 9 times in this data set, and mostly alone. Just once MIX is accompanied by an EN label, though not as a direct neighbour. This might be explained by the fact that a mixed word is a CS in itself. And, according to Poplack (1980), a quite complicated one. Therefore, an intra-morpheme CS immediately followed by another CS is maybe too complex to apply. Nevertheless, as mixed words are rarely encountered in the data set, it might just be a matter of corpus size.

The mixed words are in majority verbs that show a clear pattern. All are English stems either combined with the Dutch past participle marker 'ge-' or the infinitive marker '-en'. The result is a Dutch verb conjugation of an English verb as shown in (16) and (17). This forces the English word into the Dutch

grammar structure. Also there are two English-Dutch noun combinations, such as in example (18).

(16) @name1 ja dat is zo kapot irritant, moet je de game weer **restarten** enzo
@name1 yes that is terribly irritating, do you have to restart your game and stuff

(17) @name1 Ik heb mijn tegenstander vv's gestuurd, niet **geaccept**. Kan je misschien ook vertellen wie moet hosten?
@name1 I sent vv's to my opponent, not accepted. Can you tell me who will be hosting?

(18) Nog 2 dagen en dan ben ik eindelijk weer even van mn **avonddienst-streak** af. En 2 dagen vrij. Hallelujah.
Only 2 days and I am finally releaved from my night shift streak. And 2 days off. Hallelujah.

Except for one, none of the mixed words appear at either start or end of the tweet.

3.5 Analysis: Part-of-Speech tags

In Table 3.1 an overview of PoS tag distribution over the classes EN and MIX is displayed. All words were manually tagged by one annotator. Nouns represent the largest group in the English class (38.6%). The second and third largest are adjectives (15.4%) and verbs (14.3%). In case of the mixed word class, verbs show to be prevalent (66.7%) followed by nouns with (22.2%).

The outcomes seem to support the theory displayed by Joshi (1982). Although closed class members are switched, i.e. pronouns, determiners, conjunctions, prepositions and auxiliary verbs, all of them are part of a verb or noun phrase, none appear on their own. English words that do occur alone are all members of open classes (nouns, verbs, adjectives, adverbs); without any exceptions.

I do not know whether the users are fluent bilinguals or not, and if so, which ones. Still, it might be interesting to see if the data implies one of the two. After studying a group of Spanish-English bilinguals, Poplack (1980) finds support for her hypothesis that the kind of code-switches say something about their bilingual fluency. She states that fluent bilinguals often use *intimate code-switches*, i.e. complex intra-sentential language switches. On the other hand non-fluent bilinguals make use of the relatively easier *emblematic code-switches*, i.e. single noun switches combined with inter-sentential switches. As the definition of these terms is not totally clear, here emblematic CS is defined as the set of single noun switches and switches that are not embedded into the Dutch sentence structure. Intimate CS is the set of intra-morpheme switches and embedded switches that are not single nouns.

As we have seen in Section 3.3, many of the multi-word and some single-word

Grammar rule	EN	MIX	EN+MIX
Noun	**108 (38.6%)**	2 (22.2%)	**110 (38.1%)**
Adjective	43 (15.4%)	0 (0.0%)	43 (14.9%)
Verb	40 (14.3%)	**6 (66.7%)**	46 (15.9%)
Determiner	27 (9.6%)	0 (0.0%)	27 (9.3%)
Adverb	24 (8.6%)	0 (0.0%)	24 (8.3%)
Preposition	17 (6.1%)	0 (0.0%)	17 (5.9%)
Abbreviation	9 (3.2%)	0 (0.0%)	9 (3.1%)
Other	5 (1.8%)	0 (0.0%)	5 (1.7%)
Unclear	4 (1.4%)	1 (11.1%)	5 (1.7%)
Digit/punc.	3 (1.1%)	0 (0.0%)	3 (1.0%)
TOTAL	280 (100%)	9 (100%)	289 (100%)

Table 3.1: *PoS-tags English and mixed words; percentages between brackets.*

code-switches are not embedded, I count 61 of those. Moreover, nouns represent the largest group in the English labelled class; of those 49 occur alone. On the other hand, as we have seen in the previous section, intra-morpheme switches occur rarely, just 9 times. The amount of embedded switches that are not single nouns is not very high either; I only count 26 of them. Now, if we take all these numbers together, the number of emblematic CS is 110 (76%) occurrences compared to 35 (24%) of intimate CS. As this data set does not involve inter-sentential CS, the share of emblematic switches might be even higher. It is not a surprise that the encountered code-switches are mostly emblematic, because the collected data does not belong to a group with specific bilingual abilities, but to a broad group of random Twitter users.

These numbers might give a first indication about the bilingual language level of average Dutch Twitter users. Clearly, further research is needed to define more precisely what intimate vs emblematic CSs correspond to in Twitter data.

3.6 Conclusion

Three levels of CS are discerned: inter-sentential CS; intra-sentential CS and intra-morpheme CS. Here I focus on the last two forms. In the Annotated Corpus 136 intra-sentential code-switches are found. About 26% have the length of 3 or more words. Most of these are multi-word expressions or named entities. Often these CSs are not fully integrated in the Dutch main sentence structure, which is supported by the fact that 77% appears either at the start or the end of the message.

The largest group CSs (53%) consists of a single English word. On average this group is more embedded within the Dutch semantic and syntactic framework. Segments containing 2 words cover 21%; these seem to be somewhere between characteristics of length 1 and length 3+.

The group of intra-morphemes is extremely small with only 9 occurrences.

Most (67%) are verbs, showing a clear pattern: an English stem combined with a Dutch verb conjugation morpheme. Literally forcing the English segment into the Dutch grammar pattern. Also there are two English-Dutch noun combinations.

Most words used for CS are nouns, followed by verbs and adjectives. Other PoS tags, such as determiners, pronouns and prepositions, do occur in noun or verb phrases, but never on their own. This result supports Joshi's (1982) theory that code-switches are constrained; only open class members can be used.

The data might imply the average Twitter user to be a non-fluent bilingual. The majority of the CSs (76%) are emblematic code-switches Poplack (1980), which are relatively easy to apply. As this data set does not involve intersentential CSs, the share of emblematic switches might be even higher. Also, a better definition is needed of what intimate vs emblematic CSs correspond to in Twitter data. Therefore, further research is necessary.

Chapter 4

Automatic CS prediction

4.1 Introduction

One of the main goals of this thesis is to automatically predict code-switches (CS), at word level to and from English within Dutch Twitter messages. A better grasp of the mechanism behind code-switching can improve automatic multi-language processing. Currently, a shift in language is (often) not recognized, which leads to bad performance or even failing of widely used language technologies such as sentence parsing and machine translation.

In order to find the best performing model, I chose to start with an ideal situation in which all words are manually labelled according to the six categories described in Section 2.3.1. Clearly, this is not a realistic setting, since in a totally automated process, language identification at word level is also automated. Therefore, the next Chapter 5 is dedicated to the task of language identification and to the combination of both automated language identification and automated code-switch prediction.

This Chapter starts with a short description of the task at hand, followed by an explanation and evaluation of used features. Third, three prediction models are described and tested, namely a Multinomial Naive Bayes model, a Support Vector Machine and a Decision Tree model. Fourth, some extra attention is given to performance evaluation of the models. Finally an overview of the results and their evaluation is given in the last section.

4.2 Code-switch prediction task

The task of a CS prediction model is to predict whether a code-switch will take place between the current and next token. So, following Solorio & Liu (2008), every word boundary is taken to be a potential point of language change. As this research is specifically set to language switches between English and Dutch in both directions, I only take into account these two languages.

In this context a Dutch word has label NL and an English word label EN or MIX (more information about the meaning of language labels can be found at Section 3.2.1). Additionally, there is a group of neutral tokens, i.e. tokens without bearing any English or Dutch implication, which are either classified SMT, UNC or OTH, or solely consist of digits or punctuation signs.

In (1) there are some schematic examples shown: the * indicates a switch point, NL embodies a Dutch word, EN an English word and X a neutral token. Suppose for example that a message starts with an English word, followed by two neutral tokens, and ends with a sequence of Dutch words. In that case one switch is counted and is placed just in front of the Dutch word, since the neutral ones are overlooked (see (1a)). When a tweet contains one or more sequenced English words, surrounded by Dutch (and/or neutral tokens), two transitions are counted (see examples (1b) and (1c)).

(1) a. EN X X * NL NL NL NL NL.
b. NL NL NL * EN * NL X X X.
c. X X NL NL * EN EN EN * NL NL NL.

Example (1a) has 7 potential switch points of which 1 is an actual switch; (1b) has 7 potential points with 2 actual switches; and (1c) counts 9 potential points of which 2 are actual switches. If there are n tokens there are $n - 1$ potential switch points. The task is to recognize the actual switch-points between the potential ones.

Note that MIX embodies a word containing an English-Dutch code-switch at intra-token level. Consequently, to place switch points at word boundaries bends the truth. However, because the group of MIX labelled words is extremely small (0.1% on the word total) and language switches do take place to and from English *within* the indicated points, this error is overlooked.

As this task involves prediction, training is exclusively based on information available prior of the token to be predicted. The program is not allowed to take into account any information about the next token or everything that might follow. Such a restriction influences model choice, since all models that take into account information beyond this threshold, e.g. Conditional Random Field models, are prohibited. These measures are taken to provide the means of a tool that can be used in real time applications.

4.2.1 Challenge

Distribution of the positive and negative classes is quite uneven, due to the fact that an everyday Dutch Twitter data set was chosen, the set is dominated by Dutch. Of all potential switch points, only 1.3% is an actual switch point (positive class). The imbalanced class distribution complicates the classification task.

4.3 Feature selection

A total of 10 features are evaluated, of which 6 are selected. All considered features are found in Table 4.1.

4.3.1 Feature description

All evaluated features use information about n that refers to the word previous to the potential switch point to classify. In other words, n is the last token that decides the current language. Language can only be decided by a word tagged with EN, MIX or NL; other labels are neutral.

Index The first feature calculates the position of n relative to the start of the considered tweet. In case n does not exist, i.e. all preceding words are neutral, the value is set to -1. Such a situation may arise for example, when all preceding tokens are classified as SMT. The position of n might provide additional information about the likelihood of an upcoming language shift.

Language id Whether a token n is English or Dutch is checked in features #2 to #4. This information is based on manual language identification. If n is English (i.e. classified as EN or MIX), it will get value 1, when Dutch the value is 0 and in case n does not exist the value is -1. The same holds for its preceding tokens (up to two). When there exists no n, $n-1$ or $n-2$ due to tweet length the value is also set to -1. Information about the (amount of) preceding language tags seems of great value: most of the English words, 53%, occur alone; 21% consists of a group sized 2 and 13% of size 3.

Length Features #5 to #7 measure the length of n, $n-1$ and $n-2$ in characters. In case there exists no n, $n-1$ or $n-2$ the value of this token is -1. The length might provide information about the PoS tag of the token, for example a determiner is on average shorter in length compared to a noun or adjective.

CS count The amount of code-switches up to n is calculated in features #8, #9 and #10. Feature #8 particularly counts the switches from Dutch to English; feature #9, the specific switches from English to Dutch are calculated; and feature #10 the total switches. These features seem to be quite informative for the CS prediction task. For example, if there has been just one switch counted, e.g. from NL to EN, it is not possible to have another switch in the same direction. On the other hand it is quite likely to have a switch from EN back to NL, since Dutch is the dominant language. Now, in this new situation, in which two switches are counted (NL to EN and EN to NL), it is not that likely to have another switch within the current message. In the total set of tweets containing English words, 84.9% contains just one occurrence of adjoined English words.

#	Feature description	F-score	P-value	Rank
1	Position of n from start of tweet	2.501e+00	1.138e-001	8
2	Language id of n	4.497e+02	1.508e-098	**3**
3	Language id of $n-1$	2.060e+01	5.683e-006	**6**
4	Language id of $n-2$	1.633e+00	2.014e-001	9
5	Length in characters of n	2.095e+01	4.752e-006	**5**
6	Length in characters of $n-1$	4.587e+00	3.223e-002	7
7	Length in characters of $n-2$	3.425e–01	5.584e-001	10
8	Count of NL→EN switches before n	1.496e+03	5.060e-314	**1**
9	Count of EN→NL switches before n	1.963e+02	2.359e-044	**4**
10	Count of total switches before n	7.552e+02	8.714e-163	**2**

Table 4.1: Feature numbers and descriptions, univariate score, p-value and rank per feature; boldface rankings $p < 0.00001$

4.3.2 Feature analysis

Univariate statistical tests are performed to analyse the predictive strength of every feature. Per feature the ANOVA (*analysis of variance*) F_1-score is determined, dependent on the provided data and a null hypothesis. This score is similar to a t-test, but generalized to apply to more than two groups. Statistically significant results justify the rejection of a null hypothesis. In the setting of this research, the null hypothesis is that none of the features have any influence on learning the CS prediction task. Rejecting the null hypothesis therefore, means that different features have their own effect on training, i.e. predictive power, that is not given by chance. Based on both the F_1-score and p-values, the list of features is ranked. All information is found in Table 4.1.

Here, the significance level is set at $p < 0.01$. The features that do not have enough significant predictive power are the position of n (#1), the language tag of $n-2$ (#4) and the length of both $n-1$ and $n-2$ (#6, #7). These features, of which most capture a more distant context, are excluded for model testing.

The features that are most predictive represent the code-switches encountered hitherto (#8, #9, #10), the language id of both n and $n-1$ (#2, #3) and the length of n (#5). These features all significantly contribute to the prediction of code-switching, not only do they respect the $p = 0.01$ threshold, for all holds that $p < 0.00001$ (in Table 4.1 their rankings are printed boldface). Therefore all of these features are used for training and testing the machine learning models.

4.4 Models

Three supervised learning models are tested: a Multinomial Naive Bayes classifier, Multiclass Support Vector Machine and a Decision Tree classifier.

4.4.1 Multinomial Naive Bayes (MNB) model

A Multinomial Naive Bayes (MNB) model is an implementation of a naive bayes algorithm specified to handle multinomial distributed data. This is a classic naive bayes variant often used for classification of texts. The Naive Bayes model is available at scikit-learn.[1]

4.4.2 Decision Tree classifier model

The aim of a Decision Tree classifier is to predict a class based on simple decision rules inferred and learned from the data features. Class labels are represented by the leaves; specific feature combinations that indicate a certain class are represented by the branches. The algorithm to decide the best branch split is based on a *Gini impurity* measure. With Gini impurity one can calculate the chance that an item randomly gets an incorrect label according to the label distribution of the set. This information is used as a baseline to measure the quality of a split. The decision tree model is available at scikit-learn.[2]

4.4.3 Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a linear multi-class classification model trained on 1-slack soft-margin formulation. The model is thereby capable to handle data that is in fact not perfectly linearly separable. The best linear division is decided by calculating the smallest difference between a maximal and minimal margin. The introduction of *slack variables*, one for every example, allow a data point to be inside the margin or even on the wrong side of the decision boundary; though these are seen as margin errors and penalized as such. Both model and learner are presented by the Pystruct library.[3]

4.5 Evaluation of performance

As mentioned in Section 4.2.1, we are dealing with a highly imbalanced dataset. Consequently, a model that does not predict any switch points yields an accuracy of around 98%. Clearly, accuracy does not provide the needed insightful information. Therefore, the focus lies on precision, recall and F_1-score of the predicted switch points. Precision is the proportion of truly predicted positives amidst the total set of predicted positives; in other words, the amount of relevant items within the set of predicted items. In this context, precision is the percentage of predicted code-switches that are true among the total set of predicted code-switches (both true and false).

Recall indicates the proportion of truly predicted positives among the total number of true positives; or the percentage of selected relevant items within the total set of relevant items. To calculate recall, the set of positives that are

[1] http://scikit-learn.org/0.17/index.html
[2] http://scikit-learn.org/0.17/index.html
[3] https://pystruct.github.io/index.html

truly classified is divided by the sum of truly classified positives plus the group of items that should have been classified as positive. Within the perspective of this thesis, recall means the portion predicted code-switches that are true among the total set of true code-switches including the ones that should have been recognized.

The F_1-score is the harmonic mean of precision and recall. Compared to the commonly used arithmetic mean, the harmonic mean reduces the weight of large extreme numbers and induces the weight of small extremes. This statistical measure is often used in computer science as indication of the performance of classification algorithms. For F_1-score, precision and recall the like, it holds that 0 is the worst value while 1 is the best.

4.6 Results

For model development 10% of the annotated corpus was set aside; training and testing was performed in a 5-fold cross-validation setting using the other 90% of the corpus. Labels were divided evenly over the five folds to deal better with the skewed label distribution. Model performance is measured with precision, recall and F_1-score on the automatically selected switch points, as described in the previous section. The results can be found in Table 4.2.

As a baseline system I trained the models on the second feature only; this feature indicates whether the language of n is English, Dutch or neutral. The language tag of the last non-neutral word decides the language setting in which prediction has to be performed. There are two reasons to choose this particular feature; first, there cannot be a prediction of a language switch if the current language is unknown. Second, if n is neutral there cannot be a language shift. Hence, intuitively this knowledge seems to play a key role in code-switch prediction.

As we can see in Table 4.2, the Decision Tree model scores best on the level of code-switch prediction for the baseline model with precision 7.9%, recall 10.4% and F_1-score 9%. Both MNB and SVM do not predict any code-switches. Differences in results between Decision Tree and both MNB and SVM are not significant though, probably due to a standard deviation of 16% (precision), 21% (recall) and 18% (F_1-score).

Next to the baseline, several feature combinations were used for training the models; I only show the two most informative settings containing the best results: first, a framework using the top 3 best scoring features; second, a context in which all features are used.

First, in the top 3 setting, the three highest ranked features are used (see Table 4.1). In order of predictive strength, these are numbers #8, #10 and #2 or the amount of NL to EN switches before n, the total count of switches before n and the language id of n respectively. Second, learners of the models are trained with all selected features, i.e. #2, #3, #5, #8, #9, #10 with a significance of $p < 0.01$. Both MNB and SVM models perform better the more features are added for training, while the Decision Tree model performs best

when trained on the top 3. None of the intermediate settings (not shown here) returned better outcomes. In case of the Decision Tree model, training on the top 4 features produced the exact same results using the top 3.

Taking a closer look at Table 4.2, we see that the Decision Tree model yields the best results of all in the top 3 feature setting; with 43.9% on precision, 50.7% on recall and 46.3% on F_1-score (correspondingly, standard deviations are 12.4%, 2.7% and 5.5%). These results not only significantly improve the Decision Tree baseline (paired t-test $p < 0.05$), but also shows significant improvement compared to the outcomes of the best Naive Bayes classifier (paired t-test $p < 0.01$) and best SVM classifier (paired t-test $p < 0.05$).

In comparison, the best Naive Bayes classifier of Solorio & Liu (2008) obtains 27% F_1-score, with 18% precision and 59% recall. They used a Naive Bayes model trained on lexical and syntactic features to identify English-Spanish code-switching points. Similarly to the approach in this Chapter, Solorio & Liu use a manually annotated data set for language identification.

	MNB			**Decision Tree**			**SVM**		
	p	r	f1	p	r	f1	p	r	f1
Base(#2)	.000	.000	.000	.079	.104	.090	.000	.000	.000
Top3(#2,8,10)	.152	.074	.096	**.439**	**.507**	**.463**	.000	.000	.000
All(#2,3,5,8-10)	.188	.102	.124	.425	.415	.416	.134	.008	.016

Table 4.2: *Results of precision recall and f1-score of code-switch prediction for the Multinomial Naive Bayes (MNB) model, the Decision Tree model and the Support Vector Machine model. The best results are shown in boldface.*

As mentioned at the start of this Chapter, one should keep in mind that the presented results are idealized in the sense that feature extraction of the CS prediction learner is based on a manually annotated data set. At the end of Chapter 5, the best code-switch prediction model (i.e. the Decision Tree model), will be combined with a data set in which the language is indeed automatically identified.

4.7 Conclusion

The binary classification task is set to predicting whether a CS takes place between the current and next token; this without information about the next token or what might follow. Three supervised learning models were tested on this task: a Multinomial Naive Bayes (MNB) model, a Decision Tree model and a Support Vector Machine (SVM). All models were trained in a 5-fold cross validation setting.

Six features were used for training. These features were selected based on their ANOVA F_1-score during development. Models are evaluated by calculating precision, recall and F_1-score of the positive class. As baseline the models were tested on one feature (indicating the language of n). Best performance results

are yielded by the Decision Tree model; with 43.9% on precision, 50.7% on recall and 46.3% on F_1-score. These results significantly improve performance of the baseline and of both the best MNB and best SVM classifiers.

However, we should note that the CS prediction task was performed in an idealized setting, because the features were trained on manually tagged tokens.

Chapter 5

Automatic Language Identification

5.1 Introduction

The aim of the previous chapter was to automate code-switch prediction. The task of prediction was performed in an idealized situation, namely: features were trained on words with manually predetermined language tags. In order to fully automate code-switch prediction it is necessary to automate language identification.

Improvement of automatic language identification can also be beneficial for natural language processing in general. Inter-sentential language shifts are often not recognized, impairing performance of traditional techniques such as PoS tagging or spell checking.

In Section 2, I explain the task of language identification in more detail, followed by the challenges it involves. The third section is set apart for model description; besides a baseline model, one non-probabilistic rule-based and three machine learning models are considered. Affixed to the paragraph dedicated to the machine learning models, there is a section about feature selection. In this part all features are explained and evaluated. Fourth, there is a section on evaluation of performance. Fifth, performance results are given of the language identification models and in Section 6 these results are discussed. In Section 7, the best CS prediction model is combined with the best language identification model. Finally, the chapter is completed with a short conclusion.

5.2 Language identification task

Language identification at word level can be approached as a task of classification. Separate tokens are sorted individually into one of the designated classes. The task in itself does not require an online approach; it is perfectly fine to as-

sume access to the whole document. However, it should be stressed that in this particular real-time context (i.e. in combination with the online CS prediction task), any knowledge surpassing the current token is prohibited.

For the language identification task the same set of classes is used as for the manual annotation task; tokens are similarly divided into the following 6 classes (A clear description of class labels and the manual identification task can be found in Section 2.3.1 and Appendix A):

1. Dutch tokens are labelled as NL (85.8%)
2. English tokens are labelled as EN (1.4%)
3. Tokens that are partly Dutch and partly English, i.e. mixed tokens, are labelled as MIX (<0.1%)
4. Tokens that are identified as Social Media Terms are labelled as SMT (11.5%)
5. Tokens of another language (either German, French or South-African) are labelled as OTH (0.7%)
6. Unclear tokens that do not seem to fall into any of the other classes are labelled as UNC (0.5%)

The aim of this specific language identification task is to mimic the manual annotation of the data set as close as possible. The set of labels is therefore exactly the same.

5.2.1 Challenges

In the listing of classes above, percentages per class of the total Annotated Corpus are shown. Note that distribution of classes is rather uneven, due to the fact that everyday Dutch Twitter data was chosen (e.g. instead of a bilingual set), the set is dominated by Dutch. The imbalanced partition of the classes complicates the classification task.

Another difficulty concerning the task at hand is lexical overlap among Dutch and English dictionaries. Dictionary lookup is the main approach in this study to discern between English and Dutch (exactly how this was done is explained in Section 5.3.2). I calculated the dictionary overlap existing between the English and Dutch labelled words. This is an often used technique to illustrate this challenge.

Of all words labelled NL, 48% appear in both Dutch[1] and English[2] dictionaries. For the EN annotated words this was 42%. For comparison, in Nguyen & Doğruöz (2013) the encountered overlap between Turkish-Dutch words was 24.6% for the combined labels. The high percentage in overlap between Dutch

[1] http://www.opentaal.org/bestanden.html
[2] http://wordlist.aspell.net/dicts/

and English can be explained by the fact that these languages are closely related and therefore share many same and similar words (Broersma, 2009).

The encountered group of words appearing in both dictionaries is varied. First there are cognates, sharing the same form and meaning, such as 'in', 'festival' or 'is'. Also there are words formerly borrowed from English, but now incorporated; think of 'mountainbike' or 'business'. Moreover, there is a group of words that share the same spelling but have a different meaning and a different pronunciation; such as 'ten', 'die', 'door' or 'van'.

5.3 Models

In total five models are evaluated in this Chapter. The first is a baseline model that uses an off-the-shelf language identification tool. The second is a non-probabilistic rule-based algorithm developed for this specific task. The remaining models are all supervised machine learners: a Support Vector Machine (SVM) model, a Decision Tree model and a Chained Conditional Random Field (CRF) model. Development of the models is based on a development set, which is 10% of the annotated corpus. The remaining 90% was set aside for testing and training in a 5-fold cross validation setting.

5.3.1 SMT identification tool

All models, including the baseline, use a tool specifically developed for this study to automatically recognize social media terms or SMTs (see Section 2.3.1 for an exact definition). For every token it is decided whether it is probably an SMT and, if not, identification of the remaining language labels is performed. To make sure most of the SMTs are actually recognized, tokens are not preprocessed beforehand, e.g. removal of punctuation, but along the way correspondingly to the specific purpose. In order to identify most SMTs, for every token it is checked whether it starts with @, & or #; whether it is a URL; whether it is an emoticon; whether it is an emoji;[3] or whether it appears as a member of the provided SMT word list.[4] In Figure 5.1 a simplified version of the algorithm is provided.

5.3.2 Dictionary lookup

Furthermore, all of the evaluated models, except the baseline, make use of digital word lists, here often called dictionaries. These dictionaries function as language reference. First a Dutch dictionary is used based on a word list provided by OpenTaal[5] in combination with a Dutch names list provided by Meertens Institute.[6] English words were looked up in a list based on the Hunspell dictio-

[3] An emoticon exist of punctuation, e.g. :-) an emoji is an actual picture, e.g. ☺.
[4] This list is given in Appendix C
[5] http://www.opentaal.org/bestanden.html
[6] http://www.meertens.knaw.nl/

```
for i in tokens do
   if i starts with @, & or # then
      i ← SMT
   else if i == URL then
      i ← SMT
   else if i == emoticon then
      i ← SMT
   else if i == emoji then
      i ← SMT
   else if i in SMT list then
      i ← SMT
   else
      i ← function.find_other_label(i)
   end if
end for
```

Figure 5.1: *Pseudo code SMT identification tool.*

nary.[7] In theory the label OTH (other) embodies all other languages. Clearly, it is impossible to run a program with dictionaries for all other languages. Therefore, I selected the three main languages next to English: German, French and South-African.[8] The choice is based on manual analysis of the OTH labelled words of the development set. Since all tokens that belong to another language are annotated as OTH, no exact numbers are available.

All dictionary entries are lowered (i.e. capitals are replaced with lowercase) and in case the dictionary item contained a space the entry was split in two (since I only look at one space separated token at a time). To limit running time, all dictionaries are stored as a trie.[9]

On average, the dictionaries are composed of words in their basic form. Meaning that, when searched, other forms such as verb conjugations or plurals are not found, since there is no exact match. Especially concerning the two main languages English and Dutch focussed on in this thesis, this encountered weakness proves to be problematic. Often a stemmer, such as the well known NLTK Snowball Stemmer,[10] is used in order to find the base form of a word. Unfortunately, performance of stemmers for Dutch is low. To make sure most word forms are indeed found; I use the pattern library for both languages.[11] The pattern library provides a variety of high quality functions that turn given words into a different form according to the corresponding grammar rules of the

[7] http://wordlist.aspell.net/dicts/

[8] The dictionaries of South-African, German and French are found on http://www.winedt.org/dict.html

[9] A trie or digital tree is an efficient search tree suited for dictionary data structures. For this study the marisa-trie library was used: https://pypi.python.org/pypi/marisa-trie

[10] http://www.nltk.org/api/nltk.stem.html

[11] http://www.clips.ua.ac.be/pages/pattern-nl and http://www.clips.ua.ac.be/pages/pattern-en

chosen language. Think for example of the singularization or plurarization of a word.

Singularized noun and lemmatized verb forms are added for both Dutch and English dictionary lookup. For example, noun singularization turns the plural noun *nijlpaarden* (*hippopotamuses*) into the singular *nijlpaard* (*hippopotamus*); the first plural form is not in the Dutch dictionary, while the singular form is. The same holds for verb lemmatization which provides us with the infinitive form of a conjugated verb; e.g. *danste* (*danced*) does not exist in the Dutch dictionary, but *dansen* (*to dance*) does.

Furthermore, two extra forms are added for Dutch lookup only: first a predicative form and second a combination of a predicative and lemmatized form. Firstly, in Dutch adjectives followed by a noun are inflected with an -e suffix, for example *gele* (*yellow*) in *een gele jas* (*a yellow coat*) is only recognized if transformed to the predicative *geel* (*yellow*). Secondly, words of the form *zingende* (*singing*) as in *de zingende tijger* (*the singing tiger*) are only properly recognized as *zingen* (*to sing*) when first the predicative word form is taken and then the lemmatized form.[12]

5.3.3 Baseline

As baseline model, I use the off-the-shelf language identification tool *Langid.py*[13]. This program is pre-trained over 97 languages, including English and Dutch. The program was presented with a single word input. In case of inconclusive language identification, the token is labelled as Dutch. Note that *langid.py* was developed to identify a language at document level instead of word level. For this reason, expectations on performance are low.

5.3.4 Rule-based dictionary lookup (RBDL)

The rule-based dictionary lookup (RBDL) model is a non-probabilistic algorithm, manually developed for the specific language task at hand. As the name already indicates, the model tests whether (the base form of) a token occurs in one of the provided dictionaries. With the pattern library[14] singularized, lemmatized and predicative word forms are added for dictionary lookup purposes. Moreover, extra rules are added to identify mixed words (MIX), compound words (both EN and NL) and most common exclamations (SMT). In Figure 5.2, the simplified pseudo code of the algorithm is shown.

Rule hierarchy plays an important role in this particular algorithm. The chosen order is the result of trial and error, therefore by no means do I want to claim this hierarchy the best, only the best I could find. There are 9 main rules, so in theory there are $9! = 362,880$ possibilities. As far as I know there

[12] The singularized form is *zingende* (*singing*), the lemmatized form *zingennen* (*not proper Dutch*) and the predicative form is *zingend* (*singing*); none are in the Dutch dictionary

[13] https://github.com/saffsd/langid.py

[14] http://www.clips.ua.ac.be/pages/pattern-nl and http://www.clips.ua.ac.be/pages/pattern-en

```
for i in tokens do
   if i != SMT then
      if i == digit or i == punctuation then
         i ← NL
      else if i in NL_dict then
         i ← NL
      else if singularized(i) in NL_dict or lemmatized(i) in
       NL_dict or predicative(i) in NL_dict then
         i ← NL
      else if i in EN_dict then
         i ← EN
      else if singularized(i) in EN_dict or lemmatized(i) in
       EN_dict then
         i ← EN
      else if lemmatized(predicative(i)) in NL_dict or i ==
       NL_NL_compound then
         i ← NL
      else if i == exclamation then
         i ← SMT
      else if i in DE_dict or i in FR_dict or i in AF_dict then
         i ← OTH
      else if i == EN_NL_compound then
         i ← MIX
      else
         i ← NL
      end if
   end if
end for
```

Figure 5.2: *Pseudo code dictionary lookup tool.*

is no scientific way to find the optimal order of rules. It should be clear that this realization adds to the motivation to evaluate machine learning models, as described in the next Section.

5.3.5 Machine learning models

The models under attention in this Section are all three supervised Machine Learning (ML) models, equipped for classification problems; these are a Support Vector Machine (SVM), a Decision Tree and a Chained Conditional Random Field (CRF). Performance of both the Decision Tree model and the SVM was also tested in the code-switch prediction task in the previous chapter. These models are likewise suited for online word identification.

The CRF model cannot be used for real-time language identification, because it uses context information of a total Twitter message. Therefore it is not

applicable in combination with the code-switch prediction task. On the other hand, results are of great interest, since not all language identification tasks expect an on-the-go approach. For example, think of the Twitter language identification API.

Before describing the ML models, I will first elaborate on the features used in all of these models.

5.3.6 Feature selection

In total 30 features are tested, which all have significant predictive power ($p < 0.001$). The considered features are found in Table 5.1. Note that these features are used only for the machine learning models, but not for the baseline and rule-based model as described in the previous Sections 5.3.3 and 5.3.4.

Feature description

All evaluated features use information about either i, $i-1$ or $i-2$; i refers to the current token to classify.

Length The first feature counts the total amount of characters in i. That is, the number of alphabetical signs; numbers, punctuation and emoticons are not counted. The length might provide information about the class of the token. For example, very long tokens may indicate it to be a URL or exclamation; which both should be classified as SMT. Tokens with length 0, on the other hand, might indicate the token to be a number (most of these get labelled NL), punctuation (NL or SMT) or an emoticon (SMT); which narrows the set of options from 6 classes down to 1 or 2.

Vowel-consonant ratio Feature #2 calculates the vowel-consonant ratio of a word on a scale of 0 to 10 by *total vowels / total characters* $*10$. Since all non-alphabetic characters, such as punctuation, are removed, it is assumed that the remainder consists of consonants. In other words, if this feature has value 0, the token consists of consonants only; when the value is 10 it consists of vowels only; in case the word is formed by three vowels plus three consonants, the value is 5.

One of the main reasons to include this feature is that it might indicate if the token is a social media term in the form of an exclamation or onomatopoeia. Either high (above 7.5) or low (under 2.5) values can be a signal, since on average words will have a ratio in the middle. For example the exclamation *NOOOOOOOO!* (value = 8.9) or the onomatopoeia *mmm* (value = 0).

SMT label The boolean feature #3, tests whether i should be labelled SMT (value 1) or not (value 0). The setup of this feature is completely in line with the workings of the SMT identification tool as described in 5.3.1. Clearly, this feature is extremely important in deciding whether a token is a social media term.

Dictionary lookup Language classification is concerned in features #4 to #8. If the token does not already have an SMT label nor is a digit nor comprised of punctuation only, it is tested whether it occurs in any of the dictionaries. Per feature one language is checked, these are English (EN), Dutch (NL), German (DE), French (FR) and South-African (AF). Implementation is in line with the dictionary lookup tool in 5.3.2. The produced values are boolean; the value is 1 if the word occurs in the dictionary, otherwise 0. In features #9 and #10 the same for English and Dutch dictionaries is performed for $i-1$; in #11 and #12 for $i-2$. Dictionary lookup is of great importance for the language identification task, since it provides direct information to which language(s) the token might belong. Due to word overlap between the various dictionaries value results are not exclusive.

Extended dictionary lookup for NL an EN The boolean features #13 to #30 test the occurrence of various word forms in Dutch and English dictionaries. All forms are checked for i, $i-1$ and $i-2$. Again, implementation of the features is in line with the dictionary lookup tool (Section 5.3.2). Many of the words are not found if one only uses the dictionary lookup features described in the paragraph above. The cause lies in the inability to recognize a specific word form. The additional lookup of specific word forms in both Dutch and English dictionaries is of great value in the language identification task of separate words.

#	Feature description	F-score	P-value	Rank
1	length of i	1079.357	0.000e-000	**6**
2	Vowel-consonant ratio of i	211.546	8.897e-220	14
3	i is SMT	20378.817	0.000e-000	**1**
4	i in NL dictionary	1941.283	0.000e-000	**2**
5	i in EN dictionary	343.753	0.000e-000	**9**
6	i in DE dictionary	136.332	2.859e-142	19
7	i in FR dictionary	106.460	4.386e-111	23
8	i in AF dictionary	440.375	0.000e-000	**8**
9	$i-1$ in NL dictionary	1079.848	0.000e-000	**5**
10	$i-1$ in EN dictionary	322.916	0.000e-000	**10**
11	$i-2$ in NL dictionary	783.413	0.000e-000	**7**
12	$i-2$ in EN dictionary	261.660	1.352e-270	13
13	singular form i in NL dictionary	1134.619	0.000e-000	**4**
14	lemmatized form i in NL dictionary	179.773	3.247e-187	16
15	predicative form i in NL dictionary	1495.804	0.000e-000	**3**
16	pred-lem combiation form i in NL dictionary	198.814	9.233e-207	15
17	singular form i in EN dictionary	166.213	3.075e-173	18
18	lemmatized form i in EN dictionary	171.110	2.720e-178	17
19	singular form $i-1$ in NL dictionary	275.381	2.144e-284	12
20	lemmatized form $i-1$ in NL dictionary	53.769	1.353e-055	26
21	predicative form $i-1$ in NL dictionary	313.227	3.310e-322	11
22	pred-lem combination form $i-1$ in NL dictionary	59.687	7.508e-062	25
23	singular form $i-1$ in EN dictionary	110.108	6.636e-115	21
24	lemmatized form $i-1$ in EN dictionary	106.327	6.044e-111	24
25	singular form $i-2$ in NL dictionary	108.864	1.330e-113	22
26	lemmatized form $i-2$ in NL dictionary	24.991	3.457e-025	30
27	predicative form $i-2$ in NL dictionary	126.768	2.611e-132	20
28	pred-lem combi form $i-2$ in NL dictionary	28.762	3.676e-029	29
29	singular form $i-2$ in EN dictionary	49.909	1.635e-051	27
30	lemmatized form $i-2$ in EN dictionary	49.257	7.992e-051	28

Table 5.1: Numbers, descriptions, univariate score, p-value and rank per feature (top 10 in boldface).

Feature analysis

To analyse the predictive strength for every feature, univariate statistical tests are performed on the development set. Per feature the ANOVA F_1-score is determined similar to the calculation in Section 4.3.2. All information is found in Table 5.1.

The features have highly significant predictive power with $p < 0.001$. The most predictive is feature #3 (whether i is SMT); this probably has to do with the fact that the SMT test can filter out the majority of SMT tokens. Furthermore, there does not exist any overlap between the SMT class and other classes.

Due to existing overlap between dictionaries, training features concerned with dictionary lookup do not give exclusive values. Second, third and fourth best features are #4, #15 and #13; indicating whether the original form, predicative form and singular form of i are found in the Dutch dictionary. The high scoring features next in line are the occurrence in the Dutch dictionary of $i-1$ and $i-2$ and the length of i (#9, #11 and #1). Surprisingly, whether i is found in the South-African dictionary (#8) yields a higher F_1-score, compared to its occurrence in the English dictionary (#5). A likely explanation is that a tweet containing South-African words is most probably South-African in total. No mixed Dutch and South-African tweet exists in the Annotated data set. Contrary, English most often occurs mixed with Dutch in tweets. Finally, the tenth best feature for the language identification task is feature #10, indicating if the original form of $i-1$ exists in the English dictionary.

The feature scoring worst is #26, i.e. the lemmatized word form of $i-2$ in the Dutch dictionary. Next in line of relatively bad scoring features are #28-#30, which all have to do with both Dutch and English dictionary lookup for various word forms of $i-2$. Apparently, this information is of less importance compared to clues closer to i.

5.3.7 Decision Tree classifier model

The purpose of a Decision Tree classifier is to predict a label based on simple probabilistic decision rules inferred and learned from the provided data features. This model is the same as the one described in Chapter 4, Section 4.4.2. The decision tree model is available at scikit-learn.[15]

5.3.8 Support Vector Machine (SVM)

A Support Vector Machine (SVM) divides the different classes by use of planes in a multi-dimensional environment. As the labels do not represent perfectly separable classes a 1-slack soft-margin formulation is used for learning. This model is the same as the one decribed in Chapter 4, Section 4.4.3. Both model and learner are presented by the Pystruct library.[16]

5.3.9 Conditional Random Field (CRF) chain model

A chain CRF model is one of the most commonly used models for structured labelling tasks in natural language processing. In this approach each token is taken to be a node in a chain, connected with its neighbouring tokens through an edge. It is not needed to specify the direction of the edges as feature input is assumed to be aligned according to the nodes in the chain. The length of the chain fluctuates according to the number of tokens in the tweet. Also, it is assumed that all nodes have the same meaning, in the sense that each node has the same number of classes which are bearing the same weights.

[15] http://scikit-learn.org/0.17/index.html
[16] https://pystruct.github.io/index.html

The learner is based on a Viterbi algorithm, i.e. an efficient method to find the max-product in the passage of the nodes in the chain. Both model and learner are made available at PyStruct.[17]

5.4 Evaluation of performance

Since the data set is very skewed and a total of 6 classes are used, accuracy does not provide enough detailed information about model performance. In case all tokens are labelled as Dutch, the accuracy is 86%. While a program that identifies tokens as being either Dutch or SMT, has an accuracy of about 95%. This does not only show that the developed tool to recognize SMTs functions well, but also that a more fine grained method is needed to measure the classification performance of the less frequent classes. Hence, I concentrate on precision, recall and F_1-score per class.

Since, in this thesis, the main focus lies on the recognition of English words, most attention goes to the classes EN and MIX. However, as MIX is a particularly small group (0.05% of all tokens), even compared to the small group of EN labels (about 1.4% of all tokens), performance on identifying the EN labels is therefore of highest importance. A similar choice is made for the classes OTH and UNC; these labels lie somewhat out of the scope, therefore their performance results cannot be decisive.

In context of the language identification task, precision stands for the percentage of truly predicted language labels of a certain class among the total collection of predicted language labels of that class. Recall, embodies the percentage of truly predicted language labels of a certain class among the total set of items actually belonging to that class. Similar to Section 4.5, F_1-score (the harmonic mean of precision and recall) is calculated.

5.5 Results: language identification

Training and testing was performed in a 5-fold cross-validation setting using 90% of the annotated corpus (for means of comparison, this approach was also followed for the baseline and RBDL model; naturally without training part). The remaining 10% of the corpus was set aside for development. Labels were divided evenly over the five folds to deal better with the skewed label distribution. Model performance is measured with precision, recall and F_1-score on automatically allocated language labels per class, as described in the previous section. All results of the baseline model and the Rule Based Dictionary Lookup (RBDL) model can be found in Table 5.2. A selection of English, mixed and social media term classes is shown of the three machine learning models (Support Vector Machine (SVM), Decision Tree and Conditional Random Field (CRF)) in Tables 5.3 to 5.5.

[17] https://pystruct.github.io/index.html

As expected, the baseline model, that uses an off-the-shelf language identification tool combined with the manually developed SMT recognition tool, only scores well on NL and SMT classes. All unrecognised words are set to Dutch, and since this is by far the largest class, this is a smart heuristic approach.[18] High scores on SMT classification (i.e. $> 91\%$) show that the SMT identification tool functions well.

	Baseline			**RBDL**		
Label	p	r	f1	p	r	f1
NL	.962	.823	.887	.976	.971	.973
EN	.033	.116	.052	.326	.465	.382
MIX	.000	.000	.000	.142	.500	.218
SMT	.945	.917	.931	.937	.939	.938
OTH	.004	.076	.008	.080	.105	.091
UNC	.000	.000	.000	.000	.000	.000

Table 5.2: Results of language identification by baseline and Rule Based Dictionary Lookup (RBDL) model.

If we look at the second half of Table 5.2, we see that, except for the UNC class on which all outcomes are 0%, the RBDL model has better F_1-scores on all levels. In case of the classes NL, EN and MIX scores on precision, recall and F_1-score are all significantly better (paired t-test $p < 0.05$) compared to the results of baseline model.

In Tables 5.3 to 5.5 we can find the results for the three ML models: Support Vector Machine (SVM), Decision Tree and Conditional Random Field (CRF). Of these three models only the first two are bound to the exact constraints, i.e. have no information about what follows after the current token: needed to function together with the code-switch prediction model in Chapter 4. The latter CRF model does in fact have this information and is just added for comparison purposes.

Tables 5.3 to 5.5 cover the results on the EN, MIX and SMT classes. Per label three test settings are shown, namely: training on the *Top 10* and *Top 20* best scoring features, plus training on all features. Evaluation of intermediate feature sets are not shown here.

Scores on the Dutch label are also not shown, since all ML models yielded in all test settings between *Top 10* and *All Features* a precision, recall and F_1-score of $> 97\%$ and $< 99\%$; of which recall and F_1-score are all significant improvements in the identification of Dutch labels (paired t-test $p < 0.05$). As the set of outcomes does not add extra information, these are not displayed in a separate Table.

In Table 5.3 we can find the results on performance of English word identification. A first thing to notice is the Decision Tree performs best when trained on the best 10 features, contrary to SVM and CRF, which both improve the

[18] Without using this heuristic in the baseline model F_1-score for the NL label is 39%.

more features are added. The chained CRF yields the best F_1-score of 33.2%; not surprising since the CRF model has access to context information prohibited for SVM and Decision Tree. Although better, the improvement is not significant compared to the best F_1-scores of the other ML models.

As noted, the Decision Tree model performs best when trained on the top 10 features; leading us to the question whether there is even better performance in case the Decision Tree is trained on either less or more features. Therefore, some extra tests were carried out. The Decision Tree model achieves the best overall performance when trained on the 12 best ranked features. Precision, recall and F_1-score are 44.9%, 21.3% and 28.8% respectively. Still this F_1-score does not surpass the best scores of either CRF or SVM.

	SVM			**Decision Tree**			**CRF**		
EN label	p	r	f1	p	r	f1	p	r	f1
Top10	.237	.441	.307	.441	.188	.261	.427	.179	.233
Top20	.262	.441	.327	.313	.225	.257	.357	.297	.306
All features	.269	.433	.329	.236	.221	.224	.370	.329	.332

Table 5.3: Results language identification EN tag

Now looking at performance on recognizing the MIX class (Table 5.4); we see an opposite pattern. Both SVM and CRF do not identify any members of the MIX class in any of the training settings. While the Decision Tree, trained on the best 20 features, yields 20% on precision, recall and F_1-score the like. Nonetheless, these results are not significantly better compared to the other ML models, probably due to a standard deviation of 24% overall.

	SVM			**Decision Tree**			**CRF**		
MIX label	p	r	f1	p	r	f1	p	r	f1
Top10	.000	.000	.000	.000	.000	.000	.000	.000	.000
Top20	.000	.000	.000	.200	.200	.200	.000	.000	.000
All features	.000	.000	.000	.150	.200	.167	.000	.000	.000

Table 5.4: Results language identification MIX tag

The ML models show quite some variation in performance when identification of the SMT tag is concerned. The CRFs F_1-scores are the least compared to the rest; though, not significantly worse. In fact no significant differences in F_1-scores are found between the best scoring ML models, neither in combination with the baseline model. This even includes the best results yielded for the Decision Tree model trained on the best 9 features, in which case F_1-score is 93.4%.

As all models have low performance in recognizing labels OTH and UNC (F_1-scores are all below 10%), combined with the fact that performance results on these classes are not decisive in this specific performance task (see Section 5.4); it was chosen to discard the Tables on OTH and UNC as well.

SMT label	SVM			Decision Tree			CRF		
	p	r	f1	p	r	f1	p	r	f1
Top10	.940	.926	.933	.942	.923	.932	.884	.940	.911
Top20	.939	.927	.933	.917	.917	.917	.892	.938	.914
All features	.939	.927	.933	.903	.911	.906	.904	.934	.918

Table 5.5: Results language identification SMT tag

5.6 Discussion of results

Taking the ML models aside first, we see that the Decision Tree model seems quite unstable. In order to identify EN labels it performs best when trained on the top 12 features; in case of identifying MIX labels, 20 features is optimal; and when SMTs are concerned, 9 features is best. Despite this imbalance, it has the best scores on identifying mixed words. Still though (as described above in Section 5.4), MIX embodies such a small class that good scores on this class alone are not taken to be decisive in choosing the best model. Inasmuch as the Decision Tree has the lowest results on identifying EN tags, the Decision Tree model appears least suited for the task.

Performance of the CRF model is lower than expected. It does have the best results on identification of English labels, but as it has access to more information compared to the other ML learners, one might have expected a little more. As the CRF is excluded beforehand due to its use of context information, the SVM is the best machine learning option for online language identification at word level.

Now, comparing all models evaluated in this Chapter, we see that the rule-based RBDL model yields the highest F_1-score for classes EN, MIX and SMT (even significantly better compared to SVM and CRF in identifying MIX (paired t-test $p < 0.05$); and compared to CRF in recognizing SMT (paired t-test $p < 0.05$)). To put it mildly, this outcome was unexpected. The ML models were added to improve on the manual non-probabilistic approach, by learning probabilistic associations between features and classes. The results, however, do not show improvement in performance. Maybe the RBDL model performs better because of the hierarchy of rules, inherent to a non-probabilistic rule-based approach, which might have a positive value in this specific context. Another explanation is that, though carefully chosen, the current set of features does not provide enough information to learn this difficult language identification task better than a non-probabilistic algorithm.

5.7 Combination of automatic language identification and CS prediction

In this Section the best language identification model, i.e. the Rule Based Dictionary Lookup model, is combined with the best code-switch prediction

model from Chapter 4, i.e. the Decision Tree model. The purpose is to find the best automated approach to predict code-switches from Dutch to English and vice versa in a Twitter data set in which Dutch is the dominant language. Expectations on performance of CS prediction are naturally lower compared to the idealized setting in Chapter 4, in which features were manually trained on manually annotated tokens.

5.7.1 Results

In Table 5.6 the results are shown for the Decision Tree model for CS prediction trained on the data set with automatic language identification. For comparison, the second half of the Table shows the results of the CS prediction model based on manual language identification (as found in Section 4.5).

As in Chapter 4, the baseline is the model trained solely on feature 2, i.e. language identification of n (see Table 4.1). Outcomes on performance for the Decision Tree based on automatic annotation is 0% for precision, recall and F_1-score. As scores are less compared with the results on performance by the same model based on manual annotation (although not significant); this is in line with expectations.

	DT + AUT			**DT + MAN**		
	p	r	f1	p	r	f1
Baseline	.000	.000	.000	.079	.104	.090
Top3(#2,8,10)	.202	.320	.246	.439	.507	.463

Table 5.6: Results of precision, recall and f1-score of code-switch prediction for the Decision Tree (DT) model based on automatically annotated data (AUT) and on manually annotated data (MAN).

The same holds for the results of performance when the Decision Tree is trained on the 3 top ranked features. First, performance is significantly better compared to the baseline for precision, recall and F_1-score (paired t-test $p < 0.01$). Secondly, as expected the results are significantly less on all levels (paired t-test $p < 0.01$) compared to performance outcomes yielded by the same model trained on manual annotated data.

5.8 Conclusion

Automatic language identification is a typical classification task. Here six labels were used to discern the classes Dutch, English, mixed, SMT, other and unclear. Moreover, execution of the task is constrained to passed information in order to be combined with the real time setting of the CS prediction task.

There are two main challenges for the models to overcome. First, the classification task is complicated due to the imbalanced partition of classes. Second, there exists much overlap between the English and Dutch languages. Of all NL

tagged words 48% appears in both Dutch and English dictionaries; for English tagged words this is 42%.

In total five models were evaluated. First, a baseline that uses an off-the-shelf language identification tool. Second, a non-probabilistic rule-based dictionary lookup (RBSL) model manually developed for the specific language identification task. Models three to five are all supervised ML models: a Support Vector Machine (SVM), a Decision Tree model and a Conditional Random Field (CRF) model (the latter is only added for comparison purposes as it does not comply with the language identification constraints).

A total of 30 features were used for training the ML models. All models were tested in a 5-fold cross validation setting. For performance measures precision, recall and F_1-score were calculated; with a view to recognition of English words, identification of EN labels is valued the highest.

Quite unexpected, the RBDL model performs best on identifying the EN class (F_1-score: 38.2%, which is significantly better compared to the baseline). Moreover this model performs better on identification of the MIX class (F_1-score: 21.8%, which is significantly better than the baseline and best SVM and CRF models).

Although the CRF model has access to context information surpassing the current token, it does not perform exceptionally better on the task. With 33.2% F_1-score on EN it is slightly better than the SVM (32.9%), but not the RBDL (38.2%).

Moreover, the SMT identification tool, either in a rule-based or feature-based setting, performs quite well. Since all models use this tool, all models perform well on SMT identification. Except for the CRF, best model versions yield an F_1-score $> 93\%$.

Finally, the model performing best on the automatic language identification task, i.e. the RBDL model, was combined with the model performing best on the CS prediction task, i.e. the Decision Tree model. Combining the two provide in an automated approach from start to end. As expected, results on performance show lower outcomes compared to the Decision Tree trained on a manually annotated set. Precision, recall and F_1-score are 20.1%, 32% and 24.6% respectively; which is significantly less. Clearly, model performance is not good enough to be applied in a real-life setting.

Chapter 6

Conclusions

6.1 Introduction

Code-switching, a multi-lingual phenomenon, disturbs extensively used language interpretation tools; most are set up for mono-linguistic input exclusively. Whereas the need for automatic language interpretation increases due to expanding popularity of online applications.

In this study, code-switching was explored between the closely related languages Dutch and English. Contrary to comparable research on CS, the focus is on the occurrence of English words within everyday Dutch, instead of a specific Dutch-English bilingual community.

This research covered five main stages. In the following sections I will consider them one by one. In the final Section future steps are discussed.

6.2 Stage one: Corpus collection

As CS is more likely to appear in an informal environment, such as social media, a Twitter Corpus composed of roughly 95,000 tweets was collected. The Annotated Corpus, a subset of 1,300 tweets, was manually annotated. Tokens were classified with six different labels: Dutch (NL), English (EN), mixed word (MIX), social media term (SMT), other language (OTH) and unclear (UNC). Reliability of manual annotation was measured by inter-annotator agreement over 100 tweets, yielding a Cohen's kappa of 94%, which indicates annotation to be highly reliable.

Distribution of the 19,464 labels is uneven; of all tokens the majority is Dutch with 86.4%, followed by SMTs with 11.5% and English with 1.4%. At tweet level, 98.8% of the messages contain Dutch tokens, 86.4% contain SMTs and 8.5% contain English tokens. The data confirms that Dutch is indeed the dominant language. Furthermore, I conclude that although the ratio of English words in Dutch tweets is rather low (1.4%), the affected set of tweets is

considerable (8.5%). This outcome confirms my hypothesis that Dutch-English code-switches occur on a regular basis within ordinary Dutch.

6.3 Stage two: Analysis

To my knowledge, no analysis exists yet of English-Dutch CS in everyday Dutch. The executed analysis should therefore be seen as preliminary investigation. The analysis covers two main forms of code-switching, namely: intra-sentential code-switches (characterized by a tweet consisting of Dutch and at least one English word) and morphological code-switches (characterized by a word comprised of Dutch and English morphemes).

A total of 136 intra-sentential code-switches were found. Around 26% of these switches has a length of 3 tokens or longer; most are multi-word expressions and named entities that are not fully integrated within the Dutch sentence structure. The largest group (53%), consists of a single English word, of which most are embedded within the Dutch framework. Code-switches of length 2 cover the final 21%. These CSs keep the middle between the characteristics of the other two groups.

Intra-morpheme CSs occur seldom, just 9 times. Most of these (67%) are verbs and ensue a distinctive pattern: an English stem combined with a Dutch verb conjugation morpheme ('ge-' or '-en'). In this way the English word is literally forced into the dominating Dutch structure.

Taking both intra-sentential and intra-morpheme CSs together, the data shows that in general the words used for code-switching are in majority nouns (38.1%), followed by verbs (15.9%) and adjectives (14.9%); all members of open class words. Words belonging to closed classes do exist in the data set, but none occurs alone; closed class members are always part of a noun or verb phrase. This result endorses Joshi's 1982 theory that code-switching is limited to open class items.

The data might imply that the average Dutch Twitter user is a non-fluent bilingual. Most of the CSs (76%) are emblematic code-switches (Poplack, 1980), which are relatively easy to apply. The corpus does not involve inter-sentential CSs, hence the share of emblematic switches might be even higher. Moreover, further research is needed to define more precisely what intimate vs emblematic CSs correspond to in Twitter data.

6.4 Stage three: CS prediction

The task of predicting code-switching at word level can be operationalized as a binary classification task, where each word boundary is classified as either involving a CS or not. Labels are unevenly distributed, which complicates classification. Another challenge is posed by the constraint that models are not allowed to have information about everything that follows on the current token, in order to preserve a real-time prediction structure. Three supervised

learning models were tested on this task: a Multinomial Naive Bayes model, a Decision Tree model and a Support Vector Machine. All models were trained in a 5-fold cross validation setting.

Ten features were developed, of which six features were selected for training, based on their ANOVA F-score during development. Models are evaluated by calculating precision, recall and F_1-score of the positive class. As baseline the models were tested on one feature (indicating the language of n). The model performing best on the CS prediction task is the Decision Tree model. It yields 43.9% on precision, 50.7% on recall and 46.3% on F_1-score. These results are significantly better compared to the baseline system and to the best NB and the best SVM classifier.

However, it is important to note that the CS prediction task was performed in an idealized setting, since the features were trained on manually labelled tokens.

6.5 Stage four: Language identification

The task of automatic language identification at word level is a typical multi-class classification task. Six labels, the same as in the manual classification task, were used to discern the classes. The task involves an extra condition in order to be combined with the CS prediction task: execution of the task is constrained to passed information.

There are two main challenges to overcome. First, the classification task is complicated due to the imbalanced partition of classes. Second, as a result of being closely related, there exists much overlap between the English and Dutch languages. Of all NL tagged words 48% appear in both dictionaries; for EN tagged words this is 42%.

In total five models were tested: a baseline that uses an off-the-shelf language identification tool; a non-probabilistic rule-based dictionary lookup (RBSL) model developed for this specific language identification task; and three supervised machine learners: a Support Vector Machine (SVM), a Decision Tree model and a Conditional Random Field (CRF) (the latter uses information surpassing the current token, hence it is only added for comparison purposes). All models were tested in a 5-fold cross validation setting.

A total of 30 features were used for training the ML models. The majority of these features is based on a rule-based SMT identification tool (used by baseline and RBDL) and a rule-based dictionary lookup tool (used by RBDL). The features are for training the ML models; the tools are developed for the other two models. Performance was measured with precision, recall and F_1-score; identification of English words is valued the highest in model comparison.

Rather unexpected, the RBDL model has the best performance when identification of English is concerned (F_1-score: 38.2%, which is significantly better compared to the baseline). Additionally, it has also the best results on identifying the MIX class (F_1-score: 21.8%, which is significantly better than the baseline and best SVM and CRF models).

The data also shows that the SMT identification tool, either in a rule-based or feature-based setting, performs quite well. All models use this tool and all models perform well on SMT identification. The best model versions, except the CRF, yield an F_1-score $> 93\%$.

6.6 Stage five: Combining tasks

The model performing best on the automatic language identification task (RBDL model) was conjoined with the model performing best on the CS prediction task (Decision Tree model). Combining the two offers an automated approach from start to end. As expected, results on performance show lower outcomes compared to the CS prediction model trained on a manually annotated set. Precision, recall and F_1-score are 20.1%, 32% and 24.6% respectively; which is significantly less.

Results are still faraway from what is demanded in a real-life setting. Further research is needed in order to extend analysis and improve model performance. In the next section I discuss some possible directions to take from here.

6.7 Future work

In this research only a limited set of models was tested for the challenging tasks of CS prediction and language identification at word level. A next step is to investigate performance of alternative models. One model of interest is a neural network, but as neural networks are greedy for data, either large corpus has to be manually annotated or automatic language identification on word level has to be improved first.

In Section 3.5 it was showed that PoS tags might play an important role in CS. It would be interesting to develop features that contain such information, but, to my knowledge, PoS tagging at word level, especially for Dutch, is currently quite underdeveloped. I expect machine learning models to perform better on the tasks of CS prediction and language identification, when such information is indeed provided. Therefore, further research in this area will be valuable.

The used corpus is not suited to study intra-sentential CS. Without this information we cannot get the full picture of how the different forms of CS are distributed. Moreover, these CS forms can provide information with regard to the level of CS complexity. Following Poplack (1980), the data suggests that average Dutch Twitter users might be non-fluent bilinguals (Section 3.5). This implication is of interest, because the data collection was not directed at a specific bilingual community, but instead on ordinary Dutch language users. Without more knowledge about the share of intra-sentential code-switches, it is not possible to provide more conclusive answers about the level of Dutch-English bilingualism. Therefore, further research, based on an alternative corpus in which al three forms of CS are represented, is eligible. Also, it is needed to

better define the meaning of emblematic and intimate code-switching within Twitter data. Additionally, discerning between intimate CS (complex to apply) and emblematic CS (relatively easy to apply), is quite difficult. Consequently, future work should involve at least two annotators for this job, to calculate inter-annotator agreement.

According to Broersma (2009), closely related languages that share many trigger words are more likely to induce code-switches than less related languages. A question that might follow is whether closely related languages also induce CSs of a higher complexity, while less or unrelated languages in turn lead to CSs of low complexity. To undertake such research, multiple language sets with distinctive relation levels have to be compared with each other.

Appendices

Appendix A

Guidelines manual annotation

The tweets to be annotated have undergone minimal preprocessing. Only duplicates and automatically generated messages were deleted. This to preserve the raw user input as found on social media and how this is encountered by other users. The idea is to computationally mimic the human as closely as possible. Note that further preprocessing is likely in later stages of the process.

The tweets are annotated on word level. Every word is assigned to one of six different labels: *Dutch, English, Mixed, Social Media Term, Other* and *Unclear*. In the following sections all labels are described in more detail. To save time in the labelling process, I developed an annotation tool that is freely available.[1] The application of this tool is described in the last paragraph of this section.

Label: Social media term (SMT)

Under the label of *Social Media terms* or SMTs ranges a collection of differing words involving URLs, @names, #hashtags, emoticons (e.g. #_#, :p, n_n) emoji's (e.g. ☺), words and abbreviations specifically used on social media (e.g. LOL, tbh, tweet) and onomatopoeia (e.g. hahaaaaa, pfff). Next follws a list of examples to get a better grasp of tokens with an SMT label:

(1) **@name1** Ja beste!
@name1 Yes lad!

(2) **#ikwilvoor1dag** Alles hebben wat ik wil.
#iwantforaday To own everything I want.

(3) De vakmannen aan het werk! **https://t.co/xxx**
The professionals to work! https://t.co/xxx

(4) ...ik mag columns gaan schrijven voor **mijnserie.nl**...
...I can write columns for myseries.nl...

(5) ...Het geeft me zo veel voldoening! **:-)**...
...It gives such a feeling of satisfaction! :-)...

[1] http://illc.uva.nl/∼raquel/data/CS_prediction.zip

(6) @name1 **lol** ja. en zijn paard hoe heet ie weer.
@name1 lol yes. and his horse what's its name.

(7) ...**ff** serieus ben jij dat??...
...wait a minute (abbreviation) seriously is that you??...

(8) ...zal ik zeker doen, jij ook **xxx**
%92 *...I will, you too xxx*

(9) Dat ik terug moet werken. **Nah!**
That I have to return to work Nah!

(10) ...IK HEB HEM NOG IK GA HEM NU LEKKER OP SMIKKELEN **HAHAHAHAHA**...
...I STILL HAVE IT AND I WILL TUCK INTO IT NOW HAHAHA-HAHA...

Note that once in a while, emoticons stick to a word without separation by a space. In those specific cases, only the word is concerned, while the emoticon is passed over.
A list of collected instances is provided with words and abbreviations specifically used on social media. Unfortunately, we should assume this list is far from complete. Not only because of rapid changes, but to my knowledge no such list is available online. Therefore I collected a word-list myself from different sources, see Appendix C.

Labels: Dutch (NL), English (EN) and Other (OTH)

Numbers and names are handled as embedded within the language it is surrounded with. This means that '3' might get different labels in different messages. For example in sentence ((11)a) '3' is labelled Dutch, in ((11)b) English and in ((11)c) Other.

(11) a. Ik zag 3 beren!
b. I saw 3 bears!
c. Vi 3 osos!

Other examples of numbers are '5,6' in (12) and '19:30' in (13):

(12) ...kom ik aan met een **5,6**...
...and then me having a 5,6...

(13) LIVESTREAM vanavond om **19.30** uur...
LIVESTREAM tonight at 19.30 hour...

Names should be taken in a broad sense, including names of persons, objects, events, places etc. For example in sentence (14) 'Name1', 'Name2', 'Business Cup' and 'SBS' are all names embedded in Dutch and therefore in this case all these names are labelled Dutch. Other examples of names are 'Microsoft', 'Ipad' and 'Volvo'. Names that are preceded by # or @ are labelled as *Social media term* (see description above).

(14) ...**Name1** vs **Name2** op **Business Cup**: Ik hou het maar op **SBS**...
...Name1 vs Name2 at Business Cup: I will go for SBS...

It is possible that a number or name occurs at the transition of one language to another (e.g. I ate 6 stroopwafels / I ate 6 Dutch cookies). In that case the annotator has to make a choice.
During annotation all punctuation is neglected. In some exceptional cases punctuation is surrounded by spaces, i.e. taken as a separate word; then the punctuation is handled the same way as numbers and names. See (15) and (16) where "..." and "-" are identified as Dutch respectively, see example (7). It should be noted that quotation marks are treated similarly, so if a Dutch sentence contains an English phrase within quotation marks, the tokens are labelled as English (together with the word they stick to).

(15) ...In juli met een extra Italië-Special **...** https://t.co/xxx
...In July with an extra Italy-Special ... https://t.co/xxx

(16) Name1 (51) uit Eindhoven **-** Een aanwezigheid van iemand...
Name1 (51) from Eindhoven - A presence of someone...

Capitals, or the lack of them, are neglected. The same holds for spelling mistakes. Tweets contain many clerical mistakes of which some may be on purpose; these are not labelled as *Unclear*, but the language label they were intended to belong to. So, 'evht' in (17) should be spelled 'echt' (meaning *really*), this mistake was probably unintended. Contrary, the error in (18) seems to be intended, the extra letters 'n' emphasise the word 'crashen' (*to crash*). Both words are thus labelled Dutch.

(17) ...tja als t **evht** zo is van die 18milj...
...well if its (really) the case about this 18mil...

(18) @name1 @name2 @name3 bruiloft **crashennnn**. Omg....
@name1 @name2 @name3 wedding crashinnnng. Omg....

Swearings are also included, for example the Dutch 'godverdomme' in (19).

(19) ...**godverdomme** ik ben zo typisch
...god damn I am such a character

Non-existing or invented, but clearly understandable words or word combinations, solely based on one particular language are also tagged accordingly (see *Mixed word (NL-EN)* for invented words combined of English and Dutch elements). Therefore, 'mc-donalds-verslaafden' in (20) is tagged as Dutch as and 'young-earthers' as English in (21).

(20) @name1 Ik ben omringd door **mc-donalds-verslaafden**...
@name1 I'm surrounded by mc-donalds-addicts...

(21) Ik schrik van de hoeveelheid **young-earthers** in mijn vriendenkring...
I am alarmed by the amount of young-earthers among my friends...

Note that all exclamations and emotional expressions are SMT.

Dutch (NL) The vast majority of words are labelled as *Dutch* (NL). When in doubt, the word is searched in the digital word list provided by OpenTaal[2] In particular, originally English words incorporated in Dutch are checked. For example, the words 'chick', 'chillen', 'happy' and 'pack' are all labelled Dutch.

English (EN) English words are labelled as *English* (EN). If it is unclear whether the encountered word is in fact English, it is looked up in an English word list, i.e. the Hunspell dictionary.[3] Since most of the used English in tweets is every day language, look-up is not often needed. Some extra examples of English labels:

(22) ...Had @name1 mooi gelijk. @name2 **was not amused** ...
...@name1 was right. @name2 was not amused ...

(23) ...Ach ja, **let's go!** #rw16
...Oh well, let's go! #rw16

(24) Misschien kan de NOS een volledig **retweet-based** model voor publiek gefinancierde journalistiek ontwikkelen? ...
Maybe the NOS can develop a retweet-based model for publicly financed journalism?

(25) ...die ijslandrs gooiden letterlijk met die bal egt **im cryin.....**
....those icelandics literally threw that ball im cryin.....

(26) ...voor wie daar achter zit ik weet is 1 van me **peersandi despiseit**
...who is behind this I know is 1 of my peersandi despiseit (peers and I despise it)

Special attention should be given to words that fall in the category of *Social media terms*; many are English based or English abbreviations.

Other language (OTH) Words of any other language, besides Dutch or English, are labelled as *Other* (OTH).

(27) Strijdlied **du jour:** Aan de strijders:...
Battle song of the day: To all warriors:...

(28) **'n Gedeelte van 'n gebou by die Sable Square-winkelsentrum...**
part of a building near to the Sable Square shopping mall...

Words from other languages besides Dutch or English appear to be quite rare.

[2]See http://www.opentaal.org/
[3]http://wordlist.aspell.net/dicts/

Label: Mixed word (NL-EN)

The label of a Dutch-English mixed word encompasses a very specific group of words, namely a word containing a CS at the level of morphemes. These words do not occur in either English or Dutch dictionary, but their parts do.

(29) @name1 ja dat is zo kapot irritant, moet je de game weer **restarten** enzo
@name1 that's so incredibly irritating, restart the game again

(30) Er wordt weer lustig er op los **geframed** door de NOS over #brexit...
Again at the NOS they are freely framing about #brexit...

(31) ...Je bent altijd welkom bij een **netwerkevent** zoals de lunch op dinsdag.
...You're always welcome at a network event like the lunch on Tuesday.

To recognize and use mixed words, one needs a good grasp of both Dutch and English languages.

Label: Unclear

Any word that does not seem to fall under any of the mentioned categories is labelled *Unclear*. In general this means that the word does not seem to be a term (often) used on social media and meaning and/or language is not clear, as shown in examples (32) to (34).

(32) @name1 ja, **katexofficial**
@name1 yes, katexofficial

(33) ...Net voor de start heeft Lotto-name1 nog eens de **v...** ...
...Just before start Lotto-name1 has again v... ...

(34) ...omdat dit mailadres onder een niet actieve klantaccount hangt.***AJ**
*...because this mailaddress is connected to a inactive user account.*AJ*

Appendix B

Guidelines annotation tool

The annotation tool is written in Python 2.7, one has to import termcolor in order to show the colours used for annotation in the terminal. Make sure the background colour of the terminal is not set to black, otherwise the index numbers of the words do not show up, since these numbers are explicitly coloured black themselves.

The program is run from the command line, as shown in EXAMPLE 1. Besides the name of the program, one has to provide an input file containing

line separated tweets or sentences to annotate, e.g. `tweets_to_annot.txt`, an output directory, e.g. `data/annot`, and the name of the annotator.

```
#EXAMPLE 1: Run the program
$ python annotation_tool.py tweets_to_annot.txt data/annot
    name
```

When the program starts a simple welcome message is shown (see EXAMPLE 2). The program will either start with the first line to annotate, or, if used before with the same name and in- and output files, the program starts at the first new line to annotate. In turn, the program is saved and stopped with CTRL+C (EXAMPLE 3). If the program is stopped when the annotation of a line is not fully completed, information concerning this particular line is not stored. The next time, when the program is started again, annotation will start at this line.

```
#EXAMPLE 2: Start of the annotation program annotation_tool.
    py
Welcome! Up for some sentence annotation?
Use ctrl+c to abort this programme; all fully annotated
    tweets will be saved.
When the programme is launched again with the same arguments
it will automatically start at the next tweet to be
    annotated.
Start at line:  222
.
.
.
```

```
#EXAMPLE 3: End of the annotation program by use of ctrl+c
.
.
.
KeyboardInterrupt
.
```

In EXAMPLE 4 the annotation of a line is shown (note that colours are used in this example that may not show well when printed in black and white). Suppose we want to annotate a new line. First, the line number is shown, in this case 265. On the next line the text is printed and per token an index is printed beneath it. Followed by a dictionary {}, which is still empty and the words `Start range:0` and `Stop range:...`.

```
#EXAMPLE 4: Annotate a tweet
265
@name1 Maar wel met mn ogen dicht...

0           1    2   3   4  5    6
{}
Start range: 0
Stop range:
```

Now annotation can begin. Suppose you want classify the first token as SMT. To do so one has to fill in `0` as stop range and `4` to select the SMT label (as shown in EXAMPLE 5). You have just annotated your first token.

Now, everything is printed again, but this time the dictionary is filled with the known information up to this point. Moreover, the annotated index number is coloured corresponding to the chosen tag. Also, the new start range is already calculated (i.e. `1`).

To save time, words can be annotated in sequences. In the example the rest of the tokens is classified as NL. In order to store the information one has to approve with `y`. If the annotation is declined, annotation of the same line will start over again. In this way a mistake can be mended.

```
#EXAMPLE 5: Annotate a tweet (continuation)
265
@name1 Maar wel met mn ogen dicht...

0           1    2   3   4  5    6
{}
Start range: 0
Stop range: 0
NL(1)/EN(2)/Mixed-word(NL-EN)(3)/Social Media Term (4)/Other
    (5)/Unclear(6): 4


265
{'0-0': 4}
@name1 Maar wel met mn ogen dicht...

0           1    2   3   4  5    6
Start range: 1
Stop range: 6
NL(1)/EN(2)/Mixed-word(NL-EN)(3)/Social Media Term (4)/Other
    (5)/Unclear(6): 1


265
{'0-0': 4, '1-6': 1}
@name1 Maar wel met mn ogen dicht...

0            1    2   3   4  5    6
Do you want to save this annotation (Y/N)? y
```

Appendix C

SMT list

Collected list of social media terms (SMTs):

&amp;	PV	DFTBA	IRL	TGIF
&gt;	ROI	DGAF	JK	Thx
&lt;	RSS	ELI5	kga	thnx
<3	RT	EM	L8	thanx
API	RTD	EML	LMAO	TIL
ava	SaaS	F2F	LMK	TL;DR
B2B	SEM	FaTH	LMS	TLDR
B2C	SEO	FBF	LOL	TMI
BL	SERP	FBO	LOLz	TTYL
CAN-SPAM	SM	ff	MCM	TTYN
cdc	SMB	FFS	MM	TTYS
CMGR	SMM	fk	ms	Tx
CMS	SMM	fkn	MT	Txt
CPC	SMO	FOMO	MTFBWY	vlgs
CPM	SMP	FTFY	NM	vdg
CR	SoLoMo	FUTAB	NSFL	w/
CRM	SOV	FYI	NSFW	WBU
CSS	TOS	G2G	NVM	WCW
CTA	UGC	GG	OAN	WDYMBT
CTR	UI	Gr8	OC	WOTD
CX	URL	GTG	OMG	wtf
DM	UV	GTR	OMW	xo
ESP	UX	gwn	OOTD	YMMV
FB	WOM	HBD	ORLY	YOLO
FTW	YT	HMB	OTP	YfkK
G+	AFAIK	HMU	POTD	YT
HT	AMA	HTH	PPL	Android
HTML	ASL	IANAD	plz	api
IG	b/c	IANAL	QOTD	app
IO	B4	ICYMI	ROFL	appen
ISP	BAE	IDC	ROFLMAO	astroturfing
KPI	bc	idd	SFW	blog
LI	BFF	IDK	SMH	blogpost
NAW	BRB	iig	srs	copyleft
P2P	BTAIM	IKR	suc6	cloud
PM	BTW	ILY	sws	crowdfunding
PPC	CC	IMHO	TBH	crowdsourcing
PR	DAE	IMO	TBT	CSR

DDoSsen	GPL	moblog	splogs	virtual
Digg	GPS	MySpace	streamen	conferencing
Drupal	hashtag	NGO	streaming	webcasting
dunno	insta	nptech	sub	webinar
ebook	instagram	OpenID	subs	WhatsApp
ebooks	IOS	permalink	tag	wi-fi
embedding	ipad	podcast	tags	widget
Facebook	iphone	podsafe	troll	wiki
feed	lagg	retweet	tweet	Wikipedia
Flickr	lifecasting	RSS	tweetup	word-of-
geotagging	livestream	RT	Twitter	mouth
gig	livestreaming	screencast	Twitterverse	WordPress
google	mashup	SEO	UGC	YouTube
Gov	metadata	snapchat	unconference	
2.0	microblogging	SMS	videoblog	

Bibliography

Baldwin, T., & Lui, M. (2010). Language identification: The long and the short of the matter. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (pp. 229–237).

Broersma, M. (2009). Triggered codeswitching between cognate languages. *Bilingualism: Language and Cognition*, *12*(04), 447–462.

Das, A., & Gambäck, B. (2015). Code-mixing in social media text: The last language identification frontier? *Revue TAL*, *54*(3).

Elfardy, H., & Diab, M. (2013). Sentence level dialect identification in arabic. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, vol. 2, (pp. 456–461). Sofia, Bulgaria: Association for Computational Linguistics.

Grefenstette, G. (1995). Comparing two language identification schemes. In *The proceedings of 3rd International Conference on Statistical Analysis of Textual Data (JADT 95)*, (pp. 1–6). Rome, Italy.

Joshi, A. K. (1982). Processing of sentences with intra-sentential code-switching. In *Proceedings of the 9th Conference on Computational Linguistics - Volume 1*, COLING '82, (pp. 145–150). Czechoslovakia: Academia Praha.

Koepp, C. (2016). *TwitterSearch Documentation - Release 1.0.0*. URL `https://media.readthedocs.org/pdf/twittersearch/latest/twittersearch.pdf`

Lipski, J. (1978). Code-switching and the problem of bilingual competence. In *Aspects of bilingualism*, (pp. 250–264). Hornbeam.

Myers-Scotton, C. (1997). *Duelling languages: Grammatical structure in codeswitching*. Oxford University Press.

Nguyen, D., Dogruöz, A. S., Rosé, C. P., & de Jong, F. (2015). Computational sociolinguistics: A survey. *CoRR*, *abs/1508.07544*.

Nguyen, D., & Doğruöz, A. S. (2013). 2013 conference on empirical methods in natural language processing, emnlp 2013. In *Word level language identification in online multilingual communication*, (pp. 857–862). USA: Association for Computational Linguistics.

Papalexakis, E., Nguyen, D., & Doğruöz, A. S. (2014). Predicting code-switching in multilingual communication for immigrant communities. In *Proceedings of The First Workshop on Computational Approaches to Code Switching at EMNLP 2014*, (pp. 42–50). Doha, Qatar: Association for Computational Linguistics.

Poplack, S. (1980). Sometimes I'll start a sentence in Spanish y termino en Español: toward a typology of code-switching. *Linguistics*, *18*(7/8), 581–618.

Romaine, S. (1995). *Bilingualism, 2nd Edition*. Blackwell Publishers.

Solorio, T., & Liu, Y. (2008). Learning to predict code-switching points. In *Empirical Methods on Natural Language Processing, EMNLP-2008*, (pp. 973–981). Honolulu, Hawaii: Association for Computational Linguistics.

Thomason, S. (2001). *Language contact, an introduction*. Edinburgh University Press.

Yılmaz, E., Andringa, M., Kingma, S., Dijkstra, J., Van der Kuip, F., Van de Velde, H., Kampstra, F., Algra, J., van den Heuvel, H., & van Leeuwen, D. (2016). A longitudinal bilingual Frisian-Dutch radio broadcast database designed for code-switching research. In *Proceedings of LREC*.