university of groningen

faculty of arts

Normalization for Dutch for improved POS tagging

Youri Schuur

Master thesis
Information Science
Youri Schuur
s2748428
March 17, 2020

ABSTRACT

Non-standard language in user generated content causes problems for many NLP tasks, for example Part-of-speech tagging. Various NLP researchers argue that normalizing the non-standard language into standard language is a possible solution. Multiple studies report improved scores for NLP tasks after normalizing English data. However, for other languages not much research has been done on normalizing non-standard language and what impact it has on NLP tasks. In this thesis, collected Dutch user generated content from three domains was normalized using a state-of-the-art normalization model after which both the original data and the normalized data was POS tagged and evaluated. The performance of the tagger improved by 44%, 26% and 13% for chats, SMS messages and tweets respectively. These results show that by normalizing Dutch user generated content the tagger was able to achieve higher accuracy scores. Thus, by normalizing non-standard Dutch language the performance of the POS tagger improved significantly.

CONTENTS

PREFACE

I would like to thank my supervisors prof. dr. G.J.M. van Noord and Rob van der Goot for instructing me during the process of me writing a thesis and for giving me the feedback I needed to improve and finish my thesis.

1 INTRODUCTION

The rise of social media has caused a lot of changes within language over the last decade. For computational linguistics, this is an interesting development. These changes to language causes a lot of problems for computational linguistics, that is why Eisenstein (2013) calls this "new" language which largely emerges online via social media *bad language*. However, since the data that are available on social media keeps growing, it has become a domain that computational linguistics can not ignore. Especially natural language processing (NLP) tasks are much more complicated nowadays. Often, NLP models are trained on standard language. An example of standard language is the *Wall Street Journal*. In the late 80s, the Penn Treebank Wall Street Journal corpus was commonly used as a NLP benchmark for many tasks (Part-of-speech-tagging, parsing). Plank (2016) argues that this perception about a standard language is biased and wonders what would have happened if NLP had emerged in the last decade. Maybe the social media data would be considered as standard? The almost unlimited data that is available on social media is very exciting. Derczynski et al. (2013) states that non-standard language is often *noisy*. The noise in the data has its consequences, many models can not achieve the accuracy scores that were achieved for standard language.

An interesting question is why language on social media is so different. Social media consists of Twitter, blogs, forums etc. Most of the time, researchers use Twitter as their source to collect data. Twitter is a micro blogging service and has replaced sources like SMS messages. An advantage of Twitter is that a lot of the the data is public, which makes it easy to collect the data. Users from every country and region can post something on social media. In contrary, the Penn Treebank Wall Street Journal was written en edited mostly by working-age men (Derczynski et al., 2013). Thus, it makes sense that these domains differ a lot. Eisenstein (2013) states that there are a few reasons why language on Twitter is different: people are unsure of the correct spellings, it is faster, it has become the norm and people want to represent their own dialect. For Twitter, another reason is the length limit of a tweet. As of 7 september 2017, tweets are allowed to have 280 characters. Before, only 140 characters were allowed in a tweet, therefore, words like *about* are being written as *bout*. Further, factors like emoticons, phrasal abbreviations (e.g., LOL, SMH) and lengthening (e.g., coooolll) all play roles in the **user generated content** (Eisenstein, 2013). Social variables also play a role in language like age and location. These social variables are often associated with many features that display user generated content. Social media users often use non-standard language to show an affiliation with a region or a city (Eisenstein, 2013).

The last few years, computational linguistics has done a great deal of research on how to deal with non-standard language in natural language processing. Simply annotating more data seemed the most evident way to solve this problem. However, Eisenstein (2013); Plank (2016) do not think this is the ideal solution. Eisenstein (2013) shows in his paper that the out-of-vocubalary (OOV) words rate increases not only every year, but every day. This suggests that language changes constantly and Eisenstein (2013) states that "We cannot annotate our way out of this language problem". Plank (2016) agrees on this and adds "What we annotate today might be very distant to what we need to process tomorrow".

Another way to deal with non-standard language is by *normalizing* the bad language. Eisenstein (2013); Plank (2016); Derczynski et al. (2013) were all interested in normalizing the OOV words to reduce the noise in social media data. (Lexical) normalization is the task of converting non-standard language (e.g., *wit*) into standard-language (e.g., *with*). van der Goot and van Noord (2017) have proven that their model is able to normalize non-standard language into standard language. This approach should be investigated more since it might increase performance on various NLP tasks. Therefore, for this thesis, I want to investigate whether the normalization of non-standard language found on social media can improve the performance on a NLP task. In this case, I will be focusing on one specific NLP task, Part-of-speech tagging of Dutch non-standard language. POS tagging is a key task within NLP and there are Dutch models that are able to reach an accuracy on POS tagging of around 90% [1]. However, there is very little known of how Dutch POS tagging models perform when tested on non-standard language. With this realisation I was able to form the following research question.

- To what extent can we improve POS tagging by normalizing Dutch user generated content?

To answer this research question, collected non-standard language from three domains will be normalized after which we will POS tag both the non-normalized data and the normalized data. Subsequently, we will evaluate the performance of the POS tagger for both datasets. Finally, the scores for both datasets will be compared and a conclusion is drawn on whether the performance of the tagger improved by normalizing the data. In order to conduct this experiment, multiple aspects will be discussed within this thesis. In the next chapter,

[1] https://spacy.io/models/nl

previous work on normalization and POS tagging will be handled. In the third chapter the data and the material will be discussed. In the first section of the third chapter, I will explain how the data was collected. Subsequently, I will discuss how the data was annotated. In chapter four the setup of the experiment will be addressed. The experiment consists of two parts. Firstly, the normalization model of van der Goot (2019) will be tested on the collected data from three different domains. Secondly, the original- and the normalized data will both be POS tagged. Finally, the scores for POS tagging on both the original non-normalized data and the normalized data will be compared to each other. The results of this experiment will be evaluated and discussed in chapter five. Ultimately, in chapter 6, the research question will be answered.

2 BACKGROUND

In this chapter I will look into previous research that has been done on the subjects that are of importance for my research. This section will be divided into two parts. Firstly, previous research on POS tagging will be addressed. Consequently, previous work on normalization will be discussed.

2.1 POS TAGGING

It is well known that Part-of-speech taggers are able to achieve scores around the 97% on news texts for a while now (Toutanova et al., 2003). We have seen multiple POS tagging techniques over the last decade. Statistical approaches (N-Gram,Hidden Markov Model) and transformation based approaches (Brill's tagger) have been implemented for POS tagging. Thus, a lot of research has been done on the POS tagging the last 2 decades.

In the previous chapter, I have already talked about the impact of social media on NLP tasks like POS tagging. The score on POS tagging mentioned above (97%) was achieved by using news texts as data input. POS tagging scores high in the nineties are almost always only achieved when the data consists of standard language (like news texts). But how do POS taggers perform when they have to deal with non-standard language? Various articles have shown how state-of-the-art natural language processing (NLP) systems perform signicantly worse on social media data (Eisenstein, 2013). In part-of-speech tagging, the accuracy of the Stanford tagger (Toutanova et al., 2003) decreases from 97% accuracy on Wall Street Journal text to 85% accuracy on Twitter (Gimpel et al., 2010). Further, Derczynski et al. (2013) looked into the problem of how we can overcome noisy data when POS tagging. In their article, they present a novel approach based on vote-constrained bootstrapping with unlabeled data. They also assigned prior probabilities to some tokens and handled unknown words and slang. By using these features, they achieved a POS tagging accuracy of 88.7%. They also reduced token error by 26.8% and sentence error by 12.2% (Derczynski et al., 2013). Furthermore, Plank (2016) addressed the problem of non-standard language in NLP. She used multiple domains in her research, for example the Wall Street Journal (WSJ) and Twitter. Some interesting results were found when she tested POS tagging **cross-domain**. When the POS tagger was trained on WSJ data and also tested on the WSJ data, naturally, very high scores were reached with the TnT and BILTY tagger (96.63% and 97.25% respectively). However, when those POS taggers were trained on WSJ data and tested on Twitter data the scores fell quite a bit. The TnT tagger only achieved a score of 65.98% and the BILTY tagger scored a little bit better with 66.37%. Interestingly, when Plank (2016) used another Twitter corpus, the BILTY tagger achieved scores around the 90% while trained on the WSJ data. This tells us that within a very broad domain like Twitter, multiple domains exists and, for that reason, it is very challenging to achieve good tagging scores on non-standard language data.

Very little research has been done on this subject for the Dutch language. I found an interesting article on cross-domain POS tagging by Hupkes and Bod (2016). However, instead of social media data, historical Dutch data was used for their research. They wanted to know whether historical Dutch texts can be tagged with a decent accuracy while trained on contemporary Dutch texts. Hupkes and Bod (2016) concluded that POS taggers for contemporary Ducth data do not work well for historical Dutch data. The POS tags for the historical Dutch data were only accurately tagged 60% of the time. An analysis of the errors showed that the decrease in accuracy is mostly caused by the fact that many of the words in the other domain are unknown to the tagger (Hupkes and Bod, 2016).

2.2 NORMALIZATION

In the previous section, various consequences that come with new non-standard language/user generated content have been discussed. Normalization could be part of a solution for the problem of non-standard language on social media. In this section, some previous work on normalizing data will be addressed.

The first normalization of user generated content experiments were focused on SMS texts. A dataset was annotated by Choudhury et al. (2007) for this domain. A Hidden Markov Model was used to model the word variation in the data based on characters and phonemic transcriptions. They used the Viterbi algorithm to find the most likely replacement for each position.

In the following years, the focus shifted to social media, especially Twitter because of the large amount of data that is available. Han and Baldwin (2011) were the first to experiment with the normalization of Twitter data. They created a corpus called the *LexNorm Corpus* which consisted of 549 annotated sentences with normalization on the word level. Han and Baldwin (2011) reported results of training a Support Vector Machine (SVM) which predicts whether a certain word needs normalization or not based on dependency tree context. The replacement candidates were generated by using a combination of word similarity, N-Gram possibilities (unigrams, bigrams) and dictionary lookup.

The last decade, various approaches have been benchmarked on the LexNorm corpus. Yang and Eisenstein (2013) experimented with a log-lineair model. They used a sequential Monte Carlo algorithm to determine feature expectation after which they would rank the candidates using the Viterbi algorithm. Li and Liu (2015) reached the highest accuracy on the LexNorm corpus. They used multiple normalization systems among them a machine translation system and a spell checker. Those systems all would suggest one candidate and a Viterbi algorithm was used to rank the candidates based on their POS tags.

Most of the research on normalizing non-standard language has been done for English. However, it is important that these kind of models also perform well for other languages. In this thesis, we are focusing on normalizing Dutch data. Not many reports are available online on the normalization of Dutch data. De Clercq et al. (2014) annotated a Dutch normalization corpus collected from three user generated domains; SMS messages, SNS messages (message board posts from a social networking site) and Tweets. They experimented with machine translation on a word and character level and saw a 20% increase of their BLEU score by testing the model on their annotated normalization corpus.

Schultz extended the work of De Clercq et al. (2014) by building a multi modular system to normalize Dutch data. Each module dealt with a different normalization problem. This system consists of machine translation models, look-up lists, spell checker and some more features. They reported that their system scored a 24.6% Word Error Rate (baseline) before the normalization and a 14.9% Word Error Rate after the normalization. Further, they also report an improvement on POS tagging. Before normalizing the data, the POS tagger achieved an accuracy of 66.1% while afterwards the tagger reached an accuracy of 73.5%.

Recently, van der Goot and van Noord (2017) proposed a supervised normalization system called MoNoise. They report that their system beats state-of-the-art benchmarks for normalization which is why this normalization system was also used to conduct to the experiment for this thesis. We will take a deeper look into this model in the next subsections. Their normalization model consists of two parts:

1. Candidate generation
2. Candidate ranking

Candidate generation

For the generation of the candidates, multiple modules are involved. Since no error detection steps are included, the **orginal token** has to be in the candidate list. The next module is is a skip-gram **word embeddings** model from word2vec (Mikolov et al., 2013). For each word, the top 40 most likely candidates in the vector space are found based on the cosine distance. Typographical errors were checked by the use of the **Aspell spell checker** (van der Goot, 2019) [1]. Similarly looking and sounding words are generated by looking at the combination of character edit distance and phonetic distance. Further, a **lookup-list** is used. This lookup-list consists of replacement pairs in the training data. When a word that occurs in this list is found, every normalization replacement that occurs in the training data is seen as a candidate. Abbreviations are very common in social media texts. A special module called **Word*** deals with this matter. The module simply looks in the Aspell dictionary for words that start with the same sequence of characters as the original word. This module is only activated if the word contains more than two characters, otherwise way too many candidates would be generated. The final module generates candidates by looking at word **splits**. The word is splitted on every possible position. If the split results in two words that are found in the Aspell dictionary, the words are added as candidate. The split module is only activated if the word contains at least three characters.

[1] https://packages.debian.org/sid/aspell-nl

Candidate ranking

For the ranking of the candidates, various features are used. For example, **original token** is a binary feature that indicates whether a candidate is the original token and **word embeddings** is a feature in which the cosine distance between the original token and the candidate is used. Further, the **aspell** feature returns a ranked list of correction candidates, the rank in this list is used as a feature. Another feature is the **Lookup-list**. This feature counts the occurrences of every correction pair in the training data. Furthermore, two **N-Grams** models calculate the unigram and bigram probabilities. The first N-Gram model is trained on Twitter data while the second model is trained on Wikipedia data. Two binary features are the **dictionary lookup** feature which checks whether the candidate is present in the aspell dictionary and the **character order** feature indicating if the characters of the original token occur in the same order in the candidate that is proposed. Finally, a binary feature is used to check whether the token contains any alphabetical characters. All features are explained in depth in the thesis of van der Goot (2019).

The system was tested on multiple test datasets and compared to multiple state-of-the-art benchmarks for normalization (see table 3). The normalization corpora van der Goot (2019) used to test MoNoise are described in table 1. "Caps" indicates whether capitals are considered in the corpus. "1-N" indicates whether one-to-many normalization was considered in the corpus. "% normalized" indicates what percentage of all the tokens in the corpus were in need of normalization.

Table 1: Comparison of the normalization corpora used to test MoNoise

Corpus	Words	Language	% Normalized	Caps	1-N
GhentNorm	12.901	NL	4.8	Yes	Yes
TweetNorm	13.542	ES	6.3	Yes	Yes
LexNorm2015	73.806	EN	9.1	Yes	No

The results of the tests on these corpora are shown in table 2. According to van der Goot (2019) the results of MoNoise can differ per domain/corpus due to several factors:

- Size of training data
- The raw data used to train word embeddings. Also: differences between train- and test data can impact the scores.
- Annotation guidelines: easy normalization replacements like very common abbreviations can impact the score.

Table 2: Results MoNoise on different normalization corpora

Corpus	Recall	Precision	ERR
GhentNorm	50.77	86.84	44.62
TweetNorm	37.09	90.05	35.86
LexNorm2015	80.58	91.98	76.15

Further, they also did an extrinsic evaluation, which is also interesting for this research. To test whether they could improve POS tagging by using normalization, a bidirectional LSTM POS tagger Bilty, trained and tested on the data from Li and Liu (2015) was preprocessed by MoNoise. This led to an improvement in accuracy of 1.24% (88.53 to 89.63). Finally, van der Goot and van Noord (2017) concluded that MoNoise is able to reach a new state-of-the-art for all benchmarks. MoNoise is also able to generalize over different annotated data. Candidates are often correcly generated by using spelling correction combined with word embeddings. To rank the candidates, van der Goot and van Noord (2017) states that a Random Forest Classifier (Breiman, 2001) is able to learn to generalize over multiple normalization actions fairly well.

Table 3: Results on test data compared to the previous state-of-the-art

Corpus	Prev. state-of-the-art	Metric	Prev	MoNoise
GhentNorm	Schultz et al.(2016)	WER	3.2	1.36
TweetNorm	Porta and Sancho (2013)	OOV-Precision	63.4	70.57
LexNorm2015	Li and Liu (2015)	Accuracy	87.58	87.63

3 DATA AND MATERIAL

In this chapter of this thesis, I will discuss the data that is used to investigate what the impact of normalization is on POS tagging. In order to do this, user generated content that can be normalized is needed. Then, both the original data as well as the normalized data will be POS tagged. By comparing the performance of the tagger on both these data sets to each other the impact of the normalization will be measured. This chapter consists of two sections. In the first section I will describe what my data collection looks like and how it was obtained. In the second section I will address how the data was annotated. In this section I will shortly go in to the annotation guidelines. Furthermore, annotator agreement scores will be evaluated in this section to check the quality of the annotations.

3.1 COLLECTION

In order to conduct my research, I had to collect a lot of noisy Dutch data. The SoNar New Media corpus consists of three domains, these domains are described below in subsection 3.1.1. I decided to collect 500 noisy sentences per domain for my research. Thus, in total, 1500 sentences were collected. The sentences were split in train, development and test sets. In order to collect noisy data, I used an algorithm that checked whether a token was an OOV (Out Of Vocubulary) word by using the Aspell Dutch dictionary. If the token was not found in the dictionary, it was classified as an OOV word. I only collected a sentence when it contained at least four OOV words. This way I could be sure that all collected sentences consisted of some type of noise. I also checked whether a token was English, by using the same method, thus, using the Aspell English dictionary. If a sentence contained four or more English words, the sentence was not collected since I was not interested in English sentences. By using this algorithm I filtered out a lot data, especially tweets. In fact, very little **usable** noisy tweets remained in this corpus (see *Tweets* in subsection SoNar New Media Corpus). Therefore, I ended up using data form two different corpora to collect all the sentences.

3.1.1 SoNar New Media Corpus

Via the **Institution of Dutch language** (Instituut voor de Nederlandse taal) I found an interesting corpus called the SoNar New Media Corpus. This corpus contains news media texts that were collected for the STEVIN-project SoNaR and is available for download at the website of the Institution of Dutch language [1]. The corpus consists of 3 domains: SMS messages, Chats and Tweets. The SMS messages were provided by individuals who gave permission for the use of their messages for the benefit of the corpus. The chats were collected via public Internet forums. Lastly, the tweets were obviously collected on Twitter. I chose to work with this corpus because it consists of 3 domains (SMS, Chats, Tweets) that are considered *social media*. As explained in the intro, noisy non-standard language is mainly used in social media. An important thing to note is that all three domains of the SoNar New Media Corpus also contain Flemish sentences. Furthermore, I also came across multiple dialects within the Dutch language in all three domains of the corpus. In the next subsections, we will take a look at each of the three domains of the SoNar New Media Corpus (Oostdijk et al., 2013). In table 4 the number of collected sentences from all three domains are shown. As you can see, I was able to collect only 381 noisy usable tweets. However, I wanted my data collection for all three domains to consist of 500 sentences. In subsection *Tweets* I will address this problem.

Table 4: Collected noisy sentences from SoNar New Media Corpus

	SMS	Chats	Tweets
Train dataset	300	300	**181**
Dev dataset	100	100	100
Test dataset	100	100	100
Total	500	500	**381**

SMS Messages

The SMS corpus in SoNaR is a collection of Short Message Service messages. These messages were collected in The Netherlands and Flaunders between September and December, 2011. Only messages that were sent by the contributor are included in the corpus. The SMS messages were tokenized with UCTO. UCTO is an advanced rule-based unicode aware tokenizer. UCTO tokenizes text files: it separates words from punctuation, and splits sentences. It offers several other basic preprocessing steps such as changing case that you can all use to make your text suited for further processing such as indexing, part-of-

[1] https://ivdnt.org/downloads/taalmaterialen/tstc-sonar-nieuwe-media-corpus-1

speech tagging, or machine translation (Maarten van Gompel, 2018). In total, this domain consisted of 723,876 words.

Chats

Besides SMS messages, the SoNar New Media Corpus also consisted of chats, which are real time typed conversations over a computer network between two or more people. Like the SMS messages, the chat messages were tokenized with UCTO. The chat data was collected from multiple sources. Most of the sources were Dutch, for example, MSN messages (A total of 1,056 chat sessions) were collected for this corpus. In total, the Dutch data in this domain counted 737,520 word tokens. However, not all data sources were Dutch. Also Flemish data was collected from chat.be [2]. This is a Flemish website from which chat messages were collected between March 4, 2011 and February 11, 2012. The chats are from the main chat channel of the site. This explains why this domain also consists of a lot of Flemish messages. Is is not reported how many tokens this Flemish corpus consists of.

Tweets

The third domain from the SoNar New Media Corpus I will address is Twitter. Only tweets that were publicly available were collected. In general, only Dutch tweets were collected. However, Oostdijk et al. (2013) states that "Some twitterers publish tweets both in Dutch and in another language, primarily English". Twitterers who only publish in non-Dutch were removed from the corpus, but no language detection was done to remove single tweets that are non-Dutch. As a consequence quite a few English tweets ended up in the SoNaR corpus. Like the SMS and chat messages, the tweets were tokenized with UCTO. With a total of 23,197,211 words, the Twitter corpus was by far the largest data source from the SoNar New Media Corpus. Although Twitter was the largest data source out of the three domains of SoNar, it was also by far the noisiest data source. Even though I needed noisy data for my research, most of the Twitter data was not useful for this research. Like Oostdijk et al. (2013) stated, the data consisted of many English tweets. Furthermore, when I looked into the data, I also came across many other languages besides Dutch and English in the Twitter data. A large amount of tweets were written in German for example. Therefore, I added a check for German tokens in the algorithm by looking whether the token existed in the Aspell German dictionary. After applying the algorithm, not enough usable tweets were retrieved (< 500). For this reason, the algorithm was slightly tweaked. Instead of collecting a tweet when it contained at least four OOV tokens, I started collecting tweets that contained at

[2] www.chat.be

least three OOV tokens. This way I was able to collect some more noisy usable tweets. However, this way I was only able to collect 381 tweets. This still were not enough tweets since I wanted my datatsets to exist of 500 instances for all three domains. Therefore, I decided I needed another data source to collect noisy Tweets. This data source is discussed in the next subsection.

3.1.2 RUG Tweets

In order to conduct my research, I had to collect more tweets. Fortunately, The LWP workspace on the computers at the University of Groningen (RUG) provides a corpus named twitter2 that contains Dutch tweets from 2019. This corpus was collected by Tjong Kim Sang (2013) and the methodology of this data collection is described in the article of Erik and van den Bosch (2013). They collected billions of tweets by using the filter of Twitter's streaming API. With this software it is possible to collect tweets based on keywords in the messages or the names of the users that sent the messages. For their collection, they were only interested in Dutch tweets. Two methods were used to collect tweets. First, they searched for 229 Dutch words/hashtags. Most of these words are common Dutch words that are used frequently. The second method to collect the tweets was to collect all tweets from 5.000 users who post tweets very frequently. It is only allowed to track 5.000 users on Twitter, thus, with this method not the maximum of messages that can be retrieved are being retrieved (50 tweets per second). These are two nice methods to collect a lot of tweets, however, sometimes non-Dutch messages still slip through the filter. In order to decrease the non-Dutch tweets a language checker was applied. They chose libTextCat as a language checker for this task. LibTextCat was developed by Frank van Scheelen from WiseGuys based on the work by Gertjan van Noord [3]. After the language was checked a simple Dutch tokenizer was used to separate the words from punctuation characters. The tokenizer also added a white space before and after every character in the tweet if that character was neither a digit nor a letter (Erik and van den Bosch, 2013).

Ultimately, I ended up collecting 119 tweets from the twitter2 corpus. All these tweets were sent in August 2019. Since 381 tweets were already collected from the SoNar New Media Corpus the complete twitter collection now contained 500 tweets. In table 5 some statistics about the collected data from the three different domains are shown.

[3] https://software.wise-guys.nl/libtextcat/

Table 5: Comparison of the collected data from the three different domains

Corpus	Words	Language	% Normalized	Caps	1-N
Chats	5.193	NL	49.6	Yes	Yes
SMS	5.645	NL	26.6	Yes	Yes
Tweets	12.901	NL	21.5	Yes	Yes

3.2 ANNOTATION

In this section the task of annotating Dutch user generated content will be addressed. In the first part of this section the created annotation guidelines for the normalization task will be discussed. To make sure these guidelines are consistent and clear for the annotators, inter annotator agreement scores were calculated. These scores are reported in the second part of this section.

3.2.1 Guidelines

Before training an automatic normalization model it is important to manually annotate the non-standard language into its standard form and create a gold standard. In order to standardize the manual normalization for Dutch user generated content, it is helpful to lay down some rules for the annotator. All these rules together form the annotation guidelines. One of the few articles in which annotation guidelines for normalization for Dutch user generated content have been reported is provided by De Clercq et al. (2014). Their annotation process consists of two parts. Firstly, the actual text normalization is handled by clearing tokenization problems, determining all needed operations to get the normalized word and to write down the actual full normalized word. The second part of the annotation process consists of flagging additional information that might be helpful for other language processing purposes. For researchers whose goal it is develop an improved normalization model for Dutch (or English) these annotation guidelines are excellent because they are very detailed. However, since this was not the goal for this research, the annotation guidelines that are used for my experiments are based on the more simplistic guidelines written by Baldwin et al. (2015a). These guidelines were created as part of the shared task of the 2015 workshop on noisy user generated text (Baldwin et al., 2015b). This task consists of normalizing English twitter messages randomly collected from public streaming API. Spelling errors ("commitee" for "committee"), informal abbreviations ("tmrw" for "tomorrow") and phonetic substitutions ("4eva" for "forever") are seen as non-standard words (NSW). If a NSW is seen in a tweet, the standard form should be given next to the NSW. The guidelines consists of 10 rules and are created for English normalization. In order to use these guidelines for Dutch normalization, these guidelines were slightly adjusted. For the complete annotation guidelines for this research I refer to *Appendix A*.

3.2.2 Inter annotator agreement scores

Often in linguistics, it is hard to determine a classical definition with necessary and sufficient conditions for certain categories. For this reason, calculating inner annotator agreement might be wise when classifying linguistic categories. A corpus annotator decides what annotations are being made. However, as user of the annotated data you want to know to what extent the annotations are actually correct and consistent. Inter annotator agreement is a measure of how well two (or more) annotators can make the same annotation decision for a certain category [4]. The inter annotator agreement score tells us whether the annotations are trustworthy. Further, it also could also tell us something about the difficulty of the annotation task. For this research, Cohen's Kappa was used to measure the inter agreement on whether a word should be normalized or not. For all three domains, the Kappa score was above 0.75 (0.91 , 0.77 and 0.80 for Chats, SMS and Tweets respectively) which is seen as a strong level of agreement [5]. Below in table 6 some examples of disagreements are displayed.

Table 6: Examples of disagreements per domain

	Original token	**Annotator 1**	**Annotator 2**
Chats	dynabyte	dynabyte	Dynabyte
SMS	bedje	bedje	bed
Tweets	vakantiecolonie	vakantiekolonie	vakantiecolonie

[4] https://corpuslinguisticmethods.wordpress.com/2014/01/15/what-is-inter-annotator-agreement/

[5] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/

4 METHOD

In this chapter I will address how I conducted my experiment. This chapter will be divided into two parts. Firstly, the normalization part of the experiment will be discussed. How was the data processed for the normalization system and how did I ran the system? In the second part of this chapter, the POS tagging part of the experiment will be explained. Thus, what POS tagger and tagset was used? And how was the alignment of words done when the length of a sentence of the tagged test data differed from the tagged gold data in order to evaluate the system?

4.1 NORMALIZATION

It has already been mentioned in this thesis that MoNoise (van der Goot and van Noord, 2017) was used as a model to normalize tokens. In order to run this model, collected data was needed to train and test the system. For more information about the data I refer to chapter 3. Further, the data also needed to be processed in a certain way. Below is shown how one line of the data file given as input for the normalization system should look like.

Table 7: Format of one line from datasets for MoNoise

Original token	<tab>	<placeholder>	<tab>	Normalized token

In this format the character 'X' was used as a placeholder. Note that a sentence contains multiple lines in the format described in table 7. Between the sentences a white line was added. Below in table 8 an example of a full sentence from the collected data is displayed.

Table 8: An example of a Dutch sentence processed for MoNoise

ik	<tab>	X	<tab>	Ik
gon	<tab>	X	<tab>	ga
ni	<tab>	X	<tab>	niet
blyve	<tab>	X	<tab>	blijven
prate	<tab>	X	<tab>	praten

After all the data was processed into the correct format it was time to run the normalization model. However, a lot of options are available when running the system. For example, if "-b" is added to the command line a bad spellers mode for Aspell is activated. This could lead to higher scores but uses a lot more energy and it takes a long time to run the system. For an overview of all the available features I refer to the MoNoise bitbucket page from Rob van der Goot [1]. For this research, the system was ran only with one additional option, namely, the -C option. The -C option makes the system consider capitals. By default this setting is turned off and the data is lower cased since most corpora do not use capitals in the gold data. However, I did consider capitals in the gold data. As for why no other options were applied, the goal of this research is not to maximize the normalization system MoNoise. The goal is to improve POS tagging by normalizing non-standard language into standard language. For this reason, I thought it was best to keep the normalization model as simple as possible.

4.2 POS TAGGING

In this section the POS tagger and tagset that was used to tag the data will be discussed. Further, an issue involving the alignment of POS tags will be addressed.

For the task of POS tagging, various simple tools are available publicly. Two well-known tools for NLP tasks are NLTK and SpaCy. NLTK has been the standard tool for NLP tasks in Python for a long time while spaCy is a newer tool which has become very popular over the last few years. Both NLTK and spaCy provide stochastic taggers. Furthermore, both taggers support multiple languages, including Dutch. NLTK lets you customize your model if needed. SpaCy, in contrary, implements a single algorithm which may be replaced with an improved algorithm as the state of the art improves. This makes it extremely easy to implement spaCy and since no customization was needed for this research I decided to go with the spaCy tagger. Further, it is proven that the spaCy dramatically out-performs NLTK in POS tagging [2]. SpaCy provides a Dutch model trained on the Lassy Universal Dependencies corpus (Bouma and Van Noord, 2017). This corpus consists of written texts (news, media). The tagset that is used in the Dutch model is the Universal part-of-speech tagset [3]. These tags are all derived from the Universal Dependencies (UD) scheme [4]. This tagset is very basic and consists only of the most common tags. However, this was perfect since I did not want to over-complicate the task of POS tagging non-standard language. In the the collected Twit-

[1] https://bitbucket.org/robvanderg/monoise/src/master/
[2] https://blog.thedataincubator.com/2016/04/nltk-vs-spacy-natural-language-processing-in-python/
[3] https://spacy.io/api/annotation
[4] https://universaldependencies.org/u/pos/index.html

ter data, quite a few tokens existed of usernames and url's. It is easy to detect these type of instances since usernames always start with a "@" and url's always start with "http". Therefore, these instances were given the tag "USER" or "URL". This results in better POS tagging scores for the tweets. However, this is the case for both the non-normalized and the normalized data so the difference in scores between these two datasets will be the same.

When evaluating the tagging performance on a certain test dataset, this dataset should contain exactly the same amount of tokens as the gold dataset, otherwise the tags can't be compared. This is a problem when 1-N normalization is considered in the annotated data. For example the token *kga* which means "I'm going to" in Dutch should be normalized to *Ik ga*. In the non-normalized data this token gets one tag but in gold normalization data these tokens both get a tag. However, now the tags of the tokens can not be aligned. To solve this problem the POS annotation was manually splitted and aligned. Thus, if a token is splitted in the gold data this should also be the case for the non-normalized and normalized data. Since the test datasets were relative small (100 sentences from each domain) it was easiest to do this manually. However, if more test data is needed for a similar experiment, manually splitting and aligning might not be the best solution for this problem. The number of necessary alignments made before- and after normalizing the test data per domain are shown in the table below.

Table 9: Alignments made in test data per domain

	Before normalization	After normalization
Chats	101	54
SMS	102	37
Tweets	73	35

5 EVALUATION

In this chapter the results of the experiments will be evaluated. This chapter will be divided into two parts. In the first part, the performance of the normalization system (MoNoise) will be discussed. In the second part, the focus is shifted towards POS tagging. In this part the performance of the POS tagger before- and after normalizing the test data will be evaluated.

5.1 NORMALIZATION

In this section MoNoise (van der Goot and van Noord, 2017) will be evaluated on test data sets from three different domains; Chat messages, SMS messages and Tweets. In order to compare the results to state-of-the-art benchmarks we will use the same metrics as van der Goot and van Noord (2017) used. In the next subsection, these metrics will be discussed. In the second subsection, we will take a look at how MoNoise scored on the datasets that were collected for this research.

5.1.1 Evaluation metrics

Since there is no clear consensus on what metric to use to measure performance on normalization tasks, multiple metrics are being used to evaluate MoNoise. Some normalization models use word error rate and/or character error rate as an evaluation metric. These evaluation metrics are based on the Levenshtein distance (Levenshtein, 1966). To calculate the word error rate the minimum number of substitutions, insertions and deletions to transform the word to the output word (Levenshtein distance) is divided by the total number of words in the annotated normalized data. The character error rate is is calculated the same way, but with characters considered as units. According to van der Goot (2019) these metrics only are relevant when splitting and merging is annotated. However, van der Goot (2019) also states that for datasets which do include annotated splitting and merging, these metrics still are questionable since 1-N replacements are weighted heavier whereas it's not always clear that these specific instances are more important.

A well know metric is the **F1-score**. The F1-score is the harmonic mean between **precision** and **recall**. To understand what this score actually means, first, some concepts which define this score will be explained. In order to calculate the precision and recall, instances of the test data are classified as one of the four groups below (van der Goot, 2019).

- True Positives *(TP)*: Annotators normalized, system normalized correctly
- True Negatives *(TN)*: Annotators did not normalize, system did not normalize
- False Positives *(FP)*: Annotators did not normalize, system normalized
- False Negatives *(FN)*: Annotators normalized, but system did not find the correct normalization. This could be because it kept the original word, or proposed a wrong candidate

To determine the precision, we calculate how many replacements were made by the normalization system correctly out of all the replacements made.

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

The precision is calculated differently in some previous work. When the system normalizes a word to the wrong word, the precision score is not punished. In some previous work (van der Goot and van Noord, 2017) , these cases were being considered in both the precision (FP) and the recall (FN). However, van der Goot (2019) states that these cases should be only be considered in the recall, otherwise the replacement counts double, while the decision whether to normalize the anomaly or not is actually correct.

The recall is determined by looking at how many words were correctly normalized by the normalization system out of all the words.

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

With these two scores, it is now possible to calculate the harmonic mean of these two metrics.

$$\mathrm{F1} = 2 * \frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

The average of the scores increase when the scores are closer to each other. Thus, it rewards systems that score well on both precision and recall instead of scoring extremely well on one of them and extremely poor on the other.

van der Goot (2019) also proposes a new metric in his work to evaluate a normalization task. The **Error Reduction Rate** (ERR) is the accuracy normalized for the number of tokens that need to be normalized. With this metric, the difficulty of the task is being taken into account. A downside of accuracy is that it is very hard to compare it between multiple datasets. An accuracy of above the 90% might be very good on one dataset while it is completely meaningless on another dataset. Hence, van der Goot (2019) thought of this new metric. Below the formula of the Error Reduction Rate is displayed.

$$\mathrm{ERR} = \frac{\mathrm{TP} - \mathrm{FP}}{\mathrm{TP} + \mathrm{FN}}$$

Usually, the ERR is between 0.0 and 1.0. An ERR below 0.0 means that the model normalized more words wrongly than correctly. A score of 0.0 can be seen as a baseline score and a system that reaches 1.0 can been seen as the score of a perfect system.

All in all, the use of the Error Reduction Rate metric has multiple advantages. It is easily comparable between multiple datasets since ERR normalizes for the number of tokens that need to be normalized. Also, it is easy to interpret the ERR since it shows the percentage of how much the problem is solved. Finally, it evaluates all of the normalization task since ERR includes the impact of the error detection step. van der Goot (2019) considers the Error Reduction Rate as the main evaluation metric for the system MoNoise. However, the ERR does not tell us whether the system normalizes too careful or too aggressively. Therefore, precision and recall are also used to evaluate MoNoise.

5.1.2 Evaluation MoNoise on noisy Dutch data

In the last subsection we have seen how the normalization system MoNoise is evaluated. In this subsection the results of how MoNoise performed on the collected 'noisy' data will be addressed. Remember that the data is collected from three domains; chat messages, SMS messages and Twitter. MoNoise was evaluated on these three domains. The system was tested on data from the same domain as it was trained on. Thus, no cross-domain experiments were done for this research. For each domain the results of the normalization will be discussed. Further, these results will be compared to the results we have seen from van der Goot (2019) in the chapter *background*.

For all three domains, the precision and recall scores were quite similar. In all cases, the precision tops the recall. This is arguably what we want since the system should not replace 'correct' words (van der Goot, 2019). Especially for the chat- and SMS messages MoNoise performed equally well. This was to be expected since these type of messages are very similar in contrast to for example Tweets. Both type of messages usually contain less than 15 words per message. The size of these corpora are also quite similar. As stated in the chapter *data and material*, the chats corpus consists of 5193 words while the SMS corpus contains 5645 words. The only difference between the results of these two domains lies in the WER (Word Error Rate). MoNoise scored a WER of 29.72 on the chats while the system scored a WER of 17.15 on the SMS messages. Earlier in this chapter, the WER was already discussed. We saw that 1-N replacements are weighted heavier when calculating the WER. This explains the difference between the WER scores between these two domains. In the SMS corpus 1-N replacements occurred in 6.3% of all the words. However, in the chat corpus 1-N replacements occurred in 10.4% of all the words. Besides the precision and recall, the ERR (Error Reduction Rate) for both corpora also lie extremely close to each other. If we look at the scores reported in the work of van der Goot (2019), we can see that the ERR for the chats- and SMS corpora (+- 46) are very similar to the ERR of the GhentNorm corpus (44.62). The GhentNorm corpus, like the chats- and SMS corpora, also consists of Dutch data. Thus, it is not rare that these scores are quite similar. However, the size of the chats- and SMS corpora (5193 words and 5645 words respectively) is not half the size of the GhentNorm corpus (12,901 words). So despite there was a lot less training data, the system was able to reach a higher ERR. Naturally, compared to larger English corpora like the LexNorm corpus, the difference in ERR score was pretty big (76.15 > +- 46).

Table 10: Results MoNoise on three domains

	Recall	Precision	F1	ERR
Chats	48.81	69.71	57.42	46.44
SMS	49.62	67.36	57.14	46.80
Tweets	40.76	72.25	52.12	36.25

If we compare the Tweets to the other two domains, we can see a decrease in ERR of 22%. First, this decrease seemed surprising. However, compared to scores on the TweetNorm corpus (Porta and Sancho, 2013) reported in the thesis from van der Goot (2019) the scores actually lie really close to each other. MoNoise scored an ERR of 35.86 on the TweetNorm corpus while it scored an ERR 36.25 on the Tweets that were collected for this research. The TweetNorm corpus consists of collected Spanish Tweets. Despite the difference in language the scores were still very similar. The size of the training data probably is not the cause for this decrease since there are far more words in the Twitter corpus compared to the chats and SMS corpora. Also, all three domains are annotated based on the same annotation guidelines. Therefore, the decrease in ERR for both these corpora could mean that raw data collected via Twitter is more complicated to normalize. Further, the WER for the tweets (12.6) is considerably low compared to the other domains (29.72 for chats and 17.15 for SMS). As stated earlier in this chapter, the drop in WER probably has to do with the fact that the tweets contain way less 1-N replacements (only in 2.3% of all the words). Thus, this decrease in WER seems logical. To illustrate the effect of the normalization model, below three examples (one from each domain) are displayed. Sentence *a* is the original sentence while sentence *b* is the normalized sentence.

- **Chats**
  a) is t ni goe mss ArAgOnnn ? :p
  b) *Is het niet goed misschien* ArAgOnnn ? :p
- **SMS**
  a) K ga drna na tv kijke
  b) *Ik ga daarna naar* tv *kijken*
- **Twitter**
  a) Ik was vannacht te gast bij @WordtNuLaat op @NPORadio2 , oa over keuzes mbt columnisten , interviews en covers
  b) Ik was vannacht te gast bij @WordtNuLaat op @NPORadio2 , *onder andere* over keuzes *met betrekking tot* columnisten , interviews en covers

5.2 POS TAGGING

In this section the POS tagged normalized- and non-normalized data is evaluated and the accuracy of the POS tags will be compared to each other. This will be done for all three domains. Further, we will look at some generated classification reports for all three domains to gain some insight in what tags are being predicted correctly and what tags are being predicted incorrectly. Finally, a comparison is drawn between the tagged normalized data and the tagged gold data. The results are shown in table 11 and will be discussed per domain.

Table 11: POS tagging accuracy on three datasets per domain

	Non-normalized	Normalized	Gold Normalization
Chats	42.1	60.8	71.8
SMS	59.8	75.5	81.7
Tweets	64.3	72.5	76.1

5.2.1 Chats

The first domain that will be addressed is the chats domain. This domain has the smallest corpus out of the three domains. This is likely the reason why the accuracy of the POS tags for the non-normalized test set was only 42.1%. This is obviously a really poor score. However, the accuracy of the POS tags for the normalized test data jumped to 60.8%. Thus, by normalizing the data, the POS tagger performed almost 45% better. This is a big improvement. It would also be interesting to see which tags are being predicted correctly and which tags are being predicted incorrectly. Therefore, I generated a classification report of the tagged normalized chats which is shown in table 10.

By looking at the classification report of the tagged normalized chats, I found that that the classes DET (determiner), PRON (pronoun), PUNCT (punctuation) and VERB (verb) were all tagged with fairly high precision and recall and this resulted in a F1-scores above 0.75. ADP (adposition) instances were predicted with high precision (0.82) but low recall (0.17). In contrary, the classes ADJ (adjective), CONJ (conjunction), NOUN (noun) and NUM (numeral) reached decent recall scores (> 0.7) but poor precision (< 0.45). The tagger preformed extremely poor on the X (other) instances with a precision of 0.02 and recall of 0.12. This is not strange since this tag is almost never used. Universal Dependecies (UD) also state that "the tag X is used for words that for some reason cannot be assigned a real part-of-speech category. It should be used very restrictively". [1] The classes PROPN

[1] https://universaldependencies.org/u/pos/X.html

Table 12: Classification report of tagged normalized chats

POS Tag	Precision	Recall	F1	Support
ADJ	0.41	0.71	0.52	41
ADP	0.82	0.17	0.29	52
ADV	0.69	0.64	0.66	196
CONJ	0.38	0.71	0.49	34
DET	0.90	0.80	0.85	35
INTJ	0.00	0.00	0.00	54
NOUN	0.35	0.81	0.49	124
NUM	0.43	1.00	0.60	6
PRON	0.87	0.71	0.78	175
PROPN	0.00	0.00	0.00	48
PUNCT	1.00	1.00	1.00	49
SYM	0.00	0.00	0.00	42
VERB	0.89	0.73	0.80	223
X	0.02	0.12	0.04	8

(proper noun), SYM (symbol) and INTJ (interjection) were not correctly predicted once. For the class INTJ this was not a big surprise to me since this is also a class that is very hard to define. UD explains that "an interjection typically expresses an emotional reaction, is not syntactically related to other accompanying expressions, and may include a combination of sounds not otherwise found in the language" [2]. For example, words like *"psss"* or *"ouch"* can be classified as interjections. However, I was surprised with the PROPN instances. The test data contained some obvious proper nouns, for example *"Trump"* and *"Iran"*, but the tagger was not able to correctly tag these instances (examples above were tagged as NOUNS).

Besides comparing the the tagging performance on the normalized data to the non-normalized data, it is also interesting to see how the tagger performed on the gold data and compare it to the normalized data. The gold datasets consist of sentences from which all OOV tokens were manually normalized. An accuracy of 71.8% was achieved on the gold data. This is an 18% increase compared to the normalized data (60.8%) which tells us there is still room for improvement for the normalization model.

[2] https://universaldependencies.org/u/pos/INTJ.html

5.2.2 SMS messages

The SMS corpus is slightly larger than the chats corpus. For that reason, I did not expect a big difference in scores between the chats and SMS corpora. However, the scores did differ quite a lot. This could be explained by the difference in words in need of normalization. In the chats corpus, 49.6% of all the words needed normalization while in the SMS corpus only 26.6% words were in need of normalization. The tagger reached an accuracy of 59.8% on the non-normalized test set. On the normalized test set the tagger achieved an accuracy score of 75.5%. Thus, once again, the performance of the tagger improved by more than 25%. Below the classification report of the normalized SMS messages is shown.

Table 13: Classification report of tagged normalized SMS messages

POS Tag	Precision	Recall	F1	Support
ADJ	0.60	0.77	0.67	48
ADP	0.81	0.28	0.41	47
ADV	0.83	0.75	0.78	253
CONJ	0.49	0.69	0.57	26
DET	0.92	0.58	0.71	19
INTJ	0.00	0.00	0.00	52
NOUN	0.47	0.78	0.58	143
NUM	0.75	1.00	0.86	9
PRON	0.88	0.89	0.89	180
PROPN	0.00	0.00	0.00	14
PUNCT	1.00	1.00	1.00	216
SYM	0.00	0.00	0.00	27
VERB	0.86	0.86	0.86	265
X	0.00	0.00	0.00	16

The classification report for the SMS messages was quite similar to the classification report for the chats. For example, the tagger achieved a high recall but fairly low precision on the nouns. Further, the tagger achieved both high precision and recall on the verbs, adverbs, pronouns and punctuation. Also, all instances of proper nouns, interjections, symbols and others were predicted incorrectly. Determiners were tagged with a very high precision (0.92), however, the recall was fairly low (0.58). As with the adpositions in the chats, the adpositions in the SMS messages achieved a high precision (0.81) but very low recall (0.28).

Further, the tagger was also tested on the gold normalization data. This data was tagged with an accuracy of 81.7%. In comparison to the normalized data (75.5%), the accuracy improved by 8.2%. This improvement, compared to the chats domain, is relatively small which tells us that the normalization model performed significantly better on the SMS domain.

5.2.3 Tweets

As shown in the chapter 3, this corpus consists of more than twice as many words as the chats and SMS corpora. Further, this corpus also holds the lowest percentage of words in need of normalization (21.5%). For those reasons, I expected highest the accuracy scores for the tweets out of the three domains. This prediction was somewhat correct, since an accuracy of 64.3% was reached on the non-normalized test but only an accuracy of 72.5% was achieved on the normalized test set. It makes sense that the difference in scores between the non-normalized normalized test sets is smaller for the tweets since only 21.5% of the words in the corpus were in need of normalization while those numbers were higher for the other two domains. The classification report for the normalized tweets is shown below.

Table 14: Classification report of tagged normalized chats

POS Tag	Precision	Recall	F1	Support
ADJ	0.62	0.71	0.66	201
ADP	0.91	0.19	0.31	341
ADV	0.51	0.76	0.61	388
CONJ	0.77	0.82	0.79	184
DET	0.99	0.74	0.85	192
INTJ	0.00	0.00	0.00	34
NOUN	0.59	0.86	0.70	599
NUM	0.78	0.85	0.81	53
PRON	0.81	0.91	0.86	454
PROPN	0.00	0.00	0.00	140
PUNCT	0.98	0.98	0.98	362
SYM	0.00	0.00	0.00	27
VERB	0.86	0.79	0.82	541
X	0.03	0.02	0.03	42

Some trends regarding the precision and recall scores per class that were found in the chats and SMS messages were also found in the tweets. Verbs, pronouns and punctuation achieved high scores for both precision and recall. Determiners reached a nearly perfect precision (0.99) but slightly lower recall (0.74). Adpositions were tagged with high precision (0.91) but very poor recall (0.19). Further, the classes on whose the tagger performed extremely poorly once again included interjections, proper nouns, symbols and others.

Finally, the tagger was also tested on the gold normalization data from the twitter corpus. The tagger reached an accuracy of 76.1, which is an increase of 5% compared to the normalized data (72.5). This is the smallest improvement out of the three domains. The data normalized by the model was POS tagged almost as well as the data that was normalized manually.

5.3 DISCUSSION

In this section the results will be discussed briefly.

The first part of the experiment consisted of the normalization task. The evaluation on this task showed us that MoNoise reached scores for the normalization of the collected data similar to scores that were reported in earlier work on MoNoise. This shows us that MoNoise is capable of handling data that contain a lot of non-standard language. The second part of the experiment consisted of the POS tagging of the non-normalized data and the normalized data for all domains. The data was tagged using the spaCy tagger which uses the Universal Dependencies POS tagset. For all three domains, we saw that the tagging scores improved after the data was normalized using MoNoise. The chats domain yielded the biggest improvement, the accuracy increased by **44.4%**. A slightly smaller improvement was found when evaluating the tagger on the SMS messages, the accuracy increased by **26.3%**. The twitter corpus yielded the smallest improvement out of the three domains, the accuracy increased by **12.8%**. By generating classification reports some insight was gained on what tags were being predicted correctly and what tags were being predicted incorrectly after the data was normalized. Some trends were found within the three domains. In general, the tagger performed well on the verbs, determiners, pronouns and punctuation. In contrary, the tagger performed poorly on the proper nouns, symbols, interjections and others. Further, I found that adpositions were tagged with high precision but low recall while the nouns were tagged with high recall but fairly poor accuracy. Furthermore, a comparison was drawn between how the tagger performed on the normalized data and how it performed on the gold normalization data. The difference in scores turned out smaller than I expected, especially for the SMS messages and the tweets. The spaCy tagger should be able to tag Dutch data with an accuracy around the 90%. The scores we have seen in this thesis are significantly lower than this benchmark. However, the POS tagger did not achieve much better on the manually annotated gold normalization data. This means that the difference in performance between the spaCy benchmark for Dutch data (90%) and the normalized data is not due to poor performance from the normalization system but due to the collected data which makes sense since only sentences were collected which contained at least four OOV tokens.

6 CONCLUSION

In this chapter the answer on whether we can improve POS tagging by normalizing Dutch user generated content will be given.

A state-of-the-art normalization system named MoNoise was used to normalize user generated content collected from three different domains: chats, SMS messages and twitter. Only messages with a high volume of OOV words were collected from these three domains. Subsequently, the collected test data was POS tagged before- and after normalization. The difference between these two tagging scores can be seen as the impact of normalization on POS tagging user generated content.

All in all, I can conclude that by normalizing Dutch user generated content it is definitely possible to improve the POS tagging of Dutch user generated content, especially when the data consists of a lot of non-standard language (OOV tokens).

Despite the normalization, the tagger still had a lot of trouble tagging proper nouns and interjections correctly. Although words like *psss*, *ouch* and *hello* may be hard to define, they appear often in user generated content. Improving the tagging accuracy of these POS tags might be an interesting subject for future work.

A ANNOTATION GUIDELINES

In dit bestand zal ik wat guidelines specificeren voor het normaliseren van Nederlandse Tweets, Chats & SMS berichten. OOV woorden kunnen op de volgende manieren worden genormaliseerd:

- One-to-One normalisatie: 1 woord wordt vervangen door 1 woord
- One-to-Many normalisatie: 1 woord wordt vervangen door N woorden
- Many-to-One normalisatie: N woorden worden vervangen door 1 woord

Nu zal ik wat richtlijnen opstellen die aangehouden moeten worden bij het annoteren van de data.

- Hoofdletters worden meegenomen.
- Niet Nederlandse woorden worden volledig genegeerd (behalve vlaams), zelfs woorden die ook vaak in Nederlandse tweets voorkomen (bijv. "Thanks").
- Woorden met een "#" ervoor worden genegeerd.
- Woorden uit een Nederlands dialect worden naar de correcte standaard Nederlandse vorm geannoteerd tenzij de betekenis van het woord onbekend is.
- Woorden waarbij de nadruk op het desbetreffende woord wordt gelegd d.m.v. het gebruik van een trema worden naar de normale Nederlandse vorm gezet (bijv. héél > heel)
- Vormen van begroeting die niet behoren tot de standaard Nederlandse taal worden genormaliseerd naar "Hoi" (bijv. Hey/Haai/heej).
- Getallen met "de" of "ste" erachter worden naar bijv. "tweede" of "achtste" genormaliseerd.
- Afkortingen worden genormaliseerd naar de volledige betekenissen (bijv. "ipv" > "in plaats van").

- Het teken “=” wordt genormaliseerd naar “is”.
- Woorden als “zoooooo” en “Gooeddd” worden naar “Zo” en “Goed” genormaliseerd.
- Twee woorden waar een “-” tussen zit worden genormaliseerd naar twee losse woorden.
- Verkleinwoorden worden in sommige gevallen genormaliseerd naar de normale vorm. Dit hangt ervan af of het desbetreffende woord in het Nederlands vaak in verkleinde vorm wordt gebruikt (bijv. “kusje” wordt niet genormaliseerd naar “kus”, maar “buschauffeurtje” wordt wel genormaliseerd naar “buschauffeur”).
- Het Vlaamse woord “ne” wordt altijd genormaliseerd naar “een”.
- Woorden die direct achter elkaar worden herhaald worden terug gezet naar de eenmalige vorm (bijv. “snelsnel” > “snel”).
- Woorden als “ok”, “okay”, “Oowke” worden genormaliseerd naar “Oké”.

Als het niet duidelijk is wat het OOV woord betekent en/of wat de genormaliseerde vorm zou moeten zijn, negeer dan het woord en verander niks!

BIBLIOGRAPHY

Baldwin, T., M.-C. de Marneffe, B. Han, Y.-B. Kim, A. Ritter, and W. Xu (2015a). Guidelines for english lexical normalisation. https://github.com/noisy-text/noisy-text.github.io/blob/master/2015/files/annotation_guideline_v1.1.pdf.

Baldwin, T., M.-C. de Marneffe, B. Han, Y.-B. Kim, A. Ritter, and W. Xu (2015b). Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pp. 126–135.

Bouma, G. and G. Van Noord (2017). Increasing return on annotation investment: the automatic construction of a universal dependency treebank for dutch. In *Proceedings of the nodalida 2017 workshop on universal dependencies (udw 2017)*, pp. 19–26.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Choudhury, M., R. Saraf, V. Jain, A. Mukherjee, S. Sarkar, and A. Basu (2007). Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR) 10*(3-4), 157–174.

De Clercq, O., B. Desmet, and V. Hoste (2014). Guidelines for normalizing dutch and english user generated content. Technical report, Technical report, Ghent University.

De Clercq, O., S. Schulz, B. Desmet, and V. Hoste (2014). Towards shared datasets for normalization research. In *Language Resources and Evaluation Conference*, pp. 1218–1223. European Language Resources Association (ELRA).

Derczynski, L., A. Ritter, S. Clark, and K. Bontcheva (2013). Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pp. 198–206.

Eisenstein, J. (2013). What to do about bad language on the internet. In *Proceedings of the 2013 conference of the North American Chapter of the association for computational linguistics: Human language technologies*, pp. 359–369.

Erik and A. van den Bosch (2013). Dealing with big data: The case of twitter. *Computational Linguistics in the Netherlands Journal 3*, 121–134.

Gimpel, K., N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith (2010). Part-of-speech tagging for twitter: Annotation, features, and experiments. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.

Han, B. and T. Baldwin (2011). Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 368–378. Association for Computational Linguistics.

Hupkes, D. and R. Bod (2016). Pos-tagging of historical dutch. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 77–82.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Volume 10, pp. 707–710.

Li, C. and Y. Liu (2015). Joint pos tagging and text normalization for informal text. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Maarten van Gompel, Ko van der Sloot, I. H. A. v. d. B. (2018). Ucto: Unicode tokeniser. reference guide, language and speech technology technical report series 18-01. *Radboud University, Nijmegen*.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Oostdijk, N., M. Reynaert, V. Hoste, and H. van den Heuvel (2013). Sonar user documentation. *Online:< https://ticclops. uvt. nl/SoNaR_end-user_documentation_v 1*(4).

Plank, B. (2016). What to do about non-standard (or non-canonical) language in nlp. *arXiv preprint arXiv:1608.07836*.

Porta, J. and J.-L. Sancho (2013). Word normalization in twitter using finite-state transducers. *Tweet-Norm@ SEPLN 1086*, 49–53.

Toutanova, K., D. Klein, C. D. Manning, and Y. Singer (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pp. 173–180. Association for computational Linguistics.

van der Goot, R. and G. van Noord (2017). Monoise: Modeling noise using a modular normalization system. *arXiv preprint arXiv:1710.03476*.

van der Goot, R. M. (2019). *Normalization and Parsing Algorithms for Uncertain Input*. Ph. D. thesis, University of Groningen.

Yang, Y. and J. Eisenstein (2013). A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 61–72.