Automatic Detection and Correction of Preposition Errors in Dutch

Master Thesis Information Science
December 2015
Student: L.A.M. Kloppenburg - S2223902
Primary supervisor: Dr. M. Nissim
Secondary supervisor: Prof. Dr. J. Bos
Foreword

This work is produced as thesis for the MA degree in Information Science (Communication- and Information Sciences) at the University of Groningen. I would like to sincerely thank my supervisor Malvina for her dedication and for guiding me and teaching me along the way in the friendliest and most patient fashion. Furthermore, I would like to thank everyone who helped me annotate sentences on Voorzetselaar.nl.

I hereby claim that this work is my own.
Abstract

Automatic correction of errors in language is important because of the growing number of language learners worldwide, and the growing importance of multilingualism in a globalising world. Preposition use is problematic for learners of a new language because it is often impossible to tell what preposition should be in which context simply based on the context around it. I examined an almost unused learner corpus for Dutch, the Leerdercorpus Nederlands. My approach can be seen as language modelling through leveraging the information found in native Dutch text. I built a decision model based on a detection and a selection model. The detection model detects the presence or absence given a feature vector that denotes the context of a Dutch sentence around a certain word. The selection model selects a suitable preposition out of the fifteen most frequent prepositions in Dutch for a given feature vector. Together, these models are built to detect and correct all three identified error types for prepositions: deletion, insertion and substitution errors. The detection model is built on 2,000,000 items and the selection model on 20,000,000. On L1 native test data, the detection model achieved an F-score of 100%, and the selection model achieved 75%. On L2 learner data, based on an annotated data set of 1,499 items, 104 true errors were found through crowdsourcing. Out of all 72 true substitution errors, around 54% were corrected appropriately. There were only 4 deletion errors, of which 3 were detected. Of the 29 insertion errors, 20% were detected.
# Contents

1 Introduction ........................................ 1
   1.1 Dutch prepositions .............................. 2
   1.2 Error types .................................... 6
   1.3 Research question .............................. 8

2 Relevant work ...................................... 9
   2.1 Rule-based error correction .................... 9
   2.2 Supervised error correction .................... 10
   2.3 Frame-based error correction .................. 13
   2.4 Error detection ................................ 15
   2.5 Task difficulties .............................. 16
   2.6 Overview ...................................... 17

3 Method ............................................ 19
   3.1 Overall approach ............................... 19
   3.2 Resources .................................... 21
      3.2.1 Alpino .................................... 21
      3.2.2 LASSY Large .............................. 21
      3.2.3 Leerdercorpus Nederlands .................. 22
   3.3 Approach ...................................... 23
      3.3.1 Decision model ............................ 23
      3.3.2 Detection model ............................ 28
      3.3.3 Selection model ............................ 31
   3.4 Evaluation .................................... 34
      3.4.1 Baselines ................................ 35
      3.4.2 Native data ............................... 36
3.4.3 Learners' data ................................................. 36

4 Results ......................................................... 41
  4.1 Native test data .............................................. 41
  4.2 Learner test data ............................................ 43
    4.2.1 Voorzetselaar.nl ......................................... 43
    4.2.2 Classifier performance ................................... 43
      Detection .................................................... 44
      Selection ..................................................... 45
    4.2.3 Evaluation learner corpus ............................... 46
      Error density ............................................... 46
      Common confusion ......................................... 47

5 Discussion .................................................. 49
  5.1 Results on native data ..................................... 49
  5.2 Decision model ............................................ 50
    5.2.1 Features .................................................. 52
    5.2.2 Robustness ............................................... 53
    5.2.3 State of the art ......................................... 54
  5.3 Leerdercorpus Nederlands as learner corpus ............... 55
  5.4 Error feedback ............................................. 56

6 Conclusions .................................................. 59
  6.1 Summary .................................................... 59
  6.2 Future work ................................................ 60
  6.3 Final statement ............................................. 62
Chapter 1

Introduction

Why is the automatic detection and correction of preposition errors so important? How ought this problem be divided into different subtasks and what does the phenomenon of preposition error look like in actual language use? In this chapter, I aim to answer these questions and establish a solid understanding for the reader.

Computer-assisted language learning is a field where language technology is combined with human language acquisition. Almost everywhere in the world, people have to use a foreign language for school, research, work or the globalisation of media. In many cases, the language in question is the major language within the scope where the language is used. Learning a new language is difficult because of the interlingual or intercultural differences that become apparent during the learning process, such as grammatical differences, lexical differences and expressions that cannot be translated correctly. Grammar is particularly hard to get acquainted with, since it always involves a complex set of rules with a lot of exceptions which require a lot of practice. In many languages, prepositional constructions are the most difficult for foreign language learners [Dale et al. 2012], because of their versatile and often ambiguous character. Therefore, it makes sense to keep investigating new methods to tackle this issue in order to smooth the process for language learners, resulting in a more multilingual society and, hopefully, better communication.

Because prepositions are used so extremely frequently in language and relevant data can be collected easily, it is interesting to investigate the possibilities of what
can be done with language technology to improve the learner’s experience. This has been done intensively in the past, with numerous techniques and even more diverse results. Approaches to provide language learners with feedback have also been examined thoroughly [Dale et al., 2012].

To my knowledge, there is no available work on this particular task for Dutch language acquisition. Because there has been a substantial amount of publications for English, it stands to reason that other languages should be explored as well, and this thesis will focus on Dutch preposition error detection and correction. Because of the lack of previous work for this language, the aim of this thesis is to provide a first step into Dutch preposition error correction. I will explore different machine learning approaches and the influence of the size of the training data by considering this error correction problem as a classification problem in order to leverage Dutch native data to build informed models on preposition use. As a byproduct, I will investigate the quality of a yet unused Dutch learner corpus, the Leerdercorpus Nederlands [Perrez and Degand, 2009], for this task.

1.1 Dutch prepositions

This section aims to provide quantitative insights into the usage of the Dutch preposition. Illustrating what roles prepositions perform and in what contexts they occur may grant us a basic impression of why they are problematic for second language learners. To analyse the Dutch preposition, I used Dact. Dact (Decaffeinated Alpino Corpus Tool) is a graphical user interface which enables users to perform XPath queries on corpora annotated by Alpino [Bouma et al., 2000]. Simple frequency statistics can be outlined to gain an impression of the corpus or corpora that have been loaded. It has proven to be very effective when shaping a general description and understanding of the Dutch preposition.

Prepositions are a rather small lexical class. There are around one hundred prepositions in Dutch, but they occur so frequently that they amount to a considerable frequency. The CDB corpus[1] was used for the following experiments. It contains 125,212 tokens (18,790 preposition tokens) and 22,075 types (106 prepos-

[1]The newspaper part of the Eindhoven Corpus
ition types). This means that while preposition types contribute to only 0.66% of all types, they contribute to 15% of all tokens (based on this corpus). This emphasises that prepositions are essential building blocks in the production of correct language.

Even within the array of preposition types, there is a considerable degree of frequential variety. Table 1.1 illustrates this well. It shows that the top 4 prepositions comprise over 50% of all preposition tokens, while representing less than 4% of all preposition types. Furthermore, the top 17 prepositions (16% of all types) contribute to well over 90% of all preposition tokens. For my thesis, I focus on the top 15 prepositions in Dutch (up to als), as this results in about 90% of all preposition use in Dutch. Adding more prepositions would increase the difficulty of the task because of the low frequency they represent. Little information could be collected for these prepositions and this would impair classifier performance.

To gain a further understanding of the difficulty that prepositions may pose to (foreign) language learners, it makes sense to outline their various roles. Using the same corpus for experimenting, the most common relations of prepositional phrases can be outlined in Table 1.2. The relations have been assigned manually in correspondence with the Syntactic Annotation Manual for Lassy [van Noord et al., 2011]. Examples of these relations (in Dutch) can be found in sentences (1–6).

These are the most common roles that PP’s perform in Dutch. Consider that in less than 12% of these observed PP’s, the desired preposition is fixed and thus fairly unambiguous. A ‘fixed’ preposition is generally anchored to a certain verb or expression and can thus not easily be mistaken for a different one. But this means that in 88% of all PP’s, there is the possibility that, theoretically, there are multiple acceptable prepositions and that the actual preposition highly depends on the entire context and not just on the verb. This makes it difficult for learners to study and master the use of prepositions because there is often not a clearly defined rule of which preposition is desirable [Felice and Pulman, 2009].


(2) Veelal klopt dit ook [met de werkelijkheid] (pc).

(3) Jan Tinbergen is er heilig van overtuigd, dat oost en west sterk [naar]
Table 1.1: Frequency distribution of Dutch prepositions (N=18,790).

<table>
<thead>
<tr>
<th>Preposition</th>
<th>Translation</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>van</td>
<td>of</td>
<td>22.8%</td>
</tr>
<tr>
<td>in</td>
<td>in</td>
<td>16.3%</td>
</tr>
<tr>
<td>op</td>
<td>on</td>
<td>7.2%</td>
</tr>
<tr>
<td>te</td>
<td>at</td>
<td>7.1%</td>
</tr>
<tr>
<td>voor</td>
<td>for/ before</td>
<td>6.3%</td>
</tr>
<tr>
<td>met</td>
<td>with</td>
<td>5.7%</td>
</tr>
<tr>
<td>aan</td>
<td>on, to</td>
<td>4.3%</td>
</tr>
<tr>
<td>door</td>
<td>through, by</td>
<td>3.3%</td>
</tr>
<tr>
<td>bij</td>
<td>at, with</td>
<td>2.8%</td>
</tr>
<tr>
<td>uit</td>
<td>out</td>
<td>2.6%</td>
</tr>
<tr>
<td>om</td>
<td>by, around</td>
<td>2.6%</td>
</tr>
<tr>
<td>over</td>
<td>over, about</td>
<td>2.4%</td>
</tr>
<tr>
<td>tot</td>
<td>until, to</td>
<td>2.2%</td>
</tr>
<tr>
<td>naar</td>
<td>to</td>
<td>2.1%</td>
</tr>
<tr>
<td>als</td>
<td>if, as</td>
<td>1.2%</td>
</tr>
<tr>
<td>na</td>
<td>after</td>
<td>1.2%</td>
</tr>
<tr>
<td>tegen</td>
<td>against</td>
<td>1.1%</td>
</tr>
<tr>
<td>others</td>
<td></td>
<td>8.8%</td>
</tr>
</tbody>
</table>

Table 1.2: Roles for prepositional phrases (N=14,930)

<table>
<thead>
<tr>
<th>Relation</th>
<th>Description</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod</td>
<td>The PP modifies an NP</td>
<td>76%</td>
</tr>
<tr>
<td>pc</td>
<td>The PP has a fixed preposition which cannot be replaced</td>
<td>11.7%</td>
</tr>
<tr>
<td>ld</td>
<td>The PP acts as a label for location/destination</td>
<td>6.5%</td>
</tr>
<tr>
<td>cnj</td>
<td>The PP is part of a conjunction</td>
<td>2.3%</td>
</tr>
<tr>
<td>obj2</td>
<td>The PP acts as an indirect object (typically with aan)</td>
<td>0.9%</td>
</tr>
<tr>
<td>predc</td>
<td>The PP is a complement for a nominal predicate</td>
<td>0.8%</td>
</tr>
</tbody>
</table>
elkaar | toegroeien (ld).

(4) |Bij de burgemeester | en | bij de leiding van de politie | blijft de grote zorg het tekort aan personeel (cnj).

(5) Het werk is | aan hem | opgedragen (obj2).

(6) Het hebben van een opvolger is | van groot belang | (predc).

Furthermore, prepositions do not solely occur in PP’s, although the majority of them do (81.1%).

Table 1.3 outlines the other categories in which prepositions can occur, according to the same corpus. The sentences (7–11) exemplify these roles.

Table 1.3: Categories for preposition situations

<table>
<thead>
<tr>
<th>Context</th>
<th>Description</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>pp</td>
<td>P occurs in PP (Table 1.2)</td>
<td>81.1%</td>
</tr>
<tr>
<td>(o)ti</td>
<td>P occurs in non-finite clause</td>
<td>8.6%</td>
</tr>
<tr>
<td>mwu</td>
<td>P is part of multi-word-unit</td>
<td>5.0%</td>
</tr>
<tr>
<td>smain</td>
<td>P is part of finite clause</td>
<td>2.0%</td>
</tr>
<tr>
<td>cp</td>
<td>P is part of clause</td>
<td>1.6%</td>
</tr>
<tr>
<td>inf</td>
<td>P is product of verb inflexion</td>
<td>1.1%</td>
</tr>
<tr>
<td>other</td>
<td></td>
<td>0.6%</td>
</tr>
</tbody>
</table>

(7) Dit effect is nu bezig | te verdwijnen | ((o)ti).

(8) Dit | in verband met | de gemiddeld langere levensduur van de vrouw (mwu).

(9) |Hij liep een shock op | en kneusde enkele ribben (smain).

(10) Wij hebben niet gewacht | tot ze ons huis in brand zouden steken | (cp).

(11) Nu heeft een medische faculteit in Maastricht niets om | mee samen | te | werken | (inf).

In summary, prepositions perform many different roles and occur in different constituents. There often is no one-on-one translation between two languages. These factors contribute to why it seems to be so difficult for language learners.
The next section focuses on the different possible preposition errors in learner language.

1.2 Error types

The problem or task of correcting preposition errors in texts written by language learners should be divided into two subtasks. The first subtask is error detection, the task of teaching a system how errors can be detected. The second subtask is error correction, the task of providing the language learner with appropriate feedback or correction. Whereas error detection finds its roots within language technology, error correction can also be considered as a task within the field of language acquisition.

The totality of possible preposition errors may be divided into two separate paradigms: word- or phrase form errors and erroneous use of prepositional constructions [Eeg-Olofsson and Knutsson, 2003], where the former concerns how prepositions may be misspelled- or formed, and the latter how they might be misused. The three basic situations of preposition misuse are identified as follows by the three discussed approaches [Chodorow et al., 2007, Eeg-Olofsson and Knutsson, 2003, Liu, 2008]:

1. Insertion (where a preposition was invoked erroneously)
2. Deletion (where a preposition was omitted erroneously)
3. Substitution (where a preposition was picked erroneously)

This distribution effectively separates distinct types of errors which all require different strategies to solve. Note that the deletion and insertion errors are mirrored. After all, they both require the comparison of preposition absence on the one hand, and preposition presence on the other hand. The mirrored relationship is illustrated in the table below.

An example of an insertion error is found in (12). The bold-faced van is the erroneously inserted preposition. (13) shows how a learner whose mother
Table 1.4: Mirroring of deletion and insertion error types.

<table>
<thead>
<tr>
<th></th>
<th>Correct use</th>
<th>Learner sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deletion</td>
<td>prep $P$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Insertion</td>
<td>$\emptyset$</td>
<td>prep $P$</td>
</tr>
</tbody>
</table>

tongue is English might come to this construction because the intended preposition construction is correct in English. (14) finally shows the corrected sentence.

(12) *Ik sprong uit van het raam.
(13) I leapt out of the window.
(14) Ik sprong uit het raam.

An example of a deletion error can be found in (15), with an English approximation in (16). As exemplified in (17), in Swedish, prepositions in front of dates or years are prohibited, a possible explanation for why a Swedish learner of Dutch might make this mistake. A deletion error requires that a fitting preposition is inserted at the omitted position, as exemplified in (18), where the fitting preposition is bold-faced.

(15) *Roger Federer werd $\emptyset$ 1981 geboren.
(16) *Roger Federer was born $\emptyset$ 1981.
(17) Roger Federer föddes ($\emptyset$) 1981.
(18) Roger Federer werd in 1981 geboren.

Finally, consider that a substitution error is relatively straightforward. In both the learner sample and the native corpus, one needs to look at preposition cases. An example of a substitution error can be found in (19), where the bold-faced preposition, though arguably not excessively wrong, is unsuitable in that context. The preposition *through* in (20) can often be translated to both *via* and *door* in Dutch which explains why a learner could well make such a mistake. The desired correction is to replace the wrong preposition with a/the correct one:

$$\text{Prep } p \leftarrow \text{Prep not } p.$$  

This correction is exemplified in (21).
1.3 Research question

In summary, there is a clearly defined problem in the field of language acquisition, namely the use of prepositions by learners in a foreign language. My research can be motivated by the research question ‘Is it possible to leverage native Dutch language data to model the use of Dutch prepositions and apply this to the detection and correction of preposition errors in Dutch learning data?’ Because of the prospective use of a learner corpus for Dutch which is not yet applied to this task, this can be divided into three smaller questions:

1. Is it possible to leverage native Dutch language data to accurately create a model of the use of Dutch prepositions?

2. Is it possible to use this model to detect and correct Dutch preposition errors in Dutch learning data?

3. Is the Leerdercorpus Nederlandsia suitable corpus for this task?

In the next chapter, relevant work on preposition error detection and correction will be discussed to summarise the work that has been done in this field so far and to critically analyse the adopted techniques and motives. Additionally, it will describe the issues that arise when performing this task.

---

3 Described in Section 3.2
Chapter 2

Relevant work

Relevant work into preposition error detection and correction is becoming increasingly extensive. In this section, I aim to outline three distinct approaches to solving this problem, as well as include more detailed work on the issues that arise when exploiting learner data. It is part of a larger literature review which was written as a theoretical background for this thesis.

The first method is a rule-based error correction method by Eeg-Olofsson and Knutsson [2003], the second method is a supervised classification method by Chodorow et al. [2007], and the last one is a frame-based error correction method by Liu [2008]. Another supervised method is done by Felice and Pulman [2009], who provide rich explanation into the difficulties of this particular task.

2.1 Rule-based error correction

Eeg-Olofsson and Knutsson 2003 state that the preposition deletion error type was the most complicated error type to be detected, as it requires a rule-based system to detect an error based on the absence of a certain item, rather than its presence. An important decision by Eeg-Olofsson and Knutsson 2003 was to devise or describe the rules in such a way that would sharpen precision but cripple recall, as the so-called phenomenon of false alarms, when learners are falsely accused of errors, is very negative towards the language learner when given
feedback. The assumption is that it is better to give accurate feedback but detect less errors, than to detect more errors, but also falsely mark correct constructions.

In a test set of 2,800 words, there were 40 manually found prepositional errors. The recall was reported at 25%, 11 errors were detected by the system\[1\]. The precision was assessed rather poorly, since any detection was determined to be a success of the program, disregarding correction performance.

### 2.2 Supervised error correction

A statistical and supervised approach was adopted by Chodorow et al. [2007], using a maximum entropy classifier to statistically detect preposition errors in student essays. The first error type, **insertion**, was tackled by statistically determining the probability of any preposition in a certain situation based on the lexical features present in the local context. The system was trained on a 7 million event training set. These events are occurrences of prepositions and their contexts and are then structured into 25 contextual feature-value pairs. Features include nearby words or N-grams to the right of left of the preposition, headwords of the phrase, preceding noun phrases, prepositions, verbs and verb phrases, and preceding lemmas. Note that some feature-value pairs only take a few possible values, such as preceding phrase, which could be either a **noun phrase** or a **verb phrase**. Others, such as preceding words or N-grams, may contain thousands of different values.

Evaluation on grammatical text (L1 data) was done probabilistically – For 18,157 preposition contexts, the system computed the probability of every preposition based on the feature-value pairs as the knowledge set. For L1 data, the overall agreement between the classifier and the text was reported at 0.69, whereas the kappa-value was 0.64. Disagreement was often caused by two prepositions that shared almost equal probability. Because of this, the most probable preposition was chosen with rather little confidence, since the second best candidate was nearly equally suitable, resulting in a relatively high error probability. The test was rerun, this time increasing the confidence interval to 0.6. The difference between the best two choices had to be big enough, or the case would be skipped. 50% of the cases were skipped this way, but a much higher agreement (0.90 and kappa 0.88)

\[1\]This seems to be slightly over 25% : 27.50%
was found.

The real test was run on 2,000 preposition contexts drawn from data created by language learners (L2 data) which were annotated by two human raters (resulting in two annotated gold standards). Chodorow et al. [2007] state that there were many spelling errors. Rather than using a spelling corrector as was done by Nagata et al. [2014], these sentences were skipped. The same was done in case of punctuation errors, antonymous prepositions (with/without are interusable, so one is not necessarily more correct than the other) and benefactives. The remaining number of contexts is not reported, so the severity of these problematic factors cannot be interpreted. If the amount of problematic cases is substantial, then the decision to skip these makes little sense. It would be better to learn how to cope with these cases in the future.

The decision to skip problematic cases means that precision (agreement) was favoured at the cost of recall. The relevance of either measurement depends on the perspective of the research, which I outline in the Discussion and Conclusion chapters. Chodorow et al. [2007], Nagata et al. [2014] and Eeg-Olofsson and Knutsson [2003] all share the motivation that preposition correction is essential within the field of second language learning, because prepositions tend to be a problematic part of language acquisition. While developing tools designed to aid language learners to tackle this issue, one can argue that a high precision is more important for the learner. After all, the proper correction of prepositions is essential in the sense that false corrections tend to confuse the learner. The results on L2 data are displayed in Table 2.1.

<table>
<thead>
<tr>
<th></th>
<th>Rater 1 vs. Rater 2</th>
<th>Classifier vs. Rater 1</th>
<th>Classifier vs. Rater 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement</td>
<td>0.926</td>
<td>0.942</td>
<td>0.934</td>
</tr>
<tr>
<td>Kappa</td>
<td>0.599</td>
<td>0.365</td>
<td>0.291</td>
</tr>
<tr>
<td>Precision</td>
<td>N/A</td>
<td>0.778</td>
<td>0.677</td>
</tr>
<tr>
<td>Recall</td>
<td>N/A</td>
<td>0.259</td>
<td>0.205</td>
</tr>
</tbody>
</table>

The results show that there is a high agreement between the classifier and Kappa was lower (between 29% and 36%) because of the high level of agreement
expected by chance. Subsequently, precision was between 77.8% (Rater 1) and 67.7% (Rater 2). Recall was lower, between 25.9% and 20.5% for respectively Rater 1 and 2.

Another supervised error correction method was developed by [Felice and Pulman 2009] in service of their Dapper (Determiner And PrePosition Error Recogniser) project, with a 42% precision and a 35% recall on the task. [Felice and Pulman 2009] particularly aim to discuss why using L2 data is problematic for this task, and they explain why prepositions are difficult to use. Language learners of English frequently err when using prepositions, based on the results of an error set that outlined that 12% of all errors were preposition errors. One of the main reasons of why preposition use is problematic is that they express no clearly definable pattern in which they are used. [Felice and Pulman 2009] elaborate on this by stating that sentences of similar structure still express different prepositions, and that even very similar meanings are often accompanied by different prepositions. This unpredictable behaviour makes it particularly difficult for learners to master prepositions, as it also occludes native speakers from explaining which preposition is used in what context.

In their classifier, [Felice and Pulman 2009] made use of features of a deeper level than only POS level or N-gram level features than the other discussed papers in their work, in order to convey deeper information on preposition contexts. Based on the presumption that preposition use cannot be profiled by sheer shallow structure such as surrounding POS patterns or the origins of surrounding words, the decision to include multiple levels of sentence information is well-grounded. The preposition component contains models for nine prepositions, being at, by, for, from, in, of, on, to and with. These are the most frequent prepositions in English according to [Felice and Pulman 2009]. On a native data set of 530,000 preposition contexts structured into vectors, accuracy was reported at 70%. They added an interesting feature to their work by including an upper bound. They asked two native speakers of English to manually fill out 841 contexts. Their accuracy, measured by comparing their selections to the original prepositions, averaged 88%. The 12% that represents disagreement is not elaborated. Perhaps the prepositions simply differed but were nonetheless acceptable or perhaps the lack of context made it difficult for them to identify the desired preposition. It nonetheless emphasises that preposition use is difficult. It also raises questions
about the standard for a preposition error correction model. Should it simulate
the native speaker or the original text? Can we, at some point, expect it to outper-
form a native speaker simply because the native speakers did not achieve perfect
agreement themselves?

Evaluating on L2 data was done in two parts: on correct L2 data (5,753 in-
stances) and on incorrect L2 data (1,116 instances). Accuracy on incorrect L2 data
implies the correct detection of a preposition error and the subsequent selection
of a suitable correction. Evaluating on correct L2 data is similar to evaluating on
L1 data, and the results were roughly the same at 69% accuracy (vs. 70% on L1
data). The incorrect L2 data consisted of only erroneously used prepositions. Over
76% of these were marked by the system, indicating that it is fairly accurate when
identifying incorrect prepositions in a binary fashion. Of these 76%, 51.70% were
corrected appropriately. That is, those received a suitable correction. This means
that of 1,116 errors, approximately 441 (39.5%) received suitable corrections.

2.3 Frame-based error correction

Perhaps an elegant balance between a manual rule-based system and a supervised
probability system is the work of Nagata et al. [2014] who introduced the use of
Error case frames to solve the task of preposition error detection and correction.

Rather than collecting features that represent correct language, the system is
taught basic incorrect situations. A case frame may be constructed by parsing a
given sentence with a dependency parser. A case frame is a kind of data structure
consisting of three parts:

1. The root of the head verb

2. The basic case, consisting of the subject, the object and all the prepositions.

3. The feedback message (only in error case frames)

It is used to describe a certain phenomenon in a data set in a machine-readable
way. It is assumed that a corpus based on newspaper articles likely almost only
contains grammatical language. Parsing this (native) corpus and extracting the
relevant features into case frames, structures text into data structures that rep-
resent accepted grammatical situations. Doing the same for a learner corpus is
A learner corpus contains errors, following the assumption. Therefore, the frames that are extracted from this corpus may represent incorrect language. Taking the learner case frames and subtracting all case frames that appear also in the native corpus, would theoretically result in a set of case frames that solely represent incorrect prepositional situations, or error case frames. However, these case frames may also represent frames the native corpus simply lacked because of the sparse and diverse character of sentences. Therefore, to generalise the data, a series of steps are taken.

First of all, all sentences over 20 words or sentences with commas are omitted. This is because these sentences are often difficult for a parser to deal with, resulting in erroneous parses which are not appropriate to generate case frames with. Then, some rules are adopted to make the frame generation less strict. For example, only objects are obligatory in a case frame, and only one preposition is. Furthermore, all prepositions left of the verb are optional. This means that case frames are more general and that, while matching them with error case frames, there is a less strict procedure. Subsequently, similar case frames can be grouped into one. This enhances the performance of the system. Some case frames differ only in possible objects, but not in structure. The process is illustrated in Figure 1. Then, the proposed approach is started. The frames that are now generated in the learner corpus are compared with the ones from the native corpus. If a frame appears only in the learner corpus, it means it may be an error case frame, resulting in a candidate error case frame.

The correction process is rather straightforward. The error case frames are assumed to contain an erroneous preposition. Replacing this preposition with a different one results in a new case frame. Checking if this new frame does appear in the native case frame collection answers the question if the correction is acceptable. If there is no available correction, the case frame cannot be confirmed to be a true error case frame.

Finally, explanatory notes are added to the error case frames. These notes are manually created comments that aim to provide the learner with extensive and constructive feedback.

The system may be tested by feeding it a target text. The target text undergoes the same procedure as the other texts. It is parsed and divided into case frames. The philosophy is that the similarities between the generated case frames of this
target text and the error case frame collection represent the errors made in this text. An advantage of this system is that the target text may serve as new learner data at the same time, by applying the same procedural steps as mentioned before, a process called *active generation*.

Based on a test set of 8,031 tokens with 131 relevant errors, a precision of 0.823 (0.68 with active generation) was reported, and a recall of 0.107 (0.13 with active generation). Furthermore, 20 feedback messages were generated for errors. Three human raters evaluated the effectiveness and correctness of these messages, resulting in an average of 82%, with an (Fleiss’ kappa) agreement of 0.67 between the raters.

### 2.4 Error detection

The results achieved by the three systems vary. Eeg-Olofsson and Knutsson [2003] do not discuss their low recall thoroughly, but one thought that springs to mind is the chosen approach. A system based on manually written matching patterns has a very limited range. Even though the rules might result in a fair coverage, language is utterly recursive and versatile. Numerous constructions can be made, both grammatical and ungrammatical. Manually writing a set of rules results in a somewhat static representation of a certain language and the errors that can be made in the scope of prepositions. This set of rules proved effective at detecting errors in the way that there were no *false positives*, i.e. *false alarms*. Keeping in mind that this system is meant to *aid language learners* rather than reliably correct huge amounts of texts, sacrificing recall to maintain a high precision makes sense.

Chodorow et al. [2007] express justified concern about the precision, stating that error feedback should never include false positives, corresponding to Eeg-Olofsson and Knutsson [2003]. Also, one must keep in mind that many contexts were skipped because the system could supposedly not handle these cases, so precision would have been even lower otherwise. It can be argued that while it is difficult to train a system to be able to cope with incorrect spelling or punctuation, it is still a very common issue in learner texts. To exclude these cases and allow the system to handle only *prettier* sentences is not realistic - it is not a realistic representation of real world learner data. Other solutions should be pursued.
instead. Nagata et al. [2014], for example, use an automatic spelling corrector to correct misspellings. Felice and Pulman [2009] are concerned about their low recall (76%) because it would mean that many errors would go undetected which is also bad for the learning experience because people would be misled about their level. Nagata et al. [2014] also deal with low recall (0.107), which is somewhat improved by implementing active generation (to 0.13), but this cripples precision (0.823 without and 0.68 with), indicating that any prepositional error that was detected using active generation was likely not corrected properly. All papers express the need for high precision in this task because it is more important for learners to receive correct feedback than simply more feedback with a higher error rate. Because of this, the implementation of active generation hurt more than it helped.

Recall seems to be the most difficult aspect of detection. This is probably because of the task’s difficulty to detect insertion and deletion errors. Precision is fairly reasonable, although one must decide whether to lay focus on this measurement or not. When correction quality is the key performance measurement in a system, the philosophy of Eeg-Olofsson and Knutsson [2003] is adoptable: Sacrifice recall to maximise precision. Other suggestions concerning error correction and feedback are presented in the Discussion section.

2.5 Task difficulties

Felice and Pulman [2009] explain why using L2 data is problematic for tools that exploit natural language such as syntactic parsers that are used to acquire preposition context information. L2 data express characteristics that are not likely to be found within native data, on which most NLP tools are trained. L2 data often express simpler sentence structure, which ought to simplify the parsing process for syntactic parsers, but they simultaneously contain a lot of errors which result in parsing errors. Word order errors make it difficult for parsers to assign correct relations between words. Errors in spelling are problematic because it means that different and sometimes non-existant words appear in the parse information of a preposition context. These types are likely not included in the native-based knowledge of the classifier. Additionally, erroneous spelling can result in bad POS-tagging because there is not always a sure way for the parser to find out what type
Felice and Pulman [2009] cite that there were many issues that certainly impaired the results. Content issues arose, such as the fact that Dapper does not handle certain structures (such as temporal expressions: at first, at present) very well which occurred rather often in their learner data because of the nature of the assignments students had to write. They state that students often hold on to fixed expressions which results in certain formulations such as opiniating sentences (From my point of view/on the other hand) to be very frequently found in the L2 data. If the model cannot handle one of these, it is likely that it will make the same mistake multiple times. In summary, many classifier errors occur because L2 writing is, apart form spelling errors, very different in nature from L1 writing. The set-up of Dapper also creates problems as it only outputs one preposition class rather than possibly multiple ones. They emphasise that there are situations in which more than one preposition can be found.

2.6 Overview

There are many possible approaches to solving this problem. Different solutions generally resulted in a fair precision but lower recall, indicating that errors are often left undetected but that errors which are triggered are generally appropriately detected. Because of the infinite character that language describes, it seems infeasible to train on learner data if the available learner corpus is not of substantial size. Training on native data is easier and more convenient because native data is often available in huge quantities. Machine learning approaches are therefore suitable in this respect. The challenge when training on native data is that the training data and the L2 test data are of a very different nature, so building a robust system is hard. Language errors and stilistic differences in L2 data are often stated as issues that arise when training on native data.

The general idea is that there is a lot of room for improvement and that the problem is challenging, for both machine and language learner. What is needed is a strong and robust system that a) minimises false alarms and b) does not miss a lot of errors.

My work was based on the work and approaches by Felice and Pulman [2009] and Chodorow et al. [2007] because my approach is also based on supervised classification. The difference is that I focused on preposition error detection and
correction in Dutch rather than English, something which, to my knowledge, has not been attempted before. The language distance between Dutch and English or Swedish is relatively small, however, and I expect that the discussed challenges will not be substantially different than what they will be for Dutch. Additionally, I attempted to tackle all three identified and discussed error types using this method, not just the substitution error type. Because of the sparse background in deletion and insertion error detection and correction, a new method is needed to tackle these error types as well.
Chapter 3

Method

This chapter describes the developed approach and the used data for the detection and correction of preposition errors in learner Dutch. It explains how I processed the data and used them to build the system that automatically leverages native language in order to detect and correct preposition errors. How I evaluate this system and the other products of the thesis are outlined as well.

3.1 Overall approach

The desired product of this research is a system that, given a sentence preferably written by language learners and substantial knowledge on correct use of Dutch, is able to detect and correct preposition errors of the three previously described error types (deletion, insertion and substitution).

The system itself is taught what good language ought to look like by presenting it with a lot of examples of good language use. By collecting millions of relevant instances of preposition use by native speakers of Dutch, and building a model of these using a machine learning algorithm, I strive to create an accurate model of the fifteen Dutch prepositions described in Section 1.1. This model predicts the most suitable preposition for a given situation, the creation process of which is described in this section. Because I also focus on deletion and insertion error types, I subsequently created a model of Dutch preposition absence and presence. How these models are created, structured and used is explained in Sections 3.3.1, 3.3.2.

In order to detect errors, we need to walk through a sentence and mark those
words or positions which, without prior knowledge, could potentially hold preposition errors. These potential error cases are called *candidates*.

(22) *Ik ga al bijna 4 jaar in school waar ik ontzettend veel leer over bijvoorbeeld natuurkunde en wiskunde, maar ik had 2012 weinig motivatie om ermee door mee te gaan.

(23) *I have been going (in) school for almost 4 years, where I’m learning so much about, for instance, Physics and Match, but (o) 2012 I had little motivation (in) to go through with it.*

Insertion and substitution errors can theoretically always occur whenever a preposition is used, so every preposition is a candidate. For (22), there are five such candidates: *in, over, om, door, and mee.* For deletion errors, the situation is more complicated. After all, deletion errors are not pinned at a misused word, but rather where a word is missing. Again, without prior knowledge, this means that any whitespace could theoretically represent a position that requires a preposition, so for sentence (22) this would be every whitespace (29 in total). However, this makes little sense, because every whitespace position would have to be evaluated and deletion errors should not be expected anywhere. Therefore, a method is needed that only includes whitespaces that could be considered as positions for deletion errors, which I refer to as *absent cases* because they represent preposition absence (conversely, normal preposition cases are *present cases*). The method I implemented looks at the POS-attributes of words around every whitespace in order to determine if it is a worthy candidate for the system to evaluate. For example, a preposition is very unlikely to occur between an article and a noun, but much more so between a verb and an article. The exact details of this method are outlined in Section 3.3.2. For now, let us assume that the system says that the whitespaces between ‘ga al’ and ‘had 2012’ could be deletion errors. Recall that this is hypothetical, the model was not applied here.

When all of the candidates of a sentence have been collected, the next step is to collect linguistic information around these candidates, such as the nouns, verbs and tokens that neighbour the candidate. I use Alpino to syntactically parse the input sentence(s), which yields a lot of linguistic information such as the roles of word groups, the structure between word groups and the roles of separate words.

---

1The errors are put between brackets. The English translation is an approximation.
The exact set of linguistic features I used is outlined and explained in Section 3.3.3 for the preposition model, and in Section 3.3.2 for the binary model. At this stage, the preposition or the position of the whitespace is actually an anchor-point, and information must be extracted within the frame of this anchor-point. The combination of these features and the given preposition or whitespace are called *events*. The structured feature set of an event is a *vector*. As mentioned before, the system will eventually predict if a preposition, and in that case, which preposition, is the most appropriate for a given vector. By comparing the choice provided by the learner with the choice of the system, we can deduce if an error was found, what kind of error that was and what the desired correction ought to be. This evaluation procedure is explained in section 3.4.

### 3.2 Resources

#### 3.2.1 Alpino

Alpino [Bouma et al., 2000] is a state-of-the-art syntactic parser for Dutch, with an accuracy reported at 93%. It is capable of determining the structure of an input sentence of constituent, tagging the words and disambiguating these if necessary. I used Alpino to parse sentences that are run through the system. Parsing these sentences provides the necessary step to pinpoint error candidates and gaining linguistic information around them.

#### 3.2.2 LASSY Large

LASSY Large (Large Scale Syntactic Annotation of Written Dutch) [van Noord et al., 2013] is a syntactically annotated collection of Dutch corpora. It contains various newspaper corpora as well as an extraction of the Dutch Wikipedia. The size makes this corpus exceptionally suitable for large-scale machine-learning, with a total of 700 million words and over 64 million sentences. The corpus is used as a source of native language use. I approach the problem of error detection and correction by trying to teach a system to discriminate between correct and incorrect language, based on the substantial knowledge that the system has on correct (native) language, which it draws from this corpus.
The structure of the data is straightforward. Lassy Large is stored as a collection of sentence archives. Every sentence is expressed as an XML-file, syntactically annotated by Alpino [Bouma et al., 2000]. The XML is structured into nodes, where every node represents a leaf or branch of the complete tree structure. Nodes are enriched with information about that node, i.e. the category (preposition phrases, noun phrases, verb phrases), or the Part of Speech tags (noun, prep, det). For a full overview, refer to the Syntactic Annotation Manual by van Noord et al. [2011].

Node information is expressed as \{attribute:value\}-pairs. These data have been preprocessed, structured and saved, which contributes to the convenience of the corpus for extraction purposes.

No further preprocessing was done on my part, because the analyses were already performed, and sentences of any length and structure were accepted.

3.2.3 Leerdercorpus Nederlands

Leerdercorpus Nederlands (Dutch learner corpus, Perez and Degand [2009]) is an as of yet relatively unused corpus which contains essays and argumentative texts of students who study Dutch as a second language. The distribution of mother languages expresses quite a bit of variation, including French (1247), German (877), Polish (599), Hungarian (413), Indonesian (197), English (9) and others which are not defined further (125). As far as I know, it is the only learner corpus for the Dutch language. The corpus is used as a source of language errors produced by language learners. A subset of sentences will be drawn from the corpus and evaluated by human annotators which will yield important information about the errors that are found this way, such as the mistaken preposition and the error type. The corpus is thus used for the final assessment of the implementation of the method described below. It is unknown how suitable this corpus is for the evaluation of preposition error correction, as the corpus is not error-annotated and no similar experiments have been conducted yet. This also means that the density of (preposition) errors is unknown.

The corpus was manually converted from hand-written essays to XML-formatted documents. This means that parts were rendered unreadable sometimes. I decided to skip these sentences, as there is no way to find out what the desired sentence
had to have been. Other than that, no preprocessing was done.

3.3 Approach

In order to detect possible preposition errors, one needs to be able to discriminate between good language use and bad language use. This was done by building preposition models based on correct language from the LassyLarge corpus. The assumption is that these models only contain correct language and that they express sufficient knowledge to make the desired discriminations. Because of the size and motivation of these models, my approach to the task of preposition error detection and correction can be described as language modelling.

3.3.1 Decision model

In order to be able to automatically evaluate a sentence or text written by a learner, a decision model is needed. The decision model described in this section is a general method to how the insertion, substitution and deletion error types can be identified and corrected automatically. The procedure of detection and correction, given the three aforementioned error types and a sentence written by a learner, can be structured into Algorithm 1. The algorithm explains that a sentence is chunked into context vectors. The system iterates through these vectors and compares the decisions made by the system to those made by the learner. Explanatory comments are provided between curly brackets.

Algorithm 1 General decision algorithm to create an error report of a single sentence written by a language learner.

1: \textit{sentence} $\leftarrow$ \textit{Sentence written by language learner}
2: \textit{vectors} $\leftarrow$ \textit{Preposition vectors from sentence} \{Vector conversion is explained in Sections 3.3.2 - 3.3.3\}
3: \textbf{for all} \textit{vector} \textbf{in} \textit{vectors} \textbf{do}
4: \hspace{1em} \textit{label} $\leftarrow$ \textit{anchor of vector} \{the preposition or whitespace.\}
5: \hspace{1em} \textbf{if} \textit{label} \textbf{is any preposition} \textbf{then}
6: \hspace{2em} \{A preposition was used, vector marked as ‘present’.\}
7: reference ← present
8: else
9:  {No preposition was used, vector marked as ‘absent’.}
10: reference ← absent
11: end if
12: detection ← Binary prediction from Detection Model
13: if detection is true then
14:  {Preposition was expected by system, mark as ‘present’.
15:  machine detection ← present
16: else
17:  {Preposition not expected by system, mark as ‘absent’.
18:  machine detection ← absent
19: end if
20: if reference equals present then
21:  if machine detection equals present then
22:  {Preposition was both used by learner and expected by system}
23:  machine selection ← Preposition prediction from Selection Model
24:  if label equals machine selection then
25:  {Preposition used in sentence matches preposition predicted by system.}
26:    error message ← correct
27:  else
28:  {Preposition used in sentence does not match prediction by system.}
29:    error message ← substitution
30:    correction task ← replace label with machine selection
31:  end if
32: else
33:  {Preposition was used but not expected by system}
34:    error message ← insertion
35:    correction task ← delete label from sentence
36:  end if
37: else
if machine detection equals absent then
   {Preposition was neither used nor expected by system.}
   error message ← correct
else
   {No preposition was used but one was expected by system.}
   error message ← deletion
   machine selection ← Preposition prediction from Selection Model
   correction ← machine selection
end if
end if
end for

This algorithm has five different outcomes, as outlined in Table 3.1. Three of these express the identified error types, and a learner’s decision can be correct in two ways by correctly omitting or selecting a preposition.

Table 3.1: Different outcomes of decision model.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Detection</th>
<th>Selection</th>
<th>Error message</th>
<th>Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Yes</td>
<td>Prep X</td>
<td>Deletion</td>
<td>Insert X</td>
</tr>
<tr>
<td>Prep P</td>
<td>No</td>
<td>–</td>
<td>Insertion</td>
<td>Delete prep P</td>
</tr>
<tr>
<td>Prep P</td>
<td>Yes</td>
<td>Prep X</td>
<td>Substitution</td>
<td>Substitute P for X</td>
</tr>
<tr>
<td>Prep P</td>
<td>Yes</td>
<td>Prep P</td>
<td>Correct</td>
<td>–</td>
</tr>
<tr>
<td>None</td>
<td>No</td>
<td>–</td>
<td>Correct</td>
<td>–</td>
</tr>
</tbody>
</table>

The flowchart in Figure 3.1 aims to illustrate this algorithmic pipeline. Recall the candidates from example (22). They have been outlined in Table 3.2, which displays how these candidates are evaluated in an ideal situation (where the system does not make any errors). It explains the decision procedure by including hypothetical examples of every error type and it further shows how the deletion and insertion error types are mirrored.

For this decision algorithm, two language models are produced. The first model is a binary detection model (1 in flowchart, line 12 in Algorithm 1). This is used to predict whether the vector for a given event contains a preposition or not. Because
Table 3.2: Evaluating the candidates extracted from example (22).

<table>
<thead>
<tr>
<th>Reference</th>
<th>Detection</th>
<th>Selection</th>
<th>Error message</th>
<th>Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>voor_al</td>
<td>No</td>
<td>–</td>
<td>Correct</td>
<td>–</td>
</tr>
<tr>
<td>in</td>
<td>Yes</td>
<td>naar</td>
<td>Substitution</td>
<td>Replace <em>in</em> with <em>naar</em></td>
</tr>
<tr>
<td>over</td>
<td>Yes</td>
<td>over</td>
<td>Correct</td>
<td>–</td>
</tr>
<tr>
<td>had_2012</td>
<td>Yes</td>
<td>in</td>
<td>Deletion</td>
<td>Insert <em>in</em> between <em>had</em> and <em>2012</em></td>
</tr>
<tr>
<td>om</td>
<td>Yes</td>
<td>om</td>
<td>Correct</td>
<td>–</td>
</tr>
<tr>
<td>door</td>
<td>Yes</td>
<td>door</td>
<td>Correct</td>
<td>–</td>
</tr>
<tr>
<td>mee</td>
<td>No</td>
<td>–</td>
<td>Insertion</td>
<td>Delete <em>mee</em></td>
</tr>
</tbody>
</table>

A sentence written by a learner is chunked into these vectors, it can also be used to predict whether a preposition is necessary. The second model is the selection model (2 in flowchart and lines 23 and 44 in Algorithm 1), a classifier built on fifteen preposition classes outlined in Table 1.1, used to select a preposition if none was present (line 44 in Algorithm 1), or if the present preposition was wrong and select an appropriate correction (line 23 in Algorithm 1).

A vector from a learner’s sentence is presented to the detection model first. If the model outputs Yes, then the model effectively states that there should be a preposition. If the vector was not extracted around a preposition, the algorithm sees this as a deletion error: No preposition was used, but a preposition was expected. If we swap this, it means that a preposition was used, but it was not expected, resulting in an insertion error. Note that even if there is agreement when a preposition is used, it is still possible that it was the wrong preposition. Furthermore, when a deletion error is detected, the next step is to select a suitable preposition for that context. Selecting and substituting prepositions is done by the selection model. If the prediction of a selection model does not match the selection by a language learner, a substitution error is identified. The subsequent correction is to substitute the learner’s used preposition with the prediction by the machine. Complete agreement can occur when both the learner and the machine omit a preposition or when they select the same preposition.
Figure 3.1: The flowchart of the decision model, describing the general decision-making algorithm.
In the sections below, the creation and population of the language models will be explained in more detail. Note that this is only a classification procedure and is therefore not directly applied to any educative feedback methods.

### 3.3.2 Detection model

This model is used to detect insertion and deletion errors. The generality is that, given a feature-vector, the model predicts whether this vector is built around a white-space (an ‘absent’ preposition) or a ‘present’ preposition. The detection model relies on a set of linguistic features that occur around prepositions or whitespaces for a set of 2,000,000 vectors in order to allow the computer to form an understanding of when prepositions must be used or omitted. I included eight Ngram-based features. They are described in Table 3.3.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>bigram_left</td>
<td>Token bigram left of preposition</td>
<td>Token</td>
</tr>
<tr>
<td>bigram_right</td>
<td>Token bigram right of preposition</td>
<td>Token</td>
</tr>
<tr>
<td>trigram_left</td>
<td>Token trigram left of preposition</td>
<td>Token</td>
</tr>
<tr>
<td>trigram_right</td>
<td>Token trigram right of preposition</td>
<td>Token</td>
</tr>
<tr>
<td>bigram_postags_left</td>
<td>POS bigram left of preposition</td>
<td>POS</td>
</tr>
<tr>
<td>bigram_postags_right</td>
<td>POS bigram right of preposition</td>
<td>POS</td>
</tr>
<tr>
<td>trigram_postags_left</td>
<td>POS trigram left of preposition</td>
<td>POS</td>
</tr>
<tr>
<td>trigram_postags_right</td>
<td>POS trigram right of preposition</td>
<td>POS</td>
</tr>
</tbody>
</table>

The features denote the context in which the preposition or whitespace occurs. An attribute selection experiment\(^2\) showed that surrounding POS N-grams are highly indicative when discerning between preposition absence or presence. I did not want to choose too many complex features, because it is a binary classification task and I assumed these features would be quite effective already. The top 15 attributes from a feature-selection experiment (using Information Gain) (which contained the same features as for the selection model) show that POS N-grams dominate the results (Table 3.4). Note that PRE\_noun\#False denotes the absence of a preceding noun, and the same goes for PRE\_verb\#False.

\(^2\)Based on an equally distributed set of 150,000 events using WeKa [Hall et al., 2009](#)
### Table 3.4: Preliminary feature-selection results on Information Gain.

<table>
<thead>
<tr>
<th>InfoGain score</th>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.242</td>
<td>bigram_postags_right</td>
<td>verb_det</td>
</tr>
<tr>
<td>0.211</td>
<td>trigram_postags_right</td>
<td>verb_det_noun</td>
</tr>
<tr>
<td>0.177</td>
<td>bigram_postags_right</td>
<td>det_noun</td>
</tr>
<tr>
<td>0.133</td>
<td>PRE_noun</td>
<td>False</td>
</tr>
<tr>
<td>0.125</td>
<td>PRE_verb</td>
<td>False</td>
</tr>
<tr>
<td>0.054</td>
<td>bigram_postags_right</td>
<td>det_name</td>
</tr>
<tr>
<td>0.052</td>
<td>bigram_postags_right</td>
<td>det_adj</td>
</tr>
<tr>
<td>0.048</td>
<td>trigram_postags_right</td>
<td>det_noun_punct</td>
</tr>
<tr>
<td>0.048</td>
<td>trigram_postags_right</td>
<td>det_adj_noun</td>
</tr>
<tr>
<td>0.048</td>
<td>trigram_postags_right</td>
<td>adv_det_noun</td>
</tr>
<tr>
<td>0.048</td>
<td>bigram_postags_right</td>
<td>adv_det</td>
</tr>
<tr>
<td>0.046</td>
<td>trigram_postags_right</td>
<td>name_name_punct</td>
</tr>
<tr>
<td>0.046</td>
<td>bigram_postags_left</td>
<td>det_noun</td>
</tr>
<tr>
<td>0.046</td>
<td>bigram_postags_right</td>
<td>name_name</td>
</tr>
<tr>
<td>0.044</td>
<td>bigram_postags_right</td>
<td>noun_det</td>
</tr>
</tbody>
</table>

### Modelling preposition absence

Cases of preposition absence are not extracted in the same way as cases of preposition presence (preposition cases). Note that a vector is basically extracted around a certain anchor-point in a sentence. The anchor-point is the position of the preposition, and information around that preposition is extracted. Obviously, the same cannot be done for cases without prepositions.

Note that the most basic case of preposition absence is any whitespace (note that non-preposition words also count, but insertion errors are inserted between words rather than instead of them) between two words, or before the first or after the last word of a sentence.

These whitespaces express preposition absence and could therefore be converted to vectors. However, in the scope of actual language use, it makes little sense that whitespace should be treated equally. After all, as indicated in the table, prepositions do not occur everywhere, and should therefore not be expected everywhere when looking at learner data. In conclusion, including every whitespace
would result in a high amount of trivial vectors which do not correctly indicate situations in which prepositions are specifically omitted.

Furthermore, if every whitespace were to be included, many vectors would express a high degree of overlap, because the knowledge on which this system relies is based on linguistic features that surround a whitespace or preposition. Therefore, two whitespaces to the left and right of a certain word would contain a lot of the same feature values (such as neighbouring N-grams), which could result in an overfitting of the model. This is especially true if I limit the system to a fixed amount of vectors. If the average sentence in a corpus has a length of fourteen words, it can be said that there are roughly thirteen whitespaces. A desired training size of a million vectors could then be drawn from less than 77K sentences. Adopting a filter that only yields relevant whitespaces decreases the amount of vectors per sentence because many whitespaces are omitted. If only 30% of whitespaces remain, the same data size would be drawn from over 250K sentences. I assume this contains more diverse information as well.

What is needed therefore, is a way of intelligently pinpointing whitespaces in sentences that could, based on their surrounding structure, hypothetically harbour a preposition (but do not!). This would result in absent preposition vectors that resemble the positive ones more.

To do this, I extracted a simple frequency distribution of Part-of-Speech patterns from the previously trained native set, which contains true preposition cases. For this, the structure \(POS1\_POS2\_PREP\_POS3\_POS4\) ought to be assumed. The surrounding Part-of-Speech bigrams make up for the Part-of-Speech pattern. The unique set of these, based on 1,000,000 true events, are used for a comparison experiment. Here, raw Part-of-Speech patterns are extracted in two ways: Around prepositions (such as \(POS1\_POS2\_PREP\_POS3\_POS4\)), and as basic four-grams (\(POS1\_POS2\_POS3\_POS4\)). When extracted around prepositions, they are true cases. When extracted as bare four-grams, they express preposition absence patterns. This results in two frequency distributions: Those for true cases, and those for false cases. I compare the two by selecting those patterns that occur in true cases for 90% of the time. This results in a subset of part-of-speech patterns that lean towards true cases.

Detecting intelligent absent preposition cases is consequently done by chunking

\(^{3}\)This is hypothetical. Actual statistics were not calculated, so I took a wide estimate.
the sentences into Part-of-Speech fourgrams. If a fourgram expressing preposition absence is found which is also in the set of patterns that lean towards presence, it is marked as a non-trivial absent case and converted into a vector accordingly, because it represents a case in which prepositions are normally used, so opting to omit can be seen as a non-trivial decision which is something where a language learner might err. The vector is built around the word to the left of the whitespace (which is essentially the whitespace between POS-tokens 2 and 3 of the observed Part-of-Speech pattern).

For example, the pattern \textit{NOUN\_DET\_NOUN\_VERB} is observed to be much more frequent among presence cases than absence cases because prepositions rarely occur between determiners and nouns.

Because of the high bias interval (90\%), the amount of cases of absence from the native corpus resulted in only just over 2 million. I think this will suffice for a binary classifier. I found no comparable method in related work, and some reports discarded the deletion and insertion tasks altogether. This also means that it is wholly unclear whether the method will work and how accurate it will be.

### 3.3.3 Selection model

This model predicts the preposition for a vector from any sentence. An important difference between this selection procedure and normal text classification is that disagreements (in substitution tests) are just as interesting as agreements, if not more so. This is because a disagreement means that the model considers a different preposition to be more suitable than the one which was actually used, which directly translates to the detection of a substitution error, which is only possible when comparing the likelihood of the used preposition with the other prepositions in the model. This model exploits a set of 20,000,000 from native Dutch preposition events. This number is ten times as big as for the detection model. Because the model discriminates between fifteen different prepositions (Table 1.1) instead of just a binary preposition decision distribution like the detection model does, the larger number of events is well-justified. All fifteen prepositions must be represented sufficiently to reduce the chance that the classifier is biased towards richly populated prepositions. For every preposition, a model is built by collecting linguistic information from preposition events from the native data into vectors. The
features I used are outlined in Table 3.5 and were largely inspired by Chodorow et al. [2007].

Table 3.5: The set of features for the substitution task. The level-column refers to the syntactic level of the type of attribute.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bigram_left</td>
<td>Token bigram left of preposition</td>
</tr>
<tr>
<td>bigram_right</td>
<td>Token bigram right of preposition</td>
</tr>
<tr>
<td>trigram_left</td>
<td>Token trigram left of preposition</td>
</tr>
<tr>
<td>trigram_right</td>
<td>Token trigram right of preposition</td>
</tr>
<tr>
<td>PRE_verb</td>
<td>Verb preceding the preposition</td>
</tr>
<tr>
<td>FOLL_verb</td>
<td>Verb following the preposition</td>
</tr>
<tr>
<td>PRE_noun</td>
<td>Noun preceding the preposition</td>
</tr>
<tr>
<td>FOLL_noun</td>
<td>Noun following the preposition</td>
</tr>
<tr>
<td>bigram_postags_left</td>
<td>POS bigram left of preposition</td>
</tr>
<tr>
<td>bigram_postags_right</td>
<td>POS bigram right of preposition</td>
</tr>
<tr>
<td>trigram_postags_left</td>
<td>POS trigram left of preposition</td>
</tr>
<tr>
<td>trigram_postags_right</td>
<td>POS trigram right of preposition</td>
</tr>
<tr>
<td>FOLL_phr_head</td>
<td>Headword of following phrase</td>
</tr>
<tr>
<td>PRE_phr</td>
<td>Preceding phrase-type</td>
</tr>
<tr>
<td>FOLL_phr</td>
<td>Following phrase-type</td>
</tr>
<tr>
<td>Preposition</td>
<td>The preposition</td>
</tr>
</tbody>
</table>

Because of the broad coverage these features have and the multiple levels of depth, I think they are suitable for this task. They depict the preposition occurrence accurately in the sense that a lot of information surrounding the preposition can be extracted. A sample sentence, represented as a dependency tree from Alpino can be seen in Figure 3.2. The featurisation of this sentence is consequently described in Table 3.6.

I briefly considered converting nouns to respective semantic synonyms (automobile ← car) or superclasses (car ← vehicle) in order to generalise the model which would address possible sparsity. While this might make sense in some situations, especially when using less training data, I think it would be less effective or even harmful if done so for preposition error correction. This is because different preposi-
Itions are used in situations that might seem similar when adopting a generalisation method such as converting trigrams to only their respective POS counterparts, thus losing the important discrepancy that only tokens might be able to convey. In
other words, the desired preposition(s) severely depends on the raw tokens around it. Furthermore, prepositions are used so commonly that acquiring a sufficient amount of data to address sparsity should not be a problem. The basic examples in (24 – 29) illustrate why generalisation is potentially harmful for this task.

(24) Ik zit in mijn stoel.
(25) I am sitting in my chair.
(26) Ik zit op mijn stoel.
(27) Ik zit op de bank.
(28) I am sitting on the sofa.
(29) *Ik zit in de bank.

Consider that both a chair and a sofa are direct hypernyms of seat, according to WordNet. The same can be said for the Dutch translations (zitting, zitplaats). Here, the two prepositions cannot be used interchangeably as shown in (29).

The training algorithm is expressed in Figure 2. Preposition vectors were extracted by establishing the node of the preposition (let this be the anchor node, identifiable by an ID-attribute) and extracting information from the nodes around the anchor node. The readily available parse-trees make this trivial. A small difficulty is that the id-attribute does not always correspond to the literal position of the node or word in the sentence. A manual mapping of the nodes to their corresponding words was done in order to acquire the literal node order of the sentence.

The next section outlines how the performance of these models within the pipeline can be evaluated empirically and how these results can be interpreted.

3.4 Evaluation

The evaluation of the models within the pipeline is done in two distinct ways. The standard way of assessing classification quality can be done by running the system on unseen native data (or cross-validation on training data). Consider that the models are built in order to detect and correct incorrect use of language. Evaluation on native data rather than learners’ data thus answers the question
Algorithm 2 Standard training algorithm.

```
native treebank ← Lassy Large corpus
training data ← empty collection

for all parse tree in native treebank do
    preposition hits ← parse tree.xpath("//node[@pt='vz']")
    if preposition hits is not empty then
        {If there are preposition events in the sentence at all}
        for all preposition in preposition hits do
            vector ← extract preposition vector from parse tree
            training data ← training data + preposition vector
        end for
    end if
end for

preposition model ← build model from training data
```

‘How well do these models profile correct use of language?’ This is an immediate glimpse into their potential to detect incorrect language, which is the second way of evaluating. After all, being able to discern correct from incorrect requires solid knowledge into correct language as a start. Errors that occur there will also occur when detecting incorrect language. The philosophy behind a model built on native data is that any language which is not identified as correct by the model, must be incorrect. Ideally, such a model does not recognize incorrect language use as correct, and will also never recognize correct language use as incorrect.

3.4.1 Baselines

For both models, appropriate baselines are needed in order to assess how well the models work compared to more basic approaches. For both models, three baselines were developed, as there does not seem to be a clearly defined baseline. Papers described in the relevant work section seem to omit these as well [Nagata et al., 2014, Felice and Pulman, 2009, Chodorow et al., 2007]. I hypothesise that these baseline models will be outperformed by the proposed methods, as the proposed methods convey a deeper representation of the preposition context.
Detection model

A silly baseline is simply always predicting no preposition. Because prepositions make up for 15% of the tokens, a naive assumption is that in 85% of the time, this would be correct. The second baseline is a bit more sophisticated, being trained on the two hundred most common POS patterns for preposition presence cases. If the probability that a certain pattern is more likely to occur for a preposition case than for an absence case, the answer is True. If not, the answer is False. The third baseline is the same, except that this model is trained on every POS pattern, identified as the unique union of preposition absence and presence POS patterns.

Selection model

For the selection model, a similarly silly baseline is to simply always predict that the preposition should be van, since that is the most frequent preposition. The second baseline is a classifier trained on only the surrounding POS bigrams. A third, rather generous baseline is to include the surrounding token-level trigrams. The assumption is that the order in which these have been introduced is also the ascending order of their quality.

3.4.2 Native data

For this part of the evaluation, the model is evaluated on the classifying labels. Here, a disagreement is factually a classification failure. After all, the actual label represents the gold standard. This part of the evaluation is strongly like other multi-class classification tasks and simply answers whether it is possible to create accurate models of Dutch prepositions.

3.4.3 Learners’ data

The actual and most honest evaluation is to be done on learner’s data. In order to assess the quality of the system, knowledge on the learner’s data is required. Because the corpus I used is not error-annotated, it makes sense to gather human judgements on preposition gap texts. This was performed through a website which could be accessed by a lot of different people: Voorzetselaar.nl. The website lets
people create an account and annotate sentences. For every sentence, different events were extracted. People will be presented with a random sentence with a gap in it. Then, they have to select up to five prepositions which they deem suitable, or they can press ‘Niks past’ (nothing fits) if the gap should remain a whitespace. These events are direct outputs of the system and this results in an easy way of gathering a solid database of correct answers. It is important to make sure that people do not get the impression that they are factually judging a system, as it might cause bias. People might be inclined to penalize or forgive system errors, whereas evaluation must be done objectively. Simply prompting people to pick prepositions will not give this impression. Previous work has emphasised that crowdsourcing is a good way to receive many annotations in a relatively short amount of time and that results are potentially as good as expert annotation [Snow et al., 2008].

Furthermore, this will always prompt them to make an intelligent decision, as to reduce the likelihood that people will get bored quickly. Why? Consider that it is unknown how many errors there are. Asking people to annotate errors where they would have to mark errors based on highlighted preposition cases, could be a very tedious task as these errors might be very sparse, so they would have to press No rather often. For these reasons, I decided to do it this way. Screenshots from the annotation interface (Figure 3.3) and the instructions (Figure 3.4) are included. I tried to entice more people to join by adding the possibility of winning a prize for the person who annotated the most.

Additionally, it creates the possibility of comparing the acceptable prepositions with the actual distribution of prepositions provided in the chapter The Dutch preposition. If multiple prepositions are deemed acceptable for a random preposition event, it is interesting to find out if there is a strong correlation between the prepositions ranked by frequency and their likelihood of being an acceptable decision.

Eventually, three sets of labels will be available. Those of the learners, the system output, and the gold data set acquired by Voorzetseelaar.nl. It means that for every event, there will be a triple of answers and three pairs of evaluation. For this part of the evaluation, it makes sense to separate each pair and describe what their results mean. View Figure 3.5 for a general overview.

The chart describes how every event is evaluated. Consider that the first de-
cision diamond (1) refers to the entire procedure by the decision model described earlier this chapter. Discarding the discrepancy between different errors for this
chart, any error detection (and implied correction) is the system triggering a positive. Consequently, any agreement between learner and system is the system triggering a negative. Next, the system output is compared to the gold standard. Note that the gold standard is a set rather than a single value. The set can contain between one and five values. For every event, all the annotations are counted. The five most selected prepositions per event are included into the gold standard set.

If the system output is part of this gold standard set, it is marked as true. If it is not, then it will be marked as false. The combination of the system output (positive or negative) with its comparison to the gold standard set provides the logical distribution of true/false positives/negatives which can be used to compute the final performance of the pipeline.
Figure 3.5: The flowchart of the evaluation model.

Start

Do learner and system agree?

Gold standard and system output

Result:

+ System performance + System guess

Do learner and system agree?

System guess:

→ positive

Yes

System performance:

→ true

End:

yes

no

→ System guess

→ System performance

→ positive

→ negative

→ positive

no

→ negative

Start
Chapter 4

Results

This chapter describes all the empirical results that were acquired during the development of this system, including results on both native test data and learner test data and comparative baseline results.

4.1 Native test data

Test data was drawn from the Lassy Small corpus, with a total number of 268,895 events. Table 4.1 outlines the detection results (including results for the described baselines) on native test data, which only classifies either preposition absence or presence given a vector.

Table 4.1: Baseline detection comparison on native data

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed model</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Baseline 1</td>
<td>25%</td>
<td>50%</td>
<td>33%</td>
</tr>
<tr>
<td>Baseline 2</td>
<td>75%</td>
<td>51%</td>
<td>36%</td>
</tr>
<tr>
<td>Baseline 3</td>
<td>97%</td>
<td>97%</td>
<td>64%</td>
</tr>
</tbody>
</table>

Table 4.2 outlines the selection results on native test data. The test data was extracted in a similar way as the training data and thus has the same structure.
Table 4.2: Selection performance on native test set.

<table>
<thead>
<tr>
<th>Preposition</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>aan</td>
<td>0.72</td>
<td>0.69</td>
<td>0.71</td>
<td>11690</td>
</tr>
<tr>
<td>als</td>
<td>0.81</td>
<td>0.51</td>
<td>0.63</td>
<td>5157</td>
</tr>
<tr>
<td>bij</td>
<td>0.57</td>
<td>0.43</td>
<td>0.49</td>
<td>8369</td>
</tr>
<tr>
<td>door</td>
<td>0.66</td>
<td>0.55</td>
<td>0.60</td>
<td>9154</td>
</tr>
<tr>
<td>in</td>
<td>0.72</td>
<td>0.81</td>
<td>0.76</td>
<td>47649</td>
</tr>
<tr>
<td>met</td>
<td>0.67</td>
<td>0.67</td>
<td>0.67</td>
<td>17167</td>
</tr>
<tr>
<td>naar</td>
<td>0.69</td>
<td>0.62</td>
<td>0.65</td>
<td>6232</td>
</tr>
<tr>
<td>om</td>
<td>0.77</td>
<td>0.69</td>
<td>0.73</td>
<td>10881</td>
</tr>
<tr>
<td>op</td>
<td>0.78</td>
<td>0.78</td>
<td>0.78</td>
<td>23111</td>
</tr>
<tr>
<td>over</td>
<td>0.65</td>
<td>0.59</td>
<td>0.62</td>
<td>6036</td>
</tr>
<tr>
<td>te</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>19746</td>
</tr>
<tr>
<td>tot</td>
<td>0.75</td>
<td>0.68</td>
<td>0.71</td>
<td>6703</td>
</tr>
<tr>
<td>uit</td>
<td>0.69</td>
<td>0.56</td>
<td>0.62</td>
<td>6514</td>
</tr>
<tr>
<td>van</td>
<td>0.80</td>
<td>0.87</td>
<td>0.83</td>
<td>71102</td>
</tr>
<tr>
<td>voor</td>
<td>0.64</td>
<td>0.58</td>
<td>0.61</td>
<td>19380</td>
</tr>
<tr>
<td>Average</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>268895</td>
</tr>
</tbody>
</table>

The comparison of the baseline models with the proposed model is outlined in Table 4.3.

Table 4.3: Baseline selection comparison on native data

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed model</td>
<td>75%</td>
<td>75%</td>
<td>75%</td>
</tr>
<tr>
<td>Baseline 1</td>
<td>7%</td>
<td>26%</td>
<td>11%</td>
</tr>
<tr>
<td>Baseline 2</td>
<td>38%</td>
<td>42%</td>
<td>35%</td>
</tr>
<tr>
<td>Baseline 3</td>
<td>65%</td>
<td>65%</td>
<td>64%</td>
</tr>
</tbody>
</table>

Testing on native test data rather than on learner data gives an impression of the quality of the model when there is little noise. Indeed, one can assume that learner data will contain more erroneous spelling but less variety in structure because of the fact that most data from the Dutch Learning Corpus consists of student essays. Erroneous spelling, poor lexical choice and grammatical errors will
likely contribute to the results on learner test data.

4.2 Learner test data

4.2.1 Voorzetselaar.nl

The website turned out to be a solid way of gathering relevant results. In a span of four weeks, over 2,000 annotations were gathered by over thirty annotators. Initially, I wanted to compute the kappa of every event in order to find out how much diversity existed between the annotator’s choices, but because the amount of annotations per event were so diverse, I decided to manually go through the output in order to establish a convenient gold standard. Many events received different amounts of annotations and every annotation contained between one and five prepositions so it would be difficult to compute inter-annotator agreement effectively. Additionally, there were some obvious mistakes by users which were also filtered out manually. An automated judge would not have been able to detect these. Finally, about eight sentences were too difficult to judge for even humans because of all the spelling errors and poor lexical choices, and were removed from the final test set.

The prize incentive did not seem to make a huge difference. Many people annotated only a few sentences, meaning that almost all annotations came from a group of five people.

4.2.2 Classifier performance

The model is created to attest if one can leverage native language to detect and correct incorrect language use, typically by language learners. Because detection and correction are two different tasks, it makes sense to discriminate between these two in the result set. Assessing detection performance can be done by evaluating how well the model is able to detect either of four error types: insertion, deletion, substitution and a correct case, where the model simply states that a certain case is correct and needs no correction.
Detection

The confusion matrix from Table 4.4 and the classification report from Table 4.5 express the results for detection on the established test set of 1,499 annotated examples.

**Table 4.4: Error type detection results**

<table>
<thead>
<tr>
<th></th>
<th>Correct (without)</th>
<th>Correct (with)</th>
<th>Deletion</th>
<th>Insertion</th>
<th>Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct (without)</td>
<td>16</td>
<td>(0)</td>
<td>488</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>Correct (with)</td>
<td>(0)</td>
<td>478</td>
<td>0</td>
<td>17</td>
<td>368</td>
</tr>
<tr>
<td>Deletion</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Insertion</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>Substitution</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>65</td>
</tr>
</tbody>
</table>

**Table 4.5: Performance report for detection on learner test set.**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct (without)</td>
<td>0.94</td>
<td>0.03</td>
<td>0.06</td>
<td>527</td>
</tr>
<tr>
<td>Correct (with)</td>
<td>0.99</td>
<td>0.55</td>
<td>0.71</td>
<td>867</td>
</tr>
<tr>
<td>Deletion</td>
<td>0.01</td>
<td>0.75</td>
<td>0.01</td>
<td>4</td>
</tr>
<tr>
<td>Insertion</td>
<td>0.22</td>
<td>0.21</td>
<td>0.21</td>
<td>29</td>
</tr>
<tr>
<td>Substitution</td>
<td>0.14</td>
<td>0.90</td>
<td>0.24</td>
<td>72</td>
</tr>
<tr>
<td>Average/Total</td>
<td>0.91</td>
<td>0.38</td>
<td>0.45</td>
<td>1499</td>
</tr>
<tr>
<td>Baseline 1</td>
<td>0.14</td>
<td>0.37</td>
<td>0.20</td>
<td>–</td>
</tr>
<tr>
<td>Baseline 2</td>
<td>0.39</td>
<td>0.37</td>
<td>0.22</td>
<td>–</td>
</tr>
<tr>
<td>Baseline 3</td>
<td>0.43</td>
<td>0.55</td>
<td>0.47</td>
<td>–</td>
</tr>
</tbody>
</table>

The total recall and precision on basic **error** recognition, being the discrimination between correct and incorrect, is respectively 71% and 8%. If I discard the deletion error type altogether, recall would be 69% and precision would be 14%,
eliminating a class with relatively high recall and low precision. Finally, accuracy on case type recognition, being the discrimination between any type of detection, is 38%.

Selection

The confusion matrix from Table 4.6 expresses the results on selection. The selection results are based on the subset of cases where the actual placeholder was a preposition and not a whitespace. For this purpose, system errors where a preposition was left out were deleted. The report of this confusion matrix is subsequently displayed in Table 4.7.

Table 4.6: Confusion matrix for machine selection (columns) and gold standard annotation (rows)

<table>
<thead>
<tr>
<th></th>
<th>aan</th>
<th>als</th>
<th>bij</th>
<th>door</th>
<th>in</th>
<th>met</th>
<th>naar</th>
<th>om</th>
<th>op</th>
<th>over</th>
<th>te</th>
<th>tot</th>
<th>uit</th>
<th>van</th>
<th>voor</th>
</tr>
</thead>
<tbody>
<tr>
<td>aan</td>
<td>24</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>als</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>bij</td>
<td>2</td>
<td>0</td>
<td>14</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>door</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>in</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>117</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>met</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>15</td>
<td>41</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>naar</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>12</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>om</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>35</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>op</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>48</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>over</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>te</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>87</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>tot</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>uit</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>van</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93</td>
<td>8</td>
</tr>
<tr>
<td>voor</td>
<td>5</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>8</td>
<td>12</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>70</td>
</tr>
</tbody>
</table>

The accuracy on selection performance is 60%.
Table 4.7: Classification report on selection results

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>aan</td>
<td>0.53</td>
<td>0.51</td>
<td>0.52</td>
<td>47</td>
</tr>
<tr>
<td>als</td>
<td>0.50</td>
<td>0.07</td>
<td>0.12</td>
<td>15</td>
</tr>
<tr>
<td>bij</td>
<td>0.28</td>
<td>0.33</td>
<td>0.30</td>
<td>43</td>
</tr>
<tr>
<td>door</td>
<td>0.11</td>
<td>0.11</td>
<td>0.11</td>
<td>19</td>
</tr>
<tr>
<td>in</td>
<td>0.56</td>
<td>0.78</td>
<td>0.65</td>
<td>150</td>
</tr>
<tr>
<td>met</td>
<td>0.49</td>
<td>0.50</td>
<td>0.49</td>
<td>82</td>
</tr>
<tr>
<td>naar</td>
<td>0.67</td>
<td>0.55</td>
<td>0.60</td>
<td>22</td>
</tr>
<tr>
<td>om</td>
<td>0.52</td>
<td>0.65</td>
<td>0.58</td>
<td>54</td>
</tr>
<tr>
<td>op</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>77</td>
</tr>
<tr>
<td>over</td>
<td>0.54</td>
<td>0.51</td>
<td>0.53</td>
<td>39</td>
</tr>
<tr>
<td>te</td>
<td>0.98</td>
<td>0.88</td>
<td>0.93</td>
<td>99</td>
</tr>
<tr>
<td>tot</td>
<td>0.62</td>
<td>0.38</td>
<td>0.48</td>
<td>13</td>
</tr>
<tr>
<td>uit</td>
<td>0.47</td>
<td>0.58</td>
<td>0.52</td>
<td>12</td>
</tr>
<tr>
<td>van</td>
<td>0.76</td>
<td>0.66</td>
<td>0.71</td>
<td>140</td>
</tr>
<tr>
<td>voor</td>
<td>0.70</td>
<td>0.56</td>
<td>0.62</td>
<td>126</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.61</strong></td>
<td><strong>0.60</strong></td>
<td><strong>0.60</strong></td>
<td><strong>964</strong></td>
</tr>
<tr>
<td>Baseline 1</td>
<td>0.02</td>
<td>0.14</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>Baseline 2</td>
<td>0.35</td>
<td>0.33</td>
<td>0.27</td>
<td></td>
</tr>
<tr>
<td>Baseline 3</td>
<td>0.45</td>
<td>0.48</td>
<td>0.45</td>
<td></td>
</tr>
</tbody>
</table>

4.2.3 Evaluation learner corpus

The new learner corpus for Dutch deserves to be examined and used more thoroughly. A learner corpus ought to be a collection of typical language use by learners. As learners are expected to err, evaluating such a corpus can be done by assessing how dense it is regarding errors of language. The annotated test set proves to be a convenient way of estimating how densely populated the corpus is by preposition errors.

Error density

Summing up all true error types in the test set, there was a total of 111 preposition-related errors in a test set of 1,499, which means that about 7% of all preposition
decisions[^1] end up being an error. About 65% (72) of these are substitution errors, the most straightforward type of error. About 28% of the errors are insertion errors. The remainder consists of a nearly negligible 7 deletion errors (less than 7%).

**Common confusion**

Instead of producing a confusion matrix where machine selection and detections are compared to those of the annotated gold standard, one can compare the decisions of learners to the gold standard in order to visualise the difficulty of preposition use for language learners. The confusion matrix is presented in Table 4.8.

Table 4.8: Confusion matrix for learner selection (columns) and gold standard annotations (rows)

<table>
<thead>
<tr>
<th></th>
<th>aan</th>
<th>als</th>
<th>bij</th>
<th>door</th>
<th>in</th>
<th>met</th>
<th>naar</th>
<th>om</th>
<th>op</th>
<th>over</th>
<th>te</th>
<th>tot</th>
<th>uit</th>
<th>van</th>
<th>voor</th>
</tr>
</thead>
<tbody>
<tr>
<td>aan</td>
<td>38</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>als</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>bij</td>
<td>5</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>door</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>in</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>125</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>met</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>69</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>naar</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>om</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>43</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>op</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>61</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>over</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>te</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>98</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>tot</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>uit</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>van</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>130</td>
</tr>
<tr>
<td>voor</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>

The next chapter outlines detailed discussion of the empirical results in order

[^1]: Since deletion candidates were generated artificially due to there not being a clear prepositional anchor-point, this is not a true estimate of the degree of deletion errors.
to shape a solid understanding of why the problem is difficult to solve, create
the possibility of interpreting these results correctly and raise questions about the
nature of this approach.
Chapter 5

Discussion

This chapter includes the full analysis of the empirically acquired results. It is meant to extract intelligent knowledge from these results in order to provide suggestions that may aid others who will tackle this issue in the future.

5.1 Results on native data

The results on native data, with a total F1-score of 75% show that it is indeed possible to quite accurately build a model that predicts what preposition is used in what context. This proposed method similarly outperforms the artificial baselines by at least 11 percentage points, indicating that a combined set of features from multiple levels of depth is well-grounded. However, reporting the total F1-score does not suffice; it is a weighted average of the already averaged precision and recall measures of every preposition class. This can be demonstrated by pointing to the weakest preposition class, bij, which achieved an F1-score of only 49% and the strongest, te, achieved one of 97%. This means that there is a lot of diversity between preposition classes and that there are still prepositions that deserve a lot of attention. As mentioned before, results on native data only provide insights in the potential of a model. As it is unlikely that results on learner data would be higher than on native data, we can consider the results on native data as the upper bound of the model.
5.2 Decision model

Error detection and classification is the essential first step into automatic error correction (and feedback), because any error that arises in this step will result in an error during the correction or selection step as well. The detection model has an average F-score of 45%. The system achieves over 95% precision for correct language, though it misses more than half of all correct cases, which translates into these cases being falsely marked as errors (generally **deletion errors** or **substitution errors**). As described in the Relevant Literature section, it is bad for the learning experience if an error is marked falsely. These false positives proved to be detrimental for the overall results, with a precision of only 1% on deletion errors.

The low precision on deletion errors logically explains the low recall on correct (and empty) preposition cases. I somewhat deliberately set the confidence threshold for the deletion error type a little lower because I was curious whether it would be able to achieve a reasonable recall, which it did (75%). Table 4.5 visualises how most correct cases are mistaken for deletion errors because of this low confidence threshold. Even though there is a high recall which shows that the concept of detecting and modelling relevant whitespace cases deserves more attention, it is clearly not ready in this state, because its precision is detrimental. However, for such a low amount of observed deletion errors, its relevance is questionable.

There is a lot of room for **insertion detection performance**. Relevant papers often state insertion and deletion as the most challenging error types (see Chapter 2). A precision of 22% is not that bad, with a similarly reasonable recall at 21%. Most errors that were meant to be insertion errors were actually marked as substitution errors by the system. Keep in mind that many correct cases were falsely marked as deletion errors. These two mistakes are typical: It means that the system often expects a preposition in many cases where there ought not be one. This means that the detection model is not ready for learner data yet because it errs excessively when predicting preposition absence or presence, which is in sharp contrast when looking at the results it achieved on native data where it produced virtually error-free classification.

However, previous work sometimes discarded the **deletion** error type ([Felie and Pulman 2009](#), [Nagata et al. 2014](#), [Chodorow et al. 2007](#)) disregarded these error types completely, [Chodorow et al. 2007](#) applied simple and shallow matching rules to detect inser-
tion errors and disregarded deletion errors), which seems to make sense considering that there were only four of the type, which is only about 3.8% of all errors.

If I were to consider the error detection task as a binary task with the classes Incorrect and Correct, in line with Felice and Pulman [2009], it would correctly find 90% of all (69 out of 74) incorrect prepositions at a precision of 14%. A reduced result set consisting only of these correctly identified substitution errors showed that 54% of these were appropriately corrected by the system. This means that 50.4% of all substitution errors are correctly identified and corrected. Even though a population of 72 is rather small, these results are encouraging. Felice and Pulman [2009] reported 39.5% for this setting. However, considering that the substitution error type detection has such a low precision shows that it is still not ready since there is still an excessive amount of false positives.

Most substitution errors were detected, with a solid recall of 90%, but its precision was much lower because of the fact that a lot of correct cases were incorrectly marked as substitution errors. This is probably due to the fact that only one preposition is picked by the model and any disagreement is viewed as an error.

In summary, the system is not accurate when discerning between preposition absence and presence, since it overpredicts preposition presence so much. A possible explanation is that I decided to extract preposition absence vectors in the model at places where a preposition was commonly expected, based on the POS-pattern that is found. An absent-preposition situation with a POS-pattern which is commonly found around a present-preposition situation has the advantage that there is a clear way of discriminating between any (possibly trivial) whitespace and whitespaces that are potentially challenging, given their context. The disadvantage is that this means that vectors of preposition absence and presence turn out to be quite similar in nature, which might have overfit the model to predict preposition presence when evaluating on learner data. This is emphasised by the fact that the third baseline for the detection model outperforms the proposed method, because the third baseline is built on the premise that preposition presence is expected based on the normal characteristics of the POS N-grams that surround it, and whether they normally do harbour a preposition or not. These patterns are much more distinct and they do not make the assumption that deletion errors are likely to occur.
The selection model is the final step of the pipeline. It performs the necessary selection procedure that completes error correction. The selection model selected missing prepositions and replaced prepositions in case they were deemed incorrect. It does not suffice to only look at substitution error correction here, because the strength of the system can be measured by the times it correctly indicated that a preposition was correct in a given context. Note that the discrepancy between substitution error correction and simple preposition prediction is still important. Felice and Pulman [2009] separated their test sets into two sets with either correct preposition use and wrong preposition use, as described in Chapter 2. The results of the combined setting in this thesis are denoted in the matrix in Table 4.8. This confusion matrix shows the totality of selection performance, regardless if they originate from a substitution error or correct language. Note that the excess of deletion errors result in a lot of false selections, because if the error is marked falsely, then the subsequent correction is also wrong.

5.2.1 Features

Another question that comes to mind is whether the features for the substitution task allow for an adequate discrepancy between different prepositions. What one must keep in mind is that these features are based on the feature set presented by Chodorow et al. [2007]. English was the target language in their work. It remains a question if this has an impact on the results. There were encouraging results on the native data with an F1-score of 75%, but perhaps even better features for Dutch exist. For example, an immediate difference between Dutch and English that comes to mind, is the construction of contractions in Dutch. Nouns can often be glued into a new type in Dutch. *Soup bowl* is an English construction for *a bowl for soup*. In Dutch, however, *een kom voor soep* can be simplified to *soepkom*, resulting in a single token rather than two.

Considering that a lot of the aforementioned set of features contain information of words *around* the preposition, this difference might affect the results.
5.2.2 Robustness

Result sets show a strong difference between native data and learner data. The selection model achieves a 75% F-score on native test data, but only 60% on correct learner test data, and 54% on incorrect learner test data (omitting deletion and insertion errors). Why is this? First of all, it deserves to be said that the native test data expresses is drawn from a similar set as the native training data, whereas the learner data is from a different and new corpus altogether. Furthermore, the learner data contains a lot of non-standard use of language, including many other types of errors, such as determiner errors, conjugation errors and spelling errors.

Secondly, the syntactic parsing layer, which is executed by Alpino has its own (possibly and probably imperfect) accuracy. On the corpus from which native data was extracted, the accuracy is reported at 93% [Bouma et al., 2000]. For the learner data, the accuracy is unknown. But due to the fact that there are many types of learner errors, the accuracy will likely be lower than on native data. This immediately results in a more erroneous feature extraction. Furthermore, wrong parses by Alpino from the native data, which typically occur when sentences are very long and/or express ambiguity, result in a faulty feature extraction.

Another thing to keep in mind is that for this task, the theoretical assumption is made that both corpora types, being the native and learner corpora, express the same nature. This assumption follows the strive for robustness: A robust system would be able to handle both types of corpora equally well. Firstly, features are extracted from native data around prepositions. As we can assume that the native data are virtually error-free, the extracted features will be relevant and linguistically more coherent. The same holds for native test data, hence the relatively good results on that data set. However, the assumption certainly does not hold for learner data. Consider that I only try to detect and correct preposition errors. If one could assume that the learner data only contained preposition errors, but was otherwise error-free, it would express the same structure as the native data, apart from the preposition errors. But for the possibility of actual preposition errors, the pipeline has to cope with the other types of errors as well.

Summarising, the models I created know a lot about correct language, but nothing about (possibly) incorrect language. What we are basically asking the system is to recognise incorrect language based on its knowledge of correct lan-
guage. If this appears to be the case, we can say that the system is robust in its task. The results from this thesis, similar to the various papers that have been described here, show quite clearly that this is possible to a certain extent, but there is also a lot of room for improvement, because the upper bound of 75% on completely similar native data is not yet reached.

5.2.3 State of the art

It is very difficult to determine the state-of-the-art for this particular issue, as this problem can be carried out and evaluated in so many different ways because of the following factors:

1. Linguistic features (amount of prepositions to include, different feature levels such as POS and token-levels)

2. Language (which L1 and L2 for L2-learners)

3. Corpora (which learning corpus, what native corpus, preprocessing of ‘noisy’ data)

4. Procedure (included error types, training method, training size, machine learning algorithm)

5. Evaluation (as a pipeline or evaluate components separately)

Because of the fact that I was unable to find any comparable work for Dutch preposition error correction, it is not possible to accurately establish a state-of-the-art comparison right now. The next best thing would be to compare the basic selection and detection results to other papers that work with English as the language of interest, as was shallowly done in the above sections. For example, the selection model achieved an F-score of 75% on preposition selection on native data, possibly improving on Felice and Pulman [2009] who achieved an accuracy of 70% on native data. However, it is not possible to draw such a conclusion because Felice and Pulman [2009] trained their system on nearly 9 million events, whereas my system was trained on 20 million. Consequently, the data set and language were entirely different. Furthermore, Felice and Pulman [2009] included nine preposition labels against fifteen in my model. Felice and Pulman [2009] achieved an accuracy
of 54% on learner data when omitting the difference between correct and incorrect
prepositions (respectively 69% and 39%). Chodorow et al. [2007] did not report
accuracy, instead providing a precision of 67.7%-77.8% depending on the gold
standard annotation set that was used. Chodorow et al. [2007] decided to skip
sentences where erroneous spelling occurred around prepositions which probably
improved the results, particularly because Felice and Pulman [2009] and I did
not do this. Nagata et al. [2014] adopted an entirely different approach, drawing
knowledge primarily from known errors. Because of the many different factors
that one must take into account when comparing these results, the comparisons
in this thesis should not be assumed as conclusive. The lack of an established
baseline further impedes things, making it altogether quite difficult to interpret
the results correctly. Perhaps the best suggestion is to aim towards humanlike
quality as demonstrated by Felice and Pulman [2009], who presented an upper-
bound (review the Relevant Work chapter).

5.3 Leerdercorpus Nederlands as learner corpus

During this research project, I had the privilege to use this learner corpus for
Dutch. A learner corpus must be reliable in the sense that it should accurately
represent the behaviour of a language learner, specifically regarding (in)correctness
of spelling, lexical choice and grammar. The fact that there was a fair amount of
preposition errors is promising. Naturally, there are a lot of other errors that the
system had to cope with which likely impaired the results. A possible next step
would be to manually annotate the corpus so errors could be corrected. This way,
only prepositions could be left untouched while the rest of the corpus would be
error-free, resulting in better performance.

Some examples that make it difficult for the classifier are listed below. In sen-
tences (20) and (21), the system detected insertion errors at the bold-faced words.
While the usage of these prepositions is incorrect, deleting them is not the right
action. Rather, in sentence (20), bij and voorbeeld ought to be bijvoorbeeld (for example). The Alpino parser most likely detected bij as a preposition, but
it factually is not because it is simply the product of a misspelled word. The model
cannot handle this well. Similarly, in sentence (21) (written by an English stu-
dent) Op avond is probably an attempt to produce at night, which is properly
written as ’s Avonds (Des avonds) in Dutch. Again, these are not truly insertion errors but errors of a different nature that resulted in a preposition-related error. Sentence (22) is ideal for this task. The entire sentence is quite coherent and only the bold preposition is wrong (voor is better). Sentence (23) expresses erroneous spelling (en should be een, bestel should be bestelling) around the preposition. The sentence contains bad spelling, but correcting these mistakes around the prepositions is enough: The preposition itself fits.

(30) *Contact met een andere godsdienst ka bij voorbeeld tot bekering brengen.
(31) *Op avond hoor ik Nederland in mijn hoofd als ik Vlaamse of Nederlandse was.
(32) *Ik weet niet waarom, misschien de eerbied van de taal!
(33) *De verteller en zijn broers gaan naar de bakker om en bestel van hun vader te halen.

A small portion of the errors were deletion/insertion errors, totaling up to only 34 (31%). This can be explained well by reviewing table [1,2]. Most prepositions are found in PP’s which logically need to have prepositions as headwords, or as fixed prepositions that accompany a verb or expression. Because PP’s occur so often in language (about 19% of all constituents) where the required presence of a preposition is trivial, it is a lot more likely that people will pick the wrong preposition rather than miss one completely, or pick one in a prohibited context. The use of prepositions is therefore a lot more problematic when it comes to selecting the correct one rather than making the decision of using one at all. The former one is much less rule-bound than the latter.

5.4 Error feedback

Automatic grammatical error correction can be beneficial in multiple fields, such as computer-assisted language learning, artificial writing assistance and general grammar correction. I would like to argue that the demands of an ideal system that is developed to tackle this particular problem heavily depend on the goal. Furthermore, the quality of such a system is measured differently for whatever field in which it was designed. Regarding (computer-assisted) language learning,
the correction phase of a certain learning experience is an important step in the
process of language acquisition. After all, it allows one to learn from his or her
mistakes in order to improve their knowledge of the language. Therefore, it is
essential that the correction phase is done appropriately.

There are multiple philosophies behind the ideal method of providing a learner
with feedback. Correcting a certain error may be sufficient as a fast grammar
check within the domain of editing assistance (where the focus of writing lies not
in language acquisition but the content of the writing, i.e. writing scientific articles
or presentations), but less so as an opportunity to learn from one’s mistakes.
Nagata et al. [2014] argue that there are three distinct ways of providing a learner
with feedback (automatically), also supported by several studies described by Liu
[2008]:

1. Just indicating the correct preposition (high precision is desirable)
2. Indicating the correct preposition with feedback (high precision still desirable)
3. Just providing a feedback message without indicating the correct preposition
   (focus shifts to detection)

The first two ways illustrate the need for a high system precision. A wrong correc-
tion (a false positive or false alarm) without any message is bad for the learning ex-
perience. Providing a message (conform to the second proposed feedback method)
that does not correspond to any grammatical correction will only make the correc-
tion situation worse for the learner, because then one might receive constructive
feedback based on a false alarm. Learners will not assume that the message is
wrong, so they acquire false knowledge. The third suggestion, however, reduces
the need for high precision. Error detection (which is measured in quality by the
recall of the system) becomes a more important measurement. Rather than being
confident about the system’s ability to correct a certain error, it may focus on
detecting the error and providing the language learner with constructive feed-
back that describes exactly why it is an error. The system from this thesis has
a reasonable performance when detecting preposition cases and determining the
type of error, with a lot of improvement regarding insertion and deletion errors.
However, substitution errors are the most commonly found errors. Selection per-
formance was still rather bad at an F-score of only 20% for substitution errors,
but the model would have been efficient at detecting substitution errors which is useful when creating a system that does not focus on absolute correction power, but on language assistance.

Corrections can be given to increase the utility of the system, but studies have shown that this does not (necessarily) increase the learning experience (Liu[2008]). Instead of providing but one correction, multiple suggestions could be given. For example, the five most likely corrections can be presented to the learner. The chance that the correct answer is among them is much bigger than when only one correction is provided.

Shifting focus makes even more sense considering the fact that in various cases, there are multiple suitable prepositions, rather than just one. The gold standard is therefore not necessarily limited to a single value. The model in this thesis regrettably does not support multi-label classification, but this is a good suggestion for the future.

The next chapter summarises my findings and aims to provide suggestions for future work.
Chapter 6

Conclusions

Where can one go from here? What can be learnt from this thesis? This final chapter answers the questions posed at the beginning of this thesis report and explains what I would have done differently or what I would do, given the opportunity and time to continue this work. This way, they may serve as suggestions for other people who aim to investigate something similar.

6.1 Summary

In the Introduction chapter of this thesis, I posed the question *Is it possible to leverage Dutch native data to detect and correct preposition errors?*. Creating a model that accurately predicts Dutch prepositions from native data was the first step and refers to the first subquestion I posed: *Is it possible to build accurately model the use of Dutch prepositions?* With an F-score of 75% based on fifteen classes which is on par with relevant work in other languages, the answer to this question is that it is indeed possible to build a model that predicts Dutch prepositions with fair accuracy. The next section introduces suggestions to improve this model.

Furthermore, I set out to explore the capabilities of the Leerdercorpus Nederlands by adopting its data for this thesis. I gathered a silver standard through an online tool in order to get annotations for the learner corpus, on which I performed a final screening to establish a gold data set of 1,499 items. Out of 1,499 items, 105 errors were found (7%). Some of these items were generated artificially during
the detection of deletion errors. Omitting these leaves us with 971 preposition items and 101 errors (10.4%), of which 72 (71%) were substitution errors and 29 were insertion errors. Only four errors were deletion errors, which raises questions about the immediate relevance of detecting and correcting deletion errors, considering how difficult it is to appropriately model this. It is encouraging to know that there has been a fair amount of errors. As I only annotated a small subset of the corpus, I hypothesise that there are more. This means that the corpus is indeed suitable for the task of error detection and correction.

The results and their respective discussions have shown that it is also possible to apply this preposition model to the detection and correction of preposition errors, since the system detected 70% of all errors in the set, and 90% of all substitution errors, of which 60% were appropriately corrected. The system often triggered false errors, resulting in a fairly low precision on the detection of error types (16.5%). This is because of the fact that a) the system is built as a single-label classifier, so other likely prepositions are omitted, and b) because of the fact that the detection model does not work correctly because it marked correct preposition absences as deletion errors excessively. Even though it is possible, the model is not ready in this state because its accuracy on L2 data is still too low for commercial or educational purposes. These results, however, are encouraging, and there are many ways to improve the system, outlined in the next section.

6.2 Future work

From this work, there are multiple future directions, since the task covers a wide array of fields. From a machine learning point of view, it makes sense to try and improve the ability of the system to detect and correct errors. One obvious suggestion is to include possibly multiple gold standard prepositions, keeping in mind that there might not be one single absolute preposition for a given case. An interesting next step would be to investigate the possibilities of multilabel classification for this task, where the input of a certain vector yields multiple classes. The same issue was raised by Felice and Pulman [2009]. This will likely reduce the number of false alarms (where a language learner is wrongly given the impression that he picked a wrong preposition). Consider that the selection model, given any vector, answers the question 'What preposition is the most likely
selection for these features?' In any case, it only returns one value rather than multiple ones. As there is a distribution skew which affects preposition selection (refer to Table 4.6), a very common preposition might be selected without it being the most fitting one for that context. Consequently, a one-class classification model, for this task, deems any class that is not selected as unsuitable, hence the high degree of false alarms.

Rephrasing the question for a multiclass classification model gives us 'How likely/suitable is the given preposition for this context?' or even 'How unlikely is the given preposition for this context?' If the preposition is the third-most likely selection, is it wrong? What is the likelihood? Posing these questions to a model gives rise to the thought that a model might not (yet) be trusted fully to always make the best decision, because of the thought switch that this is a classification-based approach to a correction-based research question. Furthermore, the results on native data are not always accurate. This follows my suggestion that a model could be used for suggestions rather than absolute corrections with more efficiency, acting as a language assistance application rather than mimicking a language expert.

This can go hand in hand with determining how ambiguous certain preposition contexts are, simultaneously trying to search for new features that further disambiguate these contexts. The suggested need for, and use of multiple possibly correct answers is only necessary because certain contexts do not sufficiently cancel out all but one preposition. In addition, it would be interesting to see if semantic information would contribute to the performance of the system, which was already done by Felice and Pulman 2009.

The concept of detecting relevant preposition absence was introduced here. As mentioned before, I do not know of any group that has tried this in the same way. In this thesis, there is a lot of room for improvement, because the model clearly overclassifies preposition presence over absence, which makes little sense linguistically because people do not generally forget prepositions, as there was a negligibly small amount of deletion errors. It would be very interesting to explore more ways of improving deletion error detection, but at the same time it is debatable how much this would improve overall results.

Regarding language portability, it is interesting to see whether the features provided by Chodorow et al. 2007 or Felice and Pulman 2009 work for different
languages as well. This work has shown that the features work quite adequately for Dutch, whereas they have been taken from a paper which featured English as the main language. English and Dutch are very similar in nature, compared to other language pairs such as English and Japanese, Hungarian or Russian. The ingredients to profile preposition usage and absence are essential when tackling this issue, and thus it makes sense to experiment with feature sets across different languages.

Furthermore, the Leerdercorpus Nederlands (Dutch Learning Corpus) deserves more attention. The fact that such a dataset is available for Dutch is a fantastic development. Preposition errors are but one of the many types of errors that can be identified. Error that are commonly found in the corpus include determiner errors, verb conjugation errors, spelling errors, and more. These types of errors can be investigated as well. In this work, I only investigated 1,499 events, which means that I only had access to a small subset of the total amount of errors in the corpus. More reliable results can be drawn from bigger samples so all error types and preposition types will have better coverage.

6.3 Final statement

The field of (preposition) error correction is one with a plethora of problems and possibilities. I hope that other languages rather than just English will receive more attention in the future and that this thesis might act as a stepping stone for further research into (Dutch) error detection and correction.


of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 754–764, June 2014.


