Developing AI Tools For A Writing Assistant: Automatic Detection of dt-mistakes In Dutch

Wouter Mercelis *wouter.mercelis@kuleuven.be*
Faculteit Letteren/Onderzoekseenheid Taalkunde/Onderzoeksgroep
Kwantitatieve Lexicologie en Variatielinguïstiek (QLVL)
KU Leuven, Leuven, 3000, Belgium

Abstract

This paper describes a lightweight, scalable model that predicts whether a Dutch verb ends in -d, -t or -dt. The confusion of these three endings is a common Dutch spelling mistake. If the predicted ending is different from the ending as written by the author, the system will signal the dt-mistake. This paper explores various data sources to use in this classification task, such as the Europarl Corpus, the Dutch Parallel Corpus and a Dutch Wikipedia corpus. Different architectures are tested for the model training, focused on a transfer learning approach with ULMFiT. The trained model can predict the right ending with 99.4% accuracy, and this result is comparable to the current state-of-the-art performance. Adjustments to the training data and the use of other part-of-speech taggers may further improve this performance. As discussed in this paper, the main advantages of the approach are the short training time and the potential to use the same technique with other disambiguation tasks in Dutch or in other languages.

Keywords: NLP, Dutch, AI, Spelling Correction, Transfer Learning.

1 INTRODUCTION

1.1 Goal

This paper describes a machine learning approach to predicting whether a Dutch verb ends in *-d*, *-t* or *-dt*. The first section provides background information about this topic. The following sections give an overview of the used data sets and the different models that were used during training. The last section discusses strong and weak points of this paper's approach.[1] The methodology is deductive, as I started from a theoretic point of view (fast and lightweight neural networks) and brought this into practice. Real mistakes from students were used for analysis purposes.

The paper can be linked to other entries in the International Journal of Computational Linguistics, regarding spell checkers [1], verb analysis [2] and stemming algorithms [3].

1.2 Grammatical Error Detection

Most of the work in the NLP field regarding grammatical error detection focuses on general systems and on determining whether or not a sentence is grammatical. Examples include Liu & Liu [4] and Li et al. [5]. Rather than designing an all-purpose grammatical error detection system, this project aims to address one specific problem. An additional complicating factor is that most of the work undertaken is tailored to the specifics of English. The few studies on Dutch do not adopt the general approach; instead, they focus on one specific problem, similar to this article. Heyman

[1] This project was undertaken against the backdrop of an internship at Reimagine, a Brussels-based AI start-up. Because of this, the code used during the project cannot be made available, being property of the company. The test data was provided by ILT (Instituut voor Levende Talen), which commissioned the project as future part of a writing assistant for secondary school students. . I would like to thank my colleagues at Reimagine, especially Ferre Jacobs and Toon Lybaert, who provided useful insights for this article. I would also like to thank Alek Keersmaekers, Toon Van Hal and Liesbeth Augustinus for their proofreading and valuable guidance.

et al. [6] also worked on dt-mistakes, while Allein et al. [7] provided a model that disambiguates *die* and *dat*. Dt-mistakes will be discussed in more detail in the next section.

The first grammatical error detection systems built were rule-based. However, writing such manual rules is very time-consuming, does not generalise well and does not capture more complex phenomena like long-distance dependencies [8]. Up until recently, a statistical approach with n-grams in large corpora was used to solve this issue. Data sparsity forces the n-gram approach to work with large data sets such as the Google n-gram corpus [9]. These models need to run on expensive high processing computing systems due to their large size. Additionally, the n-gram approach still has problems capturing more complex phenomena like long-distance dependencies [8]. Heyman et al. [6] referred to an approach in which a word-based classifier is trained per word pair that can give way to a dt-mistake (e.g. *gebeurd* and *gebeurt*). However, this approach does not generalise well, as the system does not learn the actual dt-rule.

Nowadays, neural network models are considered to be better equipped to perform such error detection tasks. These models have the capability to model complex sentences containing, for example, many long-term dependencies [10].

1.3 The Dutch dt-rule

Dutch regular verbs adhere to the following conjugation rules, as illustrated in Table 1. In this table, the first six rules are considered dt-rules in a narrow sense. Rules 1-11 can be seen as the dt-rules in a broader sense, as they involve endings other than just *-d*, *-t* and *-dt*.

#	tense	usage	subj. position	rule	example + translation
1	present	1^{st} person	anywhere	stem	Ik **beantwoord** je vraag. I **answer** your question.
2	present	2^{nd} person	after the verb	stem	**Beantwoord** je de vraag? Do you **answer** the question?
3	past participle	as verb	anywhere	(ge) + stem + (d/t)†	Hij heeft de vraag **beantwoord**. He has **answered** the question.
4	imperative	/	no subject	stem	**Beantwoord** de vraag! **Answer** the question!
5	present	2^{nd} person	not after the verb	stem + t	Jij **beantwoordt** de vraag. You **answer** the question.
6	present	3^{rd} person	anywhere	stem + t	Hij **beantwoordt** je vraag. He **answers** your question.
7	past participle	adjective	anywhere	(ge) + stem + (d/t)†+ (e)	De **beantwoorde** vraag ... The **answered** question ...
8	past	singular	anywhere	stem + te/de	Hij **beantwoordde** de vraag. He **answered** the question.
9	past	plural	anywhere	stem + ten/den	Zij **beantwoordden** de vraag. They **answered** the question.
10	present	plural	anywhere	stem + en	Zij **beantwoorden** de vraag. They **answer** the question.
11	infinitive	/	anywhere	stem + en	Ik zal de vraag **beantwoorden**. I will **answer** the question.

TABLE 1: An overview of the Dutch dt-rules, cited from Heyman et al. [6].

In Dutch, *-d* at the end of a word is pronounced as unvoiced *-t*, which explains why there is no audible distinction between words such as *rat* ('rat') and *rad* ('wheel'). When the stem of the verb ends in *-d*, confusion arises. The first person singular ends in *-d*, e.g. *ik word* ('I become'), but in the second and third person singular present, the ending *-t* is added to the stem, e.g. *hij wordt*

('he becomes')[2]. The *-dt* at the end of a word, which occurs when a stem ending in *-d* receives the verb ending *-t*, also has an unvoiced pronunciation. Verhaert & Sandra [11] argued that the root cause of dt-mistakes is so-called homophone dominance. This means that the writer will choose the most frequent form of a verb, when put under stress or when distracted.

Another cause underlying many dt-mistakes is the inversion rule for second person singular. Normally, this point of view has an ending in *-t*, but when the sentence is inversed (e.g. in a question), there is no such ending. This can be illustrated with examples 1 and 2 [12]:

1. *Jij beantwoordt haar vraag.*
   You answer her question.'
2. *Beantwoord je haar vraag?*
   'Do you answer her question?'
3. *Beantwoordt je moeder haar vraag?*
   'Does your mother answer her question?'

Furthermore, the second person singular personal pronoun *jij* also has the phonetically weakened form *je*. However, the second person singular possessive pronoun *jouw* has the same phonetically weakened form *je*, e.g. *jouw/je moeder* ('your mother'). In non-inverted sentences, this does not cause any problems, as the endings for the second and third person singular are the same. However, in inverted sentences such as questions, the ending differs. This is illustrated in examples 2 and 3 [12].

A third cause of dt-mistakes lies in the past participle, which is regularly formed by adding a prefix *'ge-'* to the verb stem and a suffix. The suffix is *-t* when the verb stem ends in an unvoiced consonant, but it is *-d* when the verb stem ends in a voiced consonant. For example, the verb *suizen*, meaning 'to whiz', does not have the predicted stem *'suiz'*; rather, it has the stem *'suis'*. The underlying stem *'suiz'* is used to determine the ending, resulting in the participle *gesuisd* and not *'gesuist'* [12].

Another category of participle-based dt-mistakes occurs when the past participle is irregularly formed. For example, when a verb stem already starts with *'ge-'*, the past participle prefix *'ge-'* is dropped. When the verb stem ends in a voiced consonant, the participle ends in *-d*, while the second and third person singular present forms end in *–t* [12].

4. *Hij getuigt tegen mij.*
   'He testifies against me.'
5. *Hij heeft tegen mij getuigd.*
   'He has testified against me.'

Finally, the difference between context-dependent and context-independent mistakes needs to be stressed. All previous examples are prone to context-dependent mistakes, as the different forms of the verb all exist in Dutch but in different contexts. However, it is also possible to make a context-independent mistake and create a non-existent form. An example from the test data is *hij duwd*, a non-existent form, instead of *hij duwt* ('he pushes').

1.4 Previous Work by Heyman et al. [6]

Heyman et al. [6] focused on context dependent dt-mistakes, while this paper aims to cover both context dependent and context independent errors. While it is theoretically possible to use a dictionary lookup for these context independent mistakes, it is convenient that the model is able to predict these as well, as this reduces the amount of memory needed.

Another difference is that Heyman et al. created a system to introduce mistakes in a text to later correct them. In this paper, for the training data, we have used data without mistakes, as the

[2] Phonetically, both forms can be transcribed as follows: /ʊɔrt/

model only trains on predicting the right ending. For the test data, we did not insert mistakes in a text with an algorithm. Instead, I searched for texts containing actual mistakes and focused on sentences written by students. If the model's prediction was different from a student's written verb ending, a signal was given that the form may be incorrect.

The architecture of the models also differs. This paper makes use of the ULMFiT approach with an AWD-LSTM network, which is a regular LSTM network with tweaked hyperparameters. Heyman et al. used a custom engineered setup, consisting of a context encoder and a verb encoder, concatenated in a feed-forward neural network (FFNN). The final probability is computed using a softmax function over the FFNN-layer. A visualisation of Heyman et al.'s model is shown in Figure 1. Another difference between the two approaches lies in the use of part-of-speech taggers. Heyman et al. used TreeTagger [13], while I used spaCy's PoS tagger [14]. This decision was primarily made because the commissioning institute (ILT) already works with spaCy, thus making the final implementation easier. It is possible that spaCy's PoS tagger missed some verbs, which are thus not present in the data. In the meantime, the model aimed to accurately predict the ending of the detected verbs. In this implementation, a verb was considered correct if the written ending was the same as the model's predicted ending. Otherwise, the verb was considered to be incorrect.

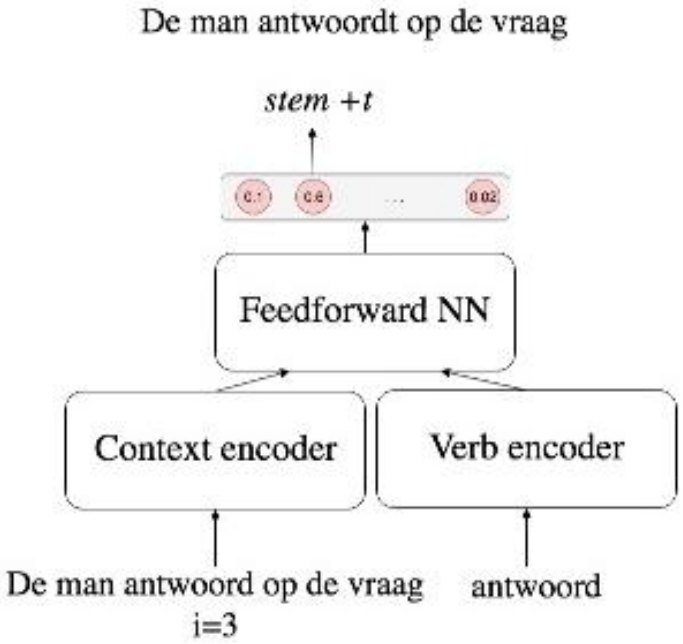

FIGURE 1: A visualisation of the model used by Heyman et al. [6].

As mentioned above, there is a difference between dt-mistakes in the narrow sense (confusing *-d*, *-t* and *-dt*), and dt-mistakes in the broad sense (adding *-de*, *-dde(n)*, *-te*, *-tte(n)* etc.). Heyman et al. [6], working on the broad sense, experienced some trouble with the multitude of labels, causing them to drop infrequent labels such as the past plural forms (number 9 in Table 1). As mistakes in the narrow sense are in absolute numbers much more frequent than mistakes in the broad sense, we decided to focus on the core task of discerning *-d*, *-t* and *-dt*. Furthermore, the broad sense is harder to grasp completely, as some minimal pairs show the doubling of vowels as well (e.g. *vergrote*, *vergrootte* ('enlarged')). However, it should be possible to train another, separate model that corrects all possible mistakes in the broad sense. We leave that problem for further research.

2 DATA

2.1 Training Data

Three main data sets were used for training: the Europarl corpus, the Dutch Parallel Corpus (DPC) and a corpus made from Dutch Wikipedia articles. The development data used during the training of the models stemmed from the random selection of 20% of the sentences in the training data. A detailed description of the data is included in the appendix in Table 5.

2.1.1 Europarl Corpus

The Europarl corpus [15] collects the proceedings of the European Parliament and is available in several languages, including Dutch. Professionals created this corpus, which means that the quality of the text data is assured. However, the domain variation is limited, both in content, as only politics are covered, and on a grammatical level, as there are few verbs found in the second person. To filter out a first wave of sentences with no verbs ending in *-d*, *-t* or *-dt* at all, I used a subset of Heyman et al.'s [6] Europarl corpus in which each sentence contains at least one verb ending subject to the Dutch dt-rule in its broad sense.

I altered the data set by evening out the counts of *-d*, *-t* and *-dt*. As *-dt* occurs less than *-d* and *-t* (between 5% and 10% of the cases, depending on the data set), it was difficult for the model to predict the dt-cases. This data alteration was also performed on the other two corpora, described infra.

2.1.2 Dutch Parallel Corpus

The Dutch Parallel Corpus (DPC) [16] is a multilingual parallel corpus. It consists of parallel texts in Dutch, English and French. The main goal of the corpus is to provide material for multilingual tasks such as machine translation, but it is also possible to use a corpus containing only one language. An advantage of the DPC is that it covers a wide range of domains, although the total size of the corpus is significantly smaller than the Europarl corpus.

2.1.3 Dutch Wikipedia Corpus

As using a complete Dutch Wikipedia dump [17] would result in too much data, I selected articles at random to create a subset of the entire corpus. The advantage of the Wikipedia corpus is that all texts are written in a (pseudo-)scientific style. As the goal of the study was to improve a tool correcting student-written texts in a similar style, this data set closely fit the expectations. A disadvantage of the Wikipedia corpus is that all the sentences are written in third person, while students use the first person when writing about their personal experiences.

2.1.4 Combinations of Data Sets

To circumvent problems such as the absence of first person pronouns in the Wikipedia corpus, I decided to combine data sets. The Wikipedia corpus was used as the base corpus, with the addition of either the DPC or the Europarl corpus. Using the equalised version ensured that the resulting corpus was small enough to be computationally efficient. Moreover, it guaranteed that there were enough dt-cases, while still maintaining the difference in usage between *-d*, *-t* and *-dt*.

In theory, combining all three data sets would also be possible. As the training time would further increase, and the current results turned out to be satisfactory, I did not train a model combining all three data sets.

2.2 Test Data

Over the course of the study, I added several sets of test data, taken from various sources. Some data sets were added to compare results with other research, while others were added to approximate real world data. The data sets are summarised in Table 2.

2.2.1 Test Corpus from Heyman et al. [6]

The test set containing online dt-quizzes was made publicly available [18]. For this paper, sentences not relating to dt-mistakes in the narrow sense were removed. This makes an accurate

comparison with Heyman et al. difficult, as I do not have access to their results on this particular subset.

2.2.2 Scholieren.com

Scholieren.com is a website where students (mainly from secondary school) can upload their writings (book reports, essays etc.). Other users can rate the uploaded writings, but the writings are not checked for spelling mistakes or factual errors. The writings uploaded to this website are thus highly approximate to the target group of secondary school students. Four texts were chosen, written by Flemish students of various ages, viz. papers written by a first year secondary school student [19], a second year secondary school student [20], a third year secondary school student [21] and a sixth year secondary school student [22], respectively.

From these texts, 100 cases were randomly selected. This is data set Schol_1 in Table 2. Afterwards, I searched for a text with a high number of mistakes. This was done to check whether spaCy's part-of-speech tagger was still able to identify verbs when they were spelled incorrectly. A book review, written by a sixth year secondary school student, meets this criterion [23]. This is data set Schol_2 in Table 2.

2.2.3 Test Set ILT

ILT provided me with their own test set containing actual errors made by students. From this list of 100 sentences, 69 sentences were suitable for d/t/dt-classification. This data set diverges from the pattern seen in the training data, where the ending -dt occurred between 5% and 10% of the cases. In this data set, it is the most frequent ending. A possible explanation is the focus in this data set on the auxiliary verb *worden*, which occurs 28 times. This verb ends in the first person on -d and in the more frequent third person on -dt.

Name	Verb Endings			Total
	d	t	dt	
Heyman	21	16	9	46
Schol_1	45	49	6	100
Schol_2	8	21	4	33
ILT	17	25	27	69
Total	91	111	46	248

TABLE 2: Overview of the used test data sets.

3 MODELS

This section first discusses the preprocessing steps and the main architecture used for the language model. This is followed by an overview of the trained models, after which attention is given to metrics other than accuracy.

3.1 Preprocessing Steps

After downloading the data, the first step consisted of filtering the actual sentences and removing other unnecessary information. For the Wikipedia corpus, this meant deleting titles and subtitles. The DPC, on the other hand, is XML-based. Therefore, the tags allowed me to extract the text and write it to one large text file. For the Europarl corpus, I used the sentence-by-sentence text file from Heyman et al. [6]. As a second step, spaCy's part-of-speech tagger was used for each set of sentences, enabling me to identify the verbs in each sentence. For each verb, the ending (*-d*, *-t* or *-dt*) was cut off and replaced with the dummy symbol #. The actual verb endings were kept to serve as a ground truth for the classifier and to determine whether a mistake was made in the test data. If the verb did not end in *-d*, *-t* or *-dt*, the dummy symbol was still added, and the ending was filled with a slash sign, such as 'is/'.

As there is often more than one verb in a sentence, I decided to repeat sentences in the data set. For each repetition, only one verb was assigned a dummy symbol. The sentence *Ik word opgehaald door mijn moeder,* meaning 'I get picked up by my mother', appears in the data set as:

- Ik wor# opgehaald door mijn moeder.
- Ik word opgehaal# door mijn moeder.

This way, the classifier does not have to predict multiple verbs at the same time, which would be impossible.

Using these steps, the text file was converted to a comma-separated values file (CSV file), with the following columns: sentence ID number, unaltered sentence, sentence with a dummy symbol, verb without an ending and verb ending. The verb without ending column was added to improve the readability of the CSV file. The preprocessing is visualised in Figure 2.

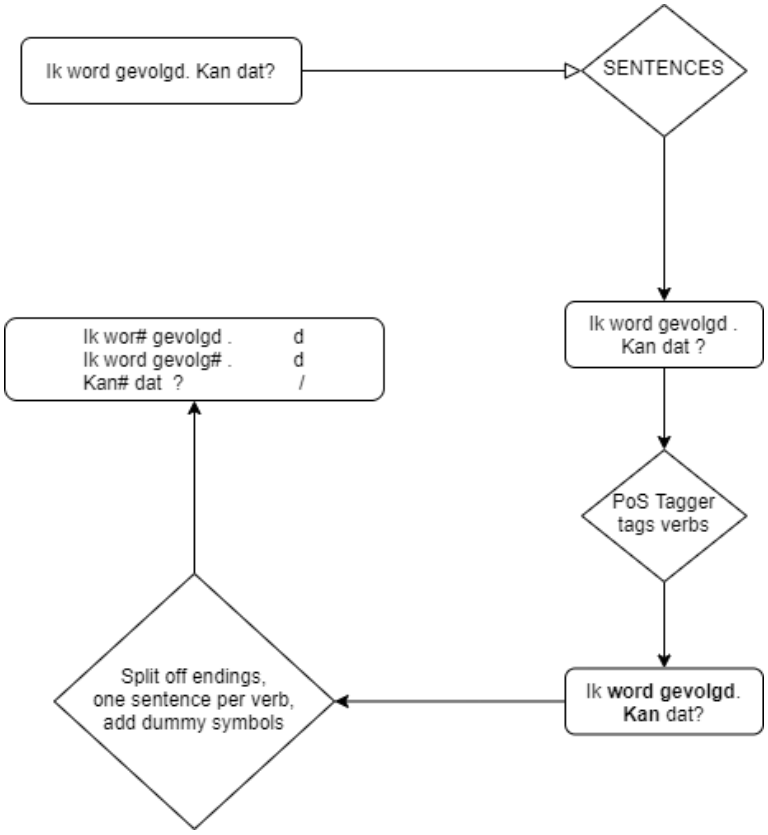

FIGURE 2: A schema highlighting the preprocessing steps.

3.2 Transfer Learning with ULMFiT

3.2.1 ULMFiT

ULMFiT (universal language model fine-tuning for text classification) is a method to introduce inductive transfer learning to various natural language processing tasks. Originally, the technique has found success in Computer Vision, but up until recently this was not the case in NLP due to a lack of knowledge [24]. The model makes use of a three-layer LSTM architecture called AWD-LSTM. This architecture is the same as a regular LSTM (long short-term memory), but uses different dropout hyperparameters. Using an LSTM makes sure that long-term dependencies are covered, as an LSTM is able to filter out important information that spans an entire sentence.

The process behind an ULMFiT model can be divided into three steps. The first step is the acquisition of a general language model. This model can be trained, or an existing model can be used. For this paper, an existing model was used [25], although we briefly considered training our own model. However, the existing model formed a good starting point because it was trained on a

large Wikipedia corpus. Nevertheless, fine-tuning these data with our own, smaller Wikipedia corpus increased the performance of the model.

The second step is fine-tuning the general language model on the target task. This takes into account that the training data often have a different distribution than the data used in the general model [24]. It is possible to use discriminative fine-tuning while training the fine-tuned model. This means that different layers of the model are trained with different learning rates, which improves the performance of the model. The vocabulary size of the language model is 60,000 words. The loss function is a standard cross-entropy loss function, which is used for the classifier training as well. This function can be described as follows [26]:

$$loss(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log\left(\sum_j \exp(x[j])\right)$$

The input of this function is a vector with three elements during the training of the classifier, one for each possible class. The predicted class is positive, while the other two are negative. The displayed result of the loss function is the average loss per epoch. A lower loss means that there is less distance between the predicted classes and the ground truth.

The third and final step involves fine-tuning the classifier. For this task, two linear layers are added to the language model, with a ReLU activation for the first layer and a softmax activation for the second layer. This way, the classifier gives a probability distribution over the classes as output. To fine-tune the classifier, it is possible to use gradual unfreezing. This means that the layers of the model are fine-tuned one by one instead of all layers being fine-tuned at the same time. This process starts at the last layer, as this layer contains the most specific knowledge. By using this technique, the risk of catastrophic forgetting is reduced [24]. A visualisation of the ULMFiT architecture is shown in Figure 3.

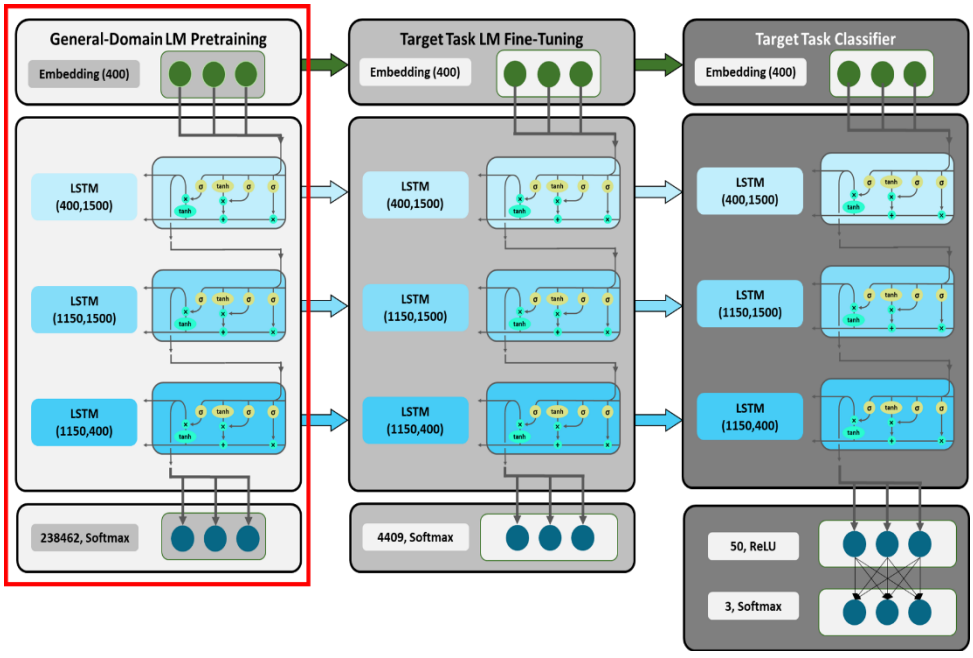

FIGURE 3: A visualisation of the ULMFiT architecture [27].

3.2.2 Fast.ai

Fast.ai is an AI library for Python, built on PyTorch, that aims to make AI more accessible to the general public [28]. As fast.ai v2 [29] was still in development during this study, I used fast.ai v1. The library is built in such a way that both people with limited coding skills and experts are able to

work with it. As the developers of fast.ai also developed ULMFiT, the platform is best suited to use this architecture. However, other techniques can be used as well, such as BERT-based models[3].

3.3 Overview of the Trained Models

For this paper, I experimented with various models, whose results will be explained in more detail here, with special attention given to the final model.

The experiments showed that using fast.ai's gradual unfreezing technique provides good results. It predicted the right learning rate per epoch, as shown in the below graph (Figure 4).

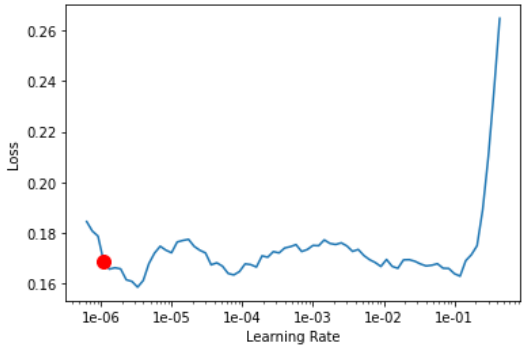

FIGURE 4: The learning rate predictor during model training. The red dot is the suggested learning rate.

Equalising the verbs, as described in section 2.1, resulted in a negative impact. Few training data remained, resulting in a performance hit, especially in the test data.

The final experiments involved the equalised data sets, adding the DPC equalised set to the well-performing Wikipedia corpus in one experiment, and adding the Europarl equalised set to the Wikipedia corpus in another. The combination Wikipedia-Europarl outperformed Wikipedia-DPC by a slight margin, possibly because the Europarl data set was larger. For an unknown reason, the Wikipedia-DPC corpus performs badly on the test data, while the Wikipedia-Europarl combination notes good results. By combining data sets in this way, I found the perfect balance between an unequalised and an equalised data set, combining advantages from both.

The last described model (Wikipedia + equalised Europarl) is my final model, of which the results are highlighted in Table 3. The model was trained in one work day (excluding preprocessing time as this was already done) and is thus easily extendable to other, similar problems. The process of this internship clearly demonstrates that it is important to use data from a domain comparable to the target domain.

The accuracy of the model can still be improved by adding more data containing inverted sentences in the second person singular present. It is not of great importance for the current task, as students are not expected to use this form often in their writings, but it is important if the tool should ever be released for general purposes. Future research can provide a part-of-speech tagger that achieves an even higher accuracy than spaCy's.

3.4 Other Metrics

The main task of this study consisted of training an accurate model. However, there are other metrics that measure whether a model's corrections are right or not, such as precision, recall and

[3] Training BERT-like models did not provide a useful outcome, mainly due to memory issues and the lack of a dummy symbol in the pretrained model.

the F1-score. These metrics are used in the same way as in Heyman et al. [6]. In this section, I will compute these metrics where possible, using the final model on the various test data sets.

As Heyman et al. used different data sets with a classifier that was trained on a slightly different task, it is not possible to make an accurate comparison between their approach and the approach of this paper.

The predicted labels (the verb endings) are divided into four groups:

- True positives (*tp*): the predicted ending corrects the original wrong ending successfully
- False positives (*fp*): the original ending was correct; the predicted ending introduces a mistake
- True negatives (*tn*): both the original and the predicted ending are correct
- False negatives (*fn*): both the original and the predicted ending are incorrect

The precision, recall and F1-score are then defined as follows:

$$prec = \frac{tp}{tp + fp}$$

$$rec = \frac{tp}{tp + fn}$$

$$F_1 = \frac{2prec * rec}{prec + rec}$$

I was not able to measure these metrics for the Heyman data set because we only had the data set with dummy symbols, without actual answers. A different problem was found in the Schol_1 data set, where all the verbs were both correctly spelled and predicted. This way, all the labels belong to the group with true negatives, resulting in a division by zero when calculating precision and recall.

For the Schol_2 data set, calculating these metrics was trivial, as there were mistakes in the original labels and the accuracy of the predictions was 100%. This means that my model achieves a precision, recall and F1-score of 1.

The ILT data set is more interesting for these metrics. The data contains 64 corrected errors, 2 untouched originally correct verbs and 3 mistakes that were not corrected. There were no false positives. This results in a precision of 1, a recall of 0.9552 and an F1-score of 0.9771.

The results using this metrics are listed in Table 4. One should keep in mind that these metrics only consider the verbs in the data set. This means that erroneously written verbs that spaCy's part-of-speech tagger did not consider verbs are not included. However, the tagger still occasionally considered non-existent forms such as *hij duwd* instead of *hij duwt* as verbs. Nevertheless, these wrongly spelled verbs may still form an area of potential improvement.

	Schol_1	**Schol_2**	**ILT**
tp	0	3	64
fp	0	0	0
tn	100	30	2
fn	0	0	3
Precision	/	1	1
Recall	/	1	0.9552
F1	/	1	0.9771

TABLE 4: An overview of the precision, recall and F1-score of the final model, tested on various test data sets. Heyman et al. are not included, as their data consisted of sentences in which the ending should be predicted. Their data did not include writers' filled-in endings with mistakes.

4 DISCUSSION

4.1 General

While the model should be able to generalise well, it has a few disadvantages. For example, the model clearly has trouble predicting the second person singular in inversed sentences. This is due to a lack of training data. As this type of sentence is rather rare in student writings from secondary schools, I chose, due to time constraints, to keep the model as it stands. Needless to say, it is possible to search for more inversed sentences and add them to the training data.

Another possible flaw is the reliance on spaCy's part-of-speech tagger. I observed a case in the test data where a name (*De Gucht*), was incorrectly tagged as a verb, masking the ending *-t*. This can obviously lead to unexpected results. A similar case occurred in the training data, where a sentence contained an enumeration of points, explicitly written as point a until point d. The tagger identified point d as a verb, replacing the ending (the same d) with a dummy symbol. The rest of the verb thus remained empty, causing an unexpected error. Although such mistakes can happen during training, the neural model can handle some noise if the size of the data set is sufficiently large. However, if these mistakes happen when the model is actually in use, the user may not get the expected result. This may occur, for example, if someone uses the name *De Gucht* and the model determines that this is a verb that should end in *–dt*,

It is possible to visualise which words the model deems the most important in order to produce the ending output, thanks to the attention mechanisms. Figure 5 shows the attention the model gives to the sentence *Vind je het een slim besluit van het kabinet dit soort subsidies af te schaffen?* meaning 'Do you think it is a smart decision from the cabinet to abolish subsidies of this kind?' It is clear in the picture that the model gives most of its attention to the verb (without the ending) *'vin'* and to the subject *'je'*. These are indeed the two important factors that determine the verb's ending. In this case, the model correctly predicts the ending *-d*.

FIGURE 5: An example of attention: the two important words are highlighted.

The trained model makes use of a dummy symbol to predict verb endings. This means that the model is easy to generalise to other cases and that it can be trained for other classification tasks as well. For example, the diphthong /ɛɪ/ is written in two different ways in Dutch: ei and ij. While a dictionary-based model can solve most of these, as only one of the two possibilities exists, a neural network-based model can be trained to solve the cases where both words exist, based on context. Examples of minimal pairs for this problem are the verbs *leiden* ('to lead') and *lijden* ('to suffer') and the nouns *peil* ('level') and *pijl* ('arrow'). Instead of replacing the verb endings with a dummy symbol, these diphthongs should be replaced in the training data.

Another problem, already studied by Allein et al. [7], is the disambiguation of the Dutch words *die* and *dat*, which was explained in section 1.2. Further research is needed to determine if the model proposed in this report more effectively addresses these specific problems compared to the current state-of-the-art models.

Finally, multi-label classification, such as predicting punctuation, should also be possible. Due to the higher number of labels, this is a harder task, but the setup used during this study can be used here as well.

4.2 Impact and Possible Comparisons with Heyman et al. [6]

The approach in this article differs from the one used by Heyman et al. [6] at several points. Firstly, the structure of the neural network is different, as Heyman et al. [6] used a recurrent neural network, whereas I made use of an LSTM-powered feed forward neural network.

Secondly, this article does not cover the same set of spelling mistakes as Heyman et al. [6], as they tried to detect dt-mistakes in the broad sense, while this article focuses on dt-mistakes in the

narrow sense. Additionally, this paper pays attention to mistakes that result in non-existing verb forms, whereas Heyman et al. [6] only detect so-called context-dependent mistakes.

Furthermore, there was no need to automatically introduce mistakes to create an evaluation corpus, as this study had access to student-written texts with real mistakes.

Finally, the model presented in this paper is easily scalable, and does not need many computational resources, as it was able to be trained in less than a day on a Google Colab server.

5 SUMMARY

This paper focused on creating a model for detecting dt-mistakes in Dutch sentences. Sentences were preprocessed in that SpaCy's part-of-speech tagger identified the verbs. One verb per sentence received a masked ending (the dummy symbol #). A neural network-based model, based on the ULMFiT transfer learning technique, was trained to classify the sentences as *-d*, *-t* or *-dt*, providing the verb ending. If the ending did not match the ending written by the author of the sentence, the system signaled that a dt-mistake was made. The final model needed one day of training on Google Colab's servers, with a training time of around 20 minutes per epoch, which is a major benefit. This model achieved high scores on various test data sets, comparable with the results of Heyman et al. [6], although it is difficult to accurately compare the two approaches.

In the future, more inverted sentences (mainly questions in the second person singular) could be added to the training data to improve performance predicting the second person singular of verbs in inverted sentences. Furthermore, the reasons causing the failure of the BERT model for this task can be researched. Finally, part-of-speech taggers other than spaCy may further improve the current model.

If the preprocessing is adjusted to a certain extent, the model should be usable, as a practical implication, for a wide range of other correction tasks as well, such as the *die/dat* problem (mentioned above) or the difference between the homophonous diphthongs *ei* and *ij*.

The model is thus of great interest for education purposes, as it can form part of a more extensive spellchecker.

6 APPENDIX: DATA OVERVIEW

Data set		Verb Endings				
Name	**Subset**	**d**	**t**	**dt**	**/**	**Total**
EUR_100K	train	34,088	61,332	11,809	253,324	360,553
	valid	8,704	15,394	3,029	63,011	90,138
	total	42,792	76,726	14,838	316,335	450,691
EUR_100K_NO_/	train	34,088	61,332	11,809	0	107,229
	valid	8,704	15,394	3,029	0	27,127
	total	42,792	76,726	14,838	0	134,356
EUR_100K_EQUAL	train	11,809	11,809	11,809	0	35,427
	valid	3,029	3,029	3,029	0	9,087
	total	14,838	14,838	14,838	0	44,514
EUR_FULL	train	512,756	925,976	178,211	3,837,637	5,454,580
	valid	128,061	231,347	44,442	959,794	1,363,644
	total	640,817	1,157,323	222,653	4,797,431	6,818,224
EUR_FULL_NO_/	train	512,756	925,976	178,211	0	1,616,943
	valid	128,061	231,347	44,442	0	403,850
	total	640,817	1,157,323	222,653	0	2,020,793
EUR_FULL_EQUAL	train	178,211	178,211	178,211	0	534,633
	valid	44,442	44,442	44,442	0	133,326
	total	222,653	222,653	222,653	0	667,959

DPC_FULL	train	116,069	201,673	37,318	706,500	1,061,560
	valid	28,519	50,281	9,332	177,257	265,389
	total	144,588	251,954	46,650	883,757	1,326,949
DPC_FULL_NO_/	train	116,069	201,673	37,318	0	355,060
	valid	28,519	50,281	9,332	0	88,132
	total	144,588	251,954	46,650	0	443,192
DPC_FULL_EQUAL	train	37,232	37,232	37,232	0	111,696
	valid	9,418	9,418	9,418	0	28,254
	total	46,650	46,650	46,650	0	139,950
WIKI	train	132,530	113,211	21,581	501,889	769,211
	valid	32,878	28,537	5,351	125,536	192,302
	total	165,408	141,748	26,932	627,425	961,513
WIKI_NO_/	train	132,530	113,211	21,581	0	267,322
	valid	32,878	28,537	5,351	0	66,766
	total	165,408	141,748	26,932	0	334,088
WIKI_EQUAL	train	21,581	21,581	21,581	0	64,743
	valid	5,351	5,351	5,351	0	16,053
	total	26,932	26,932	26,932	0	80,796
WIKI_NO_/	train	169,762	150,433	58,813	0	379,018
+ DPC_FULL_EQ.	valid	42,296	37,955	14,769	0	95,020
	total	212,058	188,398	73,582	0	474,038
WIKI_NO_/	train	310,741	291,422	199,792	0	801,955
+ EURO_FULL_EQ.	valid	77,320	72,979	49,793	0	200,092
	total	388,061	364,401	249,585	0	1,002,047

TABLE 5: Overview of the used training data sets. 'EQUAL' denotes that the amounts of *-d*, *-t* are *-dt* are made equal, 'NO_/' indicates the removal of irrelevant verb endings, which were displayed with a / sign.

7 REFERENCES

[1] L. Salifou, and H. Â Naroua. (2014, Jun.). “Design of A Spell Corrector For Hausa Language.” *International Journal of Computational Linguistics*. [On-line]. 5(2), pp. 14-26. Available: https://www.cscjournals.org/library/manuscriptinfo.php?mc=IJCL-56 [May 5, 2021].

[2] G. Alafang Malema, N. Motlogelwa, B. Okgetheng and O. Mogotlhwane. (2016, Aug.). “Setswana Verb Analyzer and Generator.” *International Journal of Computational Linguistics*. [On-line]. 7(1), pp. 1-11. Available: https://www.cscjournals.org/library/manuscriptinfo.php?mc=IJCL-73 [May 5, 2021].

[3] J.S. Sumamo, and S. Teferra. (2018, Oct.). “Designing A Rule Based Stemming Algorithm for Kambaata Language Text.” *International Journal of Computational Linguistics*. [On-line]. 9(2), pp. 41-54. Available: https://www.cscjournals.org/library/manuscriptinfo.php?mc=IJCL-73 [May 5, 2021].

[4] Z. Liu and Y. Liu. (2016). “Exploiting Unlabeled Data for Neural Grammatical Error Detection.” *arXiv.org*. [On-line]. Available: http://search.proquest.com/docview/2080422559/ [Mar. 21, 2021].

[5] Y. Li, A. Anastasopoulos, and A. W. Black. (2020, Jan.). “Towards Minimal Supervision BERT-based Grammar Error Correction.” *ArXiv200103521*. [On-line]. Available: http://arxiv.org/abs/2001.03521 [Mar. 21, 2021].

[6] G. Heyman, I. Vulić, Y. Laevaert, and M.-F. Moens. (2018, Dec.). “Automatic detection and correction of context-dependent dt-mistakes using neural networks.” *Comput. Linguist. Neth. J.* [On-line]. 8, pp. 49–65. Available: https://clinjournal.org/clinj/article/view/79 [Mar. 21, 2021].

[7] L. Allein, A. Leeuwenberg, and M.-F. Moens. (2020). "Binary and Multitask Classification Model for Dutch Anaphora Resolution: Die/Dat Prediction." *ArXiv*. [On-line]. Available: https://arxiv.org/abs/2001.02943 [Mar. 21, 2021].

[8] C. Leacock, M. Chodorow, M. Gamon, and J. Tetreault. (2014). "Automated Grammatical Error Detection for Language Learners". (2nd ed). [On-line]. Available: https://www.morganclaypool.com/doi/abs/10.2200/S00562ED1V01Y201401HLT025 [Mar. 21, 2021].

[9] T. Brants and A. Franz. (2006). "Web 1T 5-gram Version 1 - Linguistic Data Consortium." 2006. [On-line]. Available: https://catalog.ldc.upenn.edu/LDC2006T13 [Mar. 21, 2021].

[10] J. Zhang, Y. Zeng, and B. Starly. (2021, Mar.). "Recurrent neural networks with long term temporal dependencies in machine tool wear diagnosis and prognosis." *SN Appl. Sci.* [On-line]. 3(4), p. 442. Available: https://link.springer.com/article/10.1007/s42452-021-04427-5 [Apr. 28, 2021]

[11] N. Verhaert and D. Sandra. (2016). "Homofoondominantie veroorzaakt dt-fouten tijdens het spellen en maakt er ons blind voor tijdens het lezen." *Levende Talen Tijdschr*. [On-line]. Available: https://lt-tijdschriften.nl/ojs/index.php/ltt/article/view/1632 [Mar. 21, 2021].

[12] "d / dt / t." Internet: https://www.vlaanderen.be/taaladvies/d-dt-t, 2021 [Apr. 28, 2021].

[13] H. Schmid. (1997). "Probabilistic Part-of-Speech Tagging Using Decision Trees," *New Methods in Language Processing*.[On-line]. pp. 154–164. Available: https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger1.pdf [Mar. 21, 2021].

[14] M. Honnibal and I. Montani. (2017). "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." [On-line]. Available: https://sentometrics-research.com/publication/72/ [Mar. 21, 2021].

[15] P. Koehn. (2005). "Europarl: A Parallel Corpus for Statistical Machine Translation." *Conference Proceedings: the tenth Machine Translation Summit*. [On-line]. pp. 79–86. Available: http://mt-archive.info/MTS-2005-Koehn.pdf [Mar. 21, 2021].

[16] H. Paulussen, L. Macken, W. Vandeweghe, and P. Desmet. (2013). "Dutch Parallel Corpus: A Balanced Parallel Corpus for Dutch-English and Dutch-French." [On-line]. pp. 185–199. Available: https://link.springer.com/chapter/10.1007/978-3-642-30910-6_11 [Mar. 21, 2021].

[17] "Index of /nlwiki/." Internet: https://dumps.wikimedia.org/nlwiki/, 2021 [Apr. 28, 2021].

[18] "LIIR – Home." Internet: http://liir.cs.kuleuven.be/software_pages/dt_correction_dataset_preprocessing.php, 2018 [Mar. 21, 2021].

[19] "Circus Maximus." Internet: https://www.scholieren.com/verslag/werkstuk-geschiedenis-circus-maximus, 2007 [Mar. 21, 2021].

[20] "De gevolgen van de ontdekkingsreizen." Internet: https://www.scholieren.com/verslag/werkstuk-geschiedenis-de-gevolgen-van-de-ontdekkingsreizen, 2003 [Mar. 21, 2021].

[21] "Aquaducten." Internet: https://www.scholieren.com/verslag/werkstuk-latijn-aquaducten, 2021 [Mar. 21, 2021].

[22] "Internationale politiek België." Internet: https://www.scholieren.com/verslag/opdracht-geschiedenis-internationale-politiek-belgie, 2004 [Mar. 21, 2021].

[23] “Cold Skin.” Internet: https://www.scholieren.com/verslag/boekverslag-engels-cold-skin-door-steven-herrick, 2010 [Mar. 21, 2021].

[24] J. Howard and S. Ruder. (2018). “Universal Language Model Fine-tuning for Text Classification.” [On-line]. Available: http://arxiv.org/abs/1801.06146 [Mar. 21, 2021].

[25] B. van der Burgh. "110k Dutch Book Reviews Dataset for Sentiment Analysis." Internet: https://github.com/benjaminvdb/DBRD, 2019 [Mar. 21, 2021].

[26] “torch.nn - PyTorch 1.5.0 documentation.” Internet: https://pytorch.org/docs/stable/nn.html [Mar. 21, 2021].

[27] S. Faltl, M. Schimpke, and C. Hackober. "ULMFiT: State-of-the-Art in Text Analysis", Internet: https://humboldt-wi.github.io/blog/research/information_systems_1819/group4_ulmfit/, 2019 [Mar. 21, 2021].

[28] “About - fast.ai,” Internet: https://www.fast.ai/about/, 2020 [Mar. 21, 2021].

[29] J. Howard and S. Gugger. (2020, Feb.). “Fastai: A Layered API for Deep Learning.” *Information.* 11(2). p. 108. Available: https://www.mdpi.com/2078-2489/11/2/108 [May 4, 2021].