Computational Linguistics in the Netherlands Journal 8 (2018) 49–65 Submitted 06/2018; Published 12/2018

Automatic detection and correction of context-dependent dt-mistakes using neural networks

Geert Heyman[*] GEERT.HEYMAN@KULEUVEN.BE
Ivan Vulić [**] IV250@CAM.AC.UK
Yannick Laevaert[*] YANNICKLAEVAERT@TELENET.BE
Marie-Francine Moens[*] SIEN.MOENS@KULEUVEN.BE

[*] *Department of Computer Science KU Leuven, Celestijnenlaan 200A, Leuven, Belgium*

[**] *Language Technology Lab, DTAL, University of Cambridge, UK*

Abstract

We introduce a novel approach to correcting context-dependent dt-mistakes, one of the most frequent spelling errors in the Dutch language. We show that by using a neural network to estimate the probability distribution of a verb's suffix conditioned jointly on its stem and context, we obtain large improvements over state-of-the-art spell checkers on three different benchmarking datasets, achieving a perfect score on a verb spelling test from *de Standaard*, a Flemish newspaper. The method is unsupervised and only relies on basic preprocessing tools to tokenize the text and identify verbs, which enables training on millions of sentences. Furthermore, we propose a method to determine which words in a sentence cause the system to make corrections, which is valuable for providing feedback to the user.

1. Introduction

Verbs in the Dutch language get assigned different inflections depending on their grammatical role and position in a sentence. For some verbs different inflections lead to the same pronunciation, making it impossible to *hear* which grammar rule applies. This phenomenon gives rise to one of the most common spelling mistakes in Dutch, commonly referred to as dt-mistakes, which even native speakers and/or language professionals are prone to make (*Paleis maakt dt-fout in kerstboodschap* 2017, *Dt-fout in Het Journaal* 2013). According to a study of spelling errors on the Internet, dt-errors were the most frequent classical spelling mistake in Dutch (Gheuens 2012).[1]

An automatic solution for the dt-problem would be very desirable, however, previous investigations found that current grammar and spelling checkers demonstrate low recall when it comes to context-dependent dt-mistakes (Laevaert 2017). In this paper, we introduce a new approach that tackles this problem. More specifically, our work has three main contributions:

- We show how we can successfully train a neural network to correct context-dependent dt-mistakes, without annotated training examples, on millions of sentences. We report large improvements over state-of-the-art grammar and spelling checkers on three different benchmarking test sets.

- We propose a generative process for creating dt-mistakes, motivated by cognitive insights (Verhaert and Sandra 2016). This enables the creation of two large-scale evaluation sets, which we use in addition to the three aforementioned test sets.

- We introduce a method for determining which words in a sentence lead the system to make its prediction. This is a valuable means for providing feedback to the users, especially if they

1. The term "classical mistake" refers to the fact that the mistake was not intentional, unlike mistakes related to cyber-slang or dialect usage.

are still not very familiar with the conjugation rules (e.g., for misspelled past participles it identifies the corresponding auxiliary verb, hereby indicating the rule that applies).

The remainder of this paper is structured as follows. Section 2 provides a brief introduction to the dt grammar rules and defines the *context-dependent dt-error correction* task. Section 3 presents an overview of related work. Section 4 describes the proposed approach and different model architectures that we investigate. Section 5 explains how we construct the training and evaluation datasets. Sections 6 and 7 report on the setup for our experiments, and the corresponding results and observations. The conclusions and implications of this work are discussed in Section 8.

2. Dt-rules

In this section we introduce some of the Dutch verb conjugation rules (further also referred to as *dt-rules*) and the *context-dependent dt-error correction* task. Table 1 summarizes the most important dt-rules. It is not our intention to provide an in-depth explanation of all the rules, though it is important to observe that verb inflections depend on multiple factors: the verb tense, the number of the subject, the position of the subject w.r.t. the verb, the raw stem of the verb (i.e., the infinitive minus the *-en* suffix), etc. Native speakers typically do not have problems applying these rules when they can *hear* the inflection. Verbs that have a homophone form (e.g., *beantwoord* and *beantwoordt*) give rise to many spelling errors, however. Such spelling errors are typically referred to as dt-mistakes. [2] In this work, we focus on correcting context-dependent dt-mistakes. The sentence *"Ik **beantwoordt** de vraag."*, contains an example of a context-dependent dt-mistake: *beantwoordt* is a correct Dutch verb form, hence an interpretation of the rest of the sentence is required to determine which grammar rule applies and detect the mistake. If instead the misspelled verb was *beantwoort*, then it would not have been a context-dependent mistake as *beantwoort* is not part of the Dutch vocabulary and can therefore be identified using dictionary lookups.

We formally define context-dependent dt-correction as the correction of a sentence $\tilde{s}$, consisting of N words $x_{1..N}$ to a sentence s such that incorrectly spelled, context-dependent dt-errors are corrected.

3. Related work

Classical works in context-dependent spelling correction were based on n-gram language models (Atwell and Elliott 1987, Gale and Church 1990, Church and Gale 1991, Mays et al. 1991). An important drawback of these approaches is that the n-gram assumption inhibits learning long-range dependencies. The use of higher order n-gram models to mitigate this issue generalizes badly due to the sparsity of language. Another line of research views context-dependent spelling correction as a disambiguation problem, where different classifiers are trained for each word pair that can be confused (Yarowsky 1994, Gale et al. 1995, Golding 1996, Golding et al. 1999, Mangu and Brill 1997). This approach is ill-suited for dt-correction as it does not account for the fact that the correct spelling of any given verb depends on the same set of rules. Learning different classifiers for each confusion set (e.g., {antwoord, antwoordt}, or {vind, vindt}) therefore limits the generalizations that can be made from a training set.

2. Some sources use a more narrow definition of dt-mistakes and only consider rules 1-6 in Table 1.

#	tense	usage	subj. position	rule	example + translation
1	present	1st person	anywhere	stem	Ik **beantwoord** je vraag. I **answer** your question.
2	present	2nd 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	2nd person	not after the verb	stem + t	Jij **beantwoordt** de vraag. You **answer** the question.
6	present	3rd 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: Some of the main Dutch verb conjugation rules that illustrate how confusion can arise between homophone verb forms, for which it is impossible to *hear* which rule applies (e.g., *beantwoord* vs *beantwoordt*). Note that this table is not a comprehensive overview of all Dutch verb conjugation rules and that not all these rules apply to all verbs (e.g., there are verbs for which the past participles end with *-en*). † (d/t): depending on the last character of the raw stem (infinitive - *en*) *-d*, *-t* or *nothing* should be added.

Prior work in context-dependent dt-error correction typically relies on handcrafted rules (e.g., Vosse (1992) defines an augmented context-free grammar and use a shift-reduce parser to detect grammatical errors due to morphosyntactic inconsistencies) or uses a shallow statistical model from a limited context window. While rules can be precise, they are too restrictive and fail to detect errors when verb and subject/auxiliary verb are further apart. The work of Stehouwer et al. (2009) can be seen as a first attempt to learn a dt-correction system in a data-driven fashion. They use IGTree, a fast approximation of k-nearest neighbor classification (Daelemans et al. 1997), to identify the correct dt-verb forms based on fixed-length feature vectors constructed from context windows of four words. However, the use of small, fixed-length context windows is an oversimplification that will again run into problems when verb and subject/auxiliary are further apart. The architectures we propose address this issue by using a deep classifier which uses the full sentence rather than a limited window of context words, and learns to represent the context with distributed representations instead of predefined sparse feature vectors, enabling better generalization over contexts.

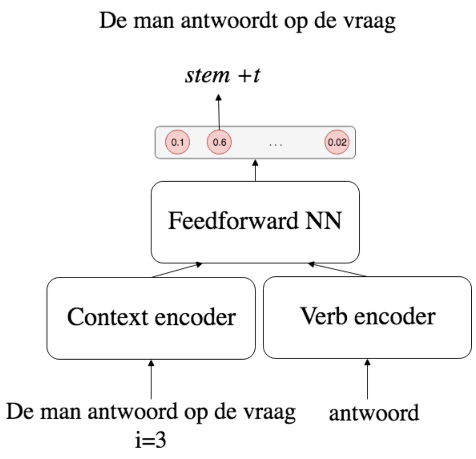

Figure 1: Architectural overview of the dt-corrector. The input to the system is the position of the verb i and the sentence where the input verb is replaced by its stem. Two neural networks are responsible for encoding the representations of the context and the verb respectively. The resulting representations are concatenated and fused by a feed-forward neural network and transformed to a probability distribution over suffixes using a softmax layer.

4. Approach

We use a neural network to estimate the conditional probability distribution of the suffix of the verb x_i at position i given the stem of the verb, the other words in the sentence, and the position of the verb in the sentence (see Equation 1). During prediction, we use this distribution to select the most likely homophone verb form.

$$p(\mathit{suffix}(x_i) \mid x_{1..i-1}, \mathit{stem}(x_i), x_{i+1..N}, i) \tag{1}$$

Figure 1 presents an architectural overview of the system. We train a neural network that estimates the conditional distribution over verb suffixes (no suffix, -t, -d, -e, -de, -te, -en, -den, -ten) given the stem of the verb, the other words in the sentence, and the position of the verb in the sentence. The network can be conceptually divided into three components: a verb encoder, which builds a representation for the stem of the input verb; a context encoder, which builds a representation for the sentence using the position i of the verb within this sentence; and a feed-forward neural network that fuses the verb and sentence representations and transforms them to a probability distribution over suffixes using a softmax layer.

We also investigated an alternative framework where instead of predicting the suffix from the stem, we predict the edit rules that correct a potentially wrongly-spelled verb given its context. This approach has the drawback that it assumes a large annotated set of dt-mistakes. In a preliminary investigation (Laevaert 2017), we created such a dataset by automatically introducing mistakes according to a handpicked distribution of dt-errors. However, we found that the performance drops significantly when the dt-error distributions of the training and test sets do not match, making it infeasible to build a practical system with this approach.

In this work we show it is possible to obtain a highly accurate dt-correction system without making assumptions about the distribution of dt-mistakes. In the remainder of this section, we describe our architecture in further detail.

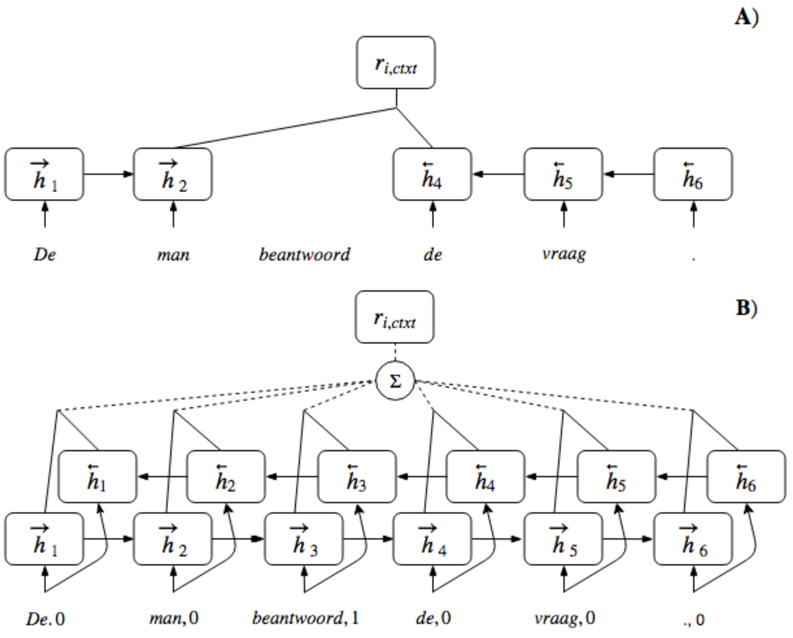

Figure 2: Illustration of the context encoders: A) BiLstm, B) BiLstm + Attention

4.1 Verb stem representation

We hypothesize that a good verb stem representation is useful for a) determining the suffix of past participles, this is either *-d*, *-t*, or *no suffix* depending on the last letter of the raw verb stem; and b) learning a bias towards the most used suffix for a given stem. In the Europarl dataset for instance, *wordt* is much more frequent than *word*. A bias towards the more frequent verb forms could benefit the precision of a system in cases where it is uncertain about how to interpret the input sentence. Based on this intuition we will experiment with three different verb stem representations:

– CHARS obtains a verb stem representation $r_{i,stem}$ from the last state of an *LSTM* (Hochreiter and Schmidhuber 1997) to which the verb stem is fed as a sequence of characters. Characters are represented as one-hot vectors;

– CHARS+WORD obtains a verb representation $r_{i,stem}$ by concatenating the CHARS representation with a word embedding of the stem;

– LAST_CHAR+WORD obtains a verb representation $r_{i,stem}$ by concatenating the last character of the stem embedded into a three-dimensional vector with a word embedding of the stem. The three-dimensional character embedding is motivated by the fact that characters can be grouped in three categories based on the suffix that should be added as a past participle.

4.2 Context representation

From the context it should be clear which dt-rule applies (see Table 1). It should therefore encode information about the subject, the tense in which the verb is used, and the position of the verb w.r.t. the subject. We experiment with two different context representations (see Figure 2):

– *BiLSTM* (Figure 2A) obtains a context representation $r_{i,ctxt}$ by concatenating the states, $\overrightarrow{h}_{i-1}$ and $\overleftarrow{h}_{i+1}$ of a bidirectional LSTMs (Schuster and Paliwal 1997, Graves and Schmidhuber 2005).

A bidirectional LSTM consists of two LSTMs: $LSTM_{forward}$ computes its states $\overrightarrow{h}_1..\overrightarrow{h}_N$ by processing the words in order, $LSTM_{backward}$ computes $\overleftarrow{h}_N..\overleftarrow{h}_1$ processes the words in reverse order.[3] Each word x_j is represented by its word embedding $E(x_j)$ before feeding it to the LSTMs.

$$r_{i,ctxt} = \overrightarrow{h}_{i-1} \;\|\; \overleftarrow{h}_{i+1} \tag{2}$$
$$\overrightarrow{h}_j = LSTM_{forward}(\overrightarrow{h}_{j-1}, E(x_j)) \tag{3}$$
$$\overleftarrow{h}_j = LSTM_{backward}(\overleftarrow{h}_{j+1}, E(x_j)) \tag{4}$$

Here $\|$ denotes concatenation.

– *BiLSTM + Attention* (Figure 2B) obtains a context representation $r_{i,ctxt}$ by first encoding the sentence, for which the relevant verb is replaced by its stem, with a bidirectional LSTM, and then using an attention mechanism to extract the relevant information from its states $h_{1..N}$. A word x_j is fed to the LSTM as the concatenation of its corresponding word embedding $E(x_j)$ and a binary indicator $I_{j=i}$ to identify the position of the verb.

An attention mechanism summarizes a sequence of vectors to a single vector as a linear combination of these vectors. The motivation for an attention mechanism is twofold. Firstly, attention models are typically beneficial when dealing with long sequences. They could therefore perform better than *BiLSTM* in cases where the subject and the direct verb are far apart. *BiLSTM* always uses the states next to the verb stem, even if the subject is not located nearby. The attention mechanism, on the other hand, calculates a weighted combination of all the states. Second, an attention mechanism could help with gaining insights in what the model has learned. By visualizing the attention weights we might uncover what words/patterns were most relevant for building the context representation.

There have been different attention mechanisms proposed in the literature. In this paper, we use single-head attention with an additive attention scoring function (Bahdanau et al. 2015), the best performing attention mechanism for neural machine translation in the comparison of Britz et al. (2017).[4] The attention weights α_j are calculated by normalizing the scores $score_1$, ..., $score_N$ calculated by the additive scoring function f_{add}, which compares a hidden state h_j with a query vector q, computes the dot product between the result (after passing it to a $tanh$ activation function) and a vector v. The vectors q and v are learned jointly with the rest of the network parameters. $E(x)$ refers to the embedding of word x.

3. Note that because the model only requires $\overrightarrow{h}_{i-1}$ and $\overleftarrow{h}_{i+1}$ we do not need to actually process the forward and backward sequences all the way to the end, see Figure 2.
4. We also performed preliminary experiments with multi-head attention, but this yielded no improvements.

$$r_{i,ctxt} = Attention(h_{1..N}) \tag{5}$$

$$h_j = \overrightarrow{h}_j \parallel \overleftarrow{h}_j \tag{6}$$

$$\overrightarrow{h}_j = LSTM_{left}(\overrightarrow{h}_{j-1}, E(x_j)) \tag{7}$$

$$\overleftarrow{h}_j = LSTM_{right}(\overleftarrow{h}_{j+1}, E(x_j)) \tag{8}$$

$$Attention(h_{1..N}) = \sum_{j=1}^{N} \alpha_j \, h_j \tag{9}$$

$$\alpha_j = \frac{exp(score_j)}{\sum_{k=1}^{N} exp(score_k)} \tag{10}$$

$$score_j = f_{add}(q, h_j) \tag{11}$$

$$f_{add}(q, h_j) = v \cdot tanh(W_{add,q} \, q + W_{add,h} \, h_j) \tag{12}$$

4.3 Transforming representations to a suffix distribution

The verb and context representations are concatenated and projected to a C-dimensional vector by a feed-forward neural network, where C is equal to the number of suffixes. The softmax function normalizes the resulting vector to a probability distribution over suffixes. For our experiments the feed-forward neural network consists of a single hidden layer with 128 dimensions and uses a ReLU activation function, which is defined as $max(0, x)$ and is the default recommendation in modern neural networks (Jarrett et al. 2009, Nair and Hinton 2010, Glorot et al. 2011, Goodfellow et al. 2016).

$$p(y|x_{1..i-1}, stem(x_i), x_{i+1..N}, i) = Softmax(l_i) \tag{13}$$

$$l_i = W_l \, r_i + b_l \tag{14}$$

$$r_i = FeedForward(r_{i,stem} \parallel r_{i,ctxt}) \tag{15}$$

4.4 Training and prediction

We use the cross-entropy with the empirical distribution of the training data $\mathcal{D}$ as the training objective:

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{x,i \in \mathcal{D}} log \, p(suffix(x_i)|x_{1..i-1}, stem(x_i), x_{i+1..N}, i) \tag{16}$$

where $|\mathcal{D}|$ denotes the number of training examples

There are nine suffix classes in total, but during prediction we reduce this to only two relevant classes for each instance based on the observation that there can be at most one (real-word) homophone verb form for any given verb. For example *beantwoord* can only be confused with *beantwoordt*, not with for instance *beantwoorde* as their pronunciations differ; similarly *beantwoorde* can only be confused with *beantwoordde*. Hence, during prediction we identify which two suffixes are relevant based on the input word[5] and force the system to predict the most likely of the two.

5. The last letter of the stem and the suffix of the input word uniquely determine the verb forms that can be confused.

5. Dataset & preprocessing

Training neural networks requires a significant amount of data. To our knowledge, no annotated corpus of dt-mistakes is available. By reducing the dt-correction task to a verb suffix prediction problem as described in the previous section, it suffices to have a large dataset of correctly written Dutch text. To this end, we used the Dutch portion of the Europarl corpus (Koehn 2005), which is written by language professionals.

To have a reliable evaluation of dt-correction systems, we require a significant number of sentences with annotated dt-mistakes. We have therefore decided to automatically introduce mistakes in 20,000 Europarl sentences to create a validation and test set of 10,000 sentences each. To verify how well a system generalizes on out-of-domain test data and to enable a fair comparison with existing systems, we construct three additional out-of-domain test sets from online verb spelling tests. The remainder of this section discusses the different preprocessing steps and our methodology for creating validation and test sets. The datasets and preprocessing scripts are available at `http://liir.cs.kuleuven.be/software.php`.

5.1 Identifying and stemming verbs

We automatically identify verbs in a sentence based on their part-of-speech (PoS) tags using TreeTagger (Schmid 1994).[6] For some verbs we can easily derive their stem based on their last character(s), e.g., if a verb ends with $-dt$. Other word endings are ambiguous, however (e.g., a *d*-suffix could be the result of rule 3 of Table 1, in which case we can not be certain if *d* is part of the stem). From the infinitive of the verb, in contrast, we can determine the stem unambiguously. We therefore automatically obtain the lemmas using TreeTagger. For verbs for which the lemma is unknown to the tagger, we derive the stem from the dt-rules if possible (e.g., when a verb ends with $-dt$).

5.2 Identifying relevant verbs

Not all verbs are relevant for training and evaluating a dt-correction system. For evaluation purposes, we are only interested in dt-homophones as these are the verbs for which spelling errors occur. We thus automatically construct a list of dt-homophones, which is used to filter verbs during evaluation and testing. For the training data, there is no need to be so restrictive: as the dt-rules apply for the majority of Dutch verbs, we can significantly increase the amount of training data if we also incorporate regular, non-dt verbs. To this end, in addition to dt-homophones, we use all regular verbs for which TreeTagger can obtain the lemma, so that the stem can be determined reliably. We verify if a verb is regular, by constructing a set of irregular verb forms from online grammar sources [7,8], and by checking verbs against this list.

5.3 A generative process for dt-mistakes

To create the validation and test datasets, we need a scheme to introduce dt-mistakes in text. Cognitive research on dt-mistakes has found that people tend to use the more frequent spelling of a homophone verb form when they are distracted or put under time pressure (Verhaert and Sandra 2016). Based on this insight, we propose a generative process for dt-mistakes, summarized in Algorithm 1.

From every sentence s in the text corpus C, the verbs are identified using TreeTagger. For every verb that has a homophone spelling, we sample a binary variable $i_{focused}$ from a Bernoulli distribution that indicates whether a person is focused when writing the verb. In the scenario where

6. Frog (Van den Bosch et al. 2007) could have been a good alternative to TreeTagger.
7. `https://educatie-en-school.infonu.nl/taal/28516-regelmatige-en-onregelmatige-werkwoorden-in-de-ott.html`
8. `http://users.telenet.be/orandago/nederlands/ww.doc`

Algorithm 1: Generative process for introducing dt-mistakes motivated by cognitive insights.

```
initialize: the focus parameter $p_f$;
for $s$ in $C$ do
    $V$ = identify_verbs($sentence$)
    for $v$ in $V$ do
        sample $i_{focused} \sim Bernoulli(p_f)$
        if not $has\_homophone\_spelling(v)$ or $i_{focused} = 1$ then
            continue
        $v_{alt} = homophone\_spelling(v)$
        sample $p_{error} \sim Beta(freq(v_{alt}), freq(v))$
        sample $i_{error} \sim Bernoulli(p_{error})$
        if $i_{error} = 1$ then
            $replace(s, v, v_{alt})$
```

the person is distracted (i.e., $i_{focused} = 0$), we sample the probability p_{error} that the person will use the wrong spelling from a Beta distribution where the frequencies of the two spellings are used as the concentration parameters.[9] p_{error} is used as the parameter of a Bernoulli distribution to sample another binary indicator i_{error}. When $i_{error} = 1$, we replace v by its homophone counterpart v_{alt}, hereby introducing a dt-error. This process ensures that we will never introduce mistakes that lead to words that do not exist in Dutch and that mistakes where the used verb form is dominant to the correct verb form are more likely, hence incorporating the insights of Verhaert and Sandra (2016).

We note that the generative process could potentially be made more cognitively plausible by incorporating contextual information as it has been shown that the immediate context in which homophones occur also plays a role in their selection (Daelemans and van den Bosch 2007). As the main goal of this work is to build a successful dt-correction system, a rigorous comparison of different generative processes falls out of our scope. We leave this for further research.

5.4 Out-of-domain test sets

To allow an unbiased comparison with existing spell checkers, we construct three out-of-domain (i.e., not related to the domain of the Europarl dataset) test sets from online spelling tests containing 20 verb conjugation exercises each: 1) *Nooit meer dt-fouten* from *de Standaard* (a Flemish news paper)[10]; 2) the *HBO taaltoets, spelling werkwoordsvormen* from *Uitgeverij Pak* (a Dutch publishing house)[11]; and 3) a test from *Nederlandse taaltest* also from *Uitgeverij Pak*[12]. The exercises use fill-in-the-blanks type of questions, where a person has to add the right suffix to a given verb stem within a given sentence. To be able to compare with other spelling checkers, we manually fill-in the blanks such that we get the wrongly-spelled homophones. For seven of the sixty verbs no real-word homophone existed (e.g., for *stond*: *stondt* and *stont* are not correct Dutch words). These are retained from the test sets because replacing them would result in context-independent errors.

5.5 Statistics

Statistics about the distribution of suffixes in the resulting train, development and test sets are shown in Table 2.

9. Note that when someone is distracted he/she can still use the correct spelling.
10. `https://www.standaard.be/cnt/dmf20141103_01356248`
11. `http://www.hbotaaltoets.nl/spelling-werkwoordsvormen`
12. `http://www.nederlandsetaaltest.nl/spellingtest-werkwoorden`

dataset	-''	-t	-d	-e	-de	-te	-en	-den	-ten	#verbs	#errors
europarl-train	314k	762k	464k	27k	102k	24k	1,097k	22k	2,428	2,814k	n/a
europarl-dev	931	6,754	3,796	65	0	3	308	0	0	12k	1,726
europarl-test	888	6,814	3,861	69	1	6	284	3	0	12k	1,665
de Standaard	1	10	9	0	0	0	0	0	0	20	20
HBO taaltoets	2	5	2	2	0	1	1	1	2	16	16
nl taaltest	2	8	1	2	0	1	1	1	1	17	17

Table 2: Label distribution of the train, development, and test sets. -'' denotes that the verb only consists of the stem.

6. Experimental setup

We used 100-dimensional word embeddings, pre-trained with continuous Skip-gram using the `word2vec` toolkit with default hyper-parameters. The LSTMs consist of two layers and 128 memory cells for each layer. We use dropout regularization (Srivastava et al. 2014) with dropout probability 0.1. For the LSTMs we use variational recurrent dropout (Gal and Ghahramani 2016).

We use the Adam optimizer with the recommended hyper-parameters (Kingma and Ba 2015) and train for a maximum of 500,000 iterations with a mini-batch size of 100, saving checkpoints every 1,000 iterations and selecting the checkpoint with the best F1-score on the development set for evaluation on the test sets.

In preliminary experiments, we found that training on plural verb forms (ending with *-en, -den, -ten*) significantly elongates the training time without having a beneficial impact on dt-correction performance due to the large imbalance between the label frequencies of *-en*, *-den* and *-ten* in the training set (see Table 2). We therefore did not train on the plural forms in the experiments reported below. During prediction, the system will not check the spelling of plural forms.

We use the following evaluation metrics (Reynaert 2008): accuracy (*acc*), precision (*prec*), recall (*rec*), and F_1 (F_1). In this context we define true positives (*tp*) as the instances where the system correctly changes the input spelling; false positives (fp) are the cases where the system introduces an error; false negatives (fn) are wrongly spelled verbs which the system did not correct; true negatives (*tn*) are correctly spelled which the system left untouched.

$$acc = \frac{tp + tn}{tp + tn + fp + fn} \tag{17}$$

$$prec = \frac{tp}{tp + fp} \tag{18}$$

$$rec = \frac{tp}{tp + fn} \tag{19}$$

$$F_1 = \frac{2\, prec \cdot rec}{prec + rec} \tag{20}$$

7. Experiments

7.1 Influence of spelling errors on PoS-tagger

In this experiment, we verify the robustness of the PoS-tagger w.r.t. dt-spelling errors. Using the development set with annotated dt-mistakes, we verified if TreeTagger could still assign the same PoS tag when a dt-verb was replaced by its homophone. We found that 98.5 % of the wrongly-spelled verbs are still identified as verbs. However, for only 19.5 % of the wrongly-spelled verbs, the correct tense was preserved. In the majority of the cases the tagger will assign the tense that is compatible with the wrongly-spelled verb. This is unsurprising as PoS-taggers are typically trained on high-quality corpora, with a low amount of spelling errors. Hence, taggers trained on such corpora will have learned that the verb form is a strong predictor for its tense. From this result it is clear that the accuracy of the tenses for wrongly-spelled verbs as recognized by TreeTagger is far too low to be useful in a dt-correction system.

7.2 Context-agnostic baselines

In this experiment, we investigate the performance of models that do not use context. We compare four models: *majority class* uses the input verb to decide which homophone suffixes it should consider (see Subsection 4.4) and then picks the suffix that is most frequent in the training data; the other three models use the different verb representations introduced in Subsection 4.1: *chars*, *last_char+word*, and *chars+word*. The results are reported in Table 3. We see that the best context-agnostic models (*chars* and *suffix+word*) obtain an F1-score 55.59 on the test set. There is not a lot of difference between verb representations. When comparing *majority class* with the other models, we find support for the hypothesis that using verb representations can help to improve the precision of the system (see Subsection 4.1).

	F_1		*acc*		*prec*		*rec*	
model/dataset	dev	test	dev	test	dev	test	dev	test
majority class	26.98	25.81	60.06	60.05	18.38	17.42	50.72	49.79
chars	**56.13**	**55.59**	88.78	88.95	65.11	63.32	49.33	49.55
last_char+word	56.02	**55.59**	88.81	89.04	65.4	64.01	48.99	49.13
chars+word	55.84	55.5	88.77	89.02	65.22	63.88	48.81	49.07

Table 3: Results of baseline models that do not use context, evaluated on the Europarl development (dev) and test (test) sets.

7.3 Context and stem encodings

In this experiment, we explore the different combinations of verb and context representations introduced in Subsections 4.1 and 4.2. The results are displayed in Table 4. We find that models that use attention to encode the context outperform the other models for all verb representations. The verb representation does not seem to have a large impact on the results. With attention, *chars+word* obtains the best results on the development set, but on the test set there are no significant differences between the three encodings. All six architectures' results exhibit large performance gains over the baselines that do not use context (Table 3). This shows that models are successfully leveraging the context information.

The fact that these neural networks only use the stem of the verb that is being checked makes these results particularly impressive. It means that the system output is largely independent of the number of mistakes in the input spelling.[13] We can hence expect nearly identical accuracy scores in test sets with a higher or lower ratio of mistakes.

13. With exception of the plural forms (-en, -ten, -den) for which we always leave the input spelling unchanged.

model/dataset	F_1 dev	F_1 test	*acc* dev	*acc* test	*prec* dev	*prec* test	*rec* dev	*rec* test
BiLSTM+chars	97.98	97.44	99.42	99.29	98.88	97.88	97.1	97
BiLSTM+last_char+word	97.77	97.43	99.36	99.29	98.99	98.17	96.58	96.7
BiLSTM+chars+word	97.75	97.61	99.35	99.34	98.64	98.35	96.87	96.88
BiLSTM+Attn+chars	98.07	**97.85**	99.44	99.4	98.77	98.6	97.39	97.12
BiLSTM+Attn+last_char+word	98.19	97.82	99.48	99.4	99.11	98.48	97.28	97.18
BiLSTM+Attn+chars+word	**98.33**	**97.85**	99.52	99.4	99.29	98.48	97.39	97.24

Table 4: Comparison of combinations of verb and context encodings on *Europarl-dev* (dev) and *Europarl-test* (test).

7.4 Extending the training data

In this experiment, we analyzed the effect of omitting non-dt verbs from the training data (all other experiments include regular, non-dt verbs in the training set, see Subsection 5.2). We find F_1-scores of 97.75 and 97.52 on the Europarl development and test sets respectively. This is a decrease compared to when the model is also trained on non-dt verbs, which obtained F_1-scores of 98.33 and 97.85 (see Table 4). This confirms the hypothesis that a model that is also trained on non-dt verbs generalizes better.

7.5 Comparison with existing systems

In this experiment, we compare our system with existing grammar and spell checking systems on the out-of-domain test sets: *de Standaard (dS)*, *HBO taaltoets* (HBO) , and *Nederlandse taaltest* (NLTT). We compare with the following systems:

– *Microsoft Office Word*, we experimented with three versions: Office Professional Plus 2013; Office 365, desktop version; and Office 365, Word online[14]. We report the results for Office 365, Word online which demonstrated the best performance.

– *Schrijfassistent* (D'Hertefelt et al. 2014, Houthuys 2016, *VRT lanceert online schrijfhulp* 2016)[15], a tool for checking grammar, writing style, and spelling in Dutch developed by a collaboration of *het Instituut Levende Talen, KU Leuven*; *de Standaard* (a Flemish newspaper); and *VRT* (the Flemish public broadcasting organization).

– *languatool.org* [16], an open-source grammar, writing style, and spelling checker for several languages including Dutch. For dt-correction it relies on a rule-based system (OpenTaal 2014).

– *valkuil.net*, an online spell checker for Dutch which relies on a data-driven approach for dt-rules. This system uses a statistical, context-based system which uses a fixed context window of four words to identify the correct verb form (Stehouwer and Van den Bosch 2009).

Table 5 reports the results. We find that *BiLSTM+Attn+chars+word* outperforms all the other grammar and spelling checkers by a significant margin on all test sets. It is particularly effective on the *de Standaard* test set, where it obtains a perfect score. For the other two test sets the system has perfect precision[17], though the recall is significantly lower than for *de Standaard*. This is due to

14. Office 365 was tested in April 2018.
15. `http://schrijfassistent.standaard.be/index.php`
16. `https://languagetool.org/nl`
17. It should be noted that due to the large number of errors in the test sets, high precision scores are not surprising as there is little opportunity to introduce errors in correctly written words.

the fact that these test sets query all verb forms (i.e., verbs ending with *-en, -e, -d,* or *-t*), whereas the *de Standaard* test set focuses on verbs that end with *-d* or *-t*. Homophones that end with *-ten, -den, -te, -de, -e* occur much less frequently and thus are more difficult for our system to predict correctly.[18] Furthermore, some of the input verb forms did not occur at all in the Europarl corpus, and will therefore not be recognized as a dt-verb.

The performance improvement with respect to the existing spell checkers comes from a higher recall. We found that the spell checkers were unable to find any of the errors where the misspelled verb and the relevant context words (e.g., auxiliary verb or subject) were not adjacent to each other. It is striking that valkuil.net did not detect any of the mistakes in the test sets. This is probably related to the fact that the system was tuned to have high precision.[19] The higher recall scores obtained by *BiLSTM+Attn+chars+word* support the intuition that learning distributed representations of the whole context yield better generalization than sparse one-hot representations of fixed context windows. A straightforward way to further improve the recall of our system is to extend our training corpus with other high quality Dutch texts such as news articles. Furthermore, it could be beneficial to consider a strategy to subsample the frequent suffix classes to boost the recall for the infrequent suffixes.

	F_1			*prec*			*rec*		
model/dataset	dS	HBO	NLTT	dS	HBO	NLTT	dS	HBO	NLTT
language tool	0.00	0.00	11.11	0.00	0.00	100.00	0.00	0.00	5.88
Schrijfassistent	33.33	30.00	38.10	100.00	75.00	100.00	20.00	18.75	23.53
valkuil.net	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MS Word	26.09	11.76	21.05	100.00	100.00	100.00	15.00	6.25	11.76
BiLSTM+Attn +chars+word	**100.00**	**57.14**	**66.67**	100.00	100.00	100.00	100.00	40.00	50.00

Table 5: Comparison with other grammar and spelling checkers.

7.6 Acquiring insight in the model predictions

From previous experiments it is clear that the models have learned to successfully leverage context for making accurate dt-corrections. The aim of this experiment is to uncover *how* the models are using the context. In particular, we are interested in measuring the impact of each word in the sentence on the prediction. We experimented with two alternative *word saliency* metrics:

– The weight α_j that the attention mechanism assigns to the state corresponding to the word x_j.

– The difference in the probability of the predicted class when we dropout the word embedding $E(x_j)$ of x_j (i.e., replacing $E(x_j)$ with a vector with all zeros) compared to normal prediction, where we use all words.

We validated the effectiveness of these word saliency metrics by visualizing their output and manually inspecting if it provides a plausible explanation for the model's prediction. We found that the dropout-based probability model yields intuitive results. Figure 3 shows heatmaps for the dropout-based word saliency metric for sentences of the *de Standaard* test dataset. For finite verbs in present tense the word with the largest impact is the subject, for past participles this is the auxiliary verb. For the attention-based metric the results were not intuitive. The fact that attention

18. Recall from the experiment setup that the system was in fact not trained on plural forms because verbs that end with *-ten* and *-den* are too infrequent.
19. http://valkuil.net/info

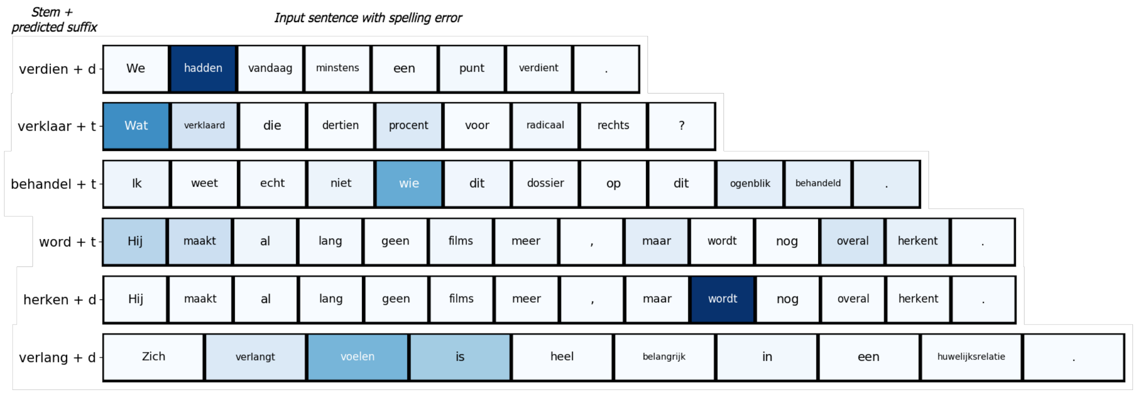

Figure 3: Heatmaps of the impact of each word on the prediction of the neural network for wrongly spelled dt-verbs from the *De Standaard* test set. Impact of a word is measured by the decrease in probability mass of the predicted class when we mask the word in the context encoding (i.e., replace its word embedding by a vector of zeros).

weights have so little explanatory value is surprising, it indicates that the states to which the neural network attends still encode much relevant information from previous and/or subsequent states.

8. Conclusion

In this work, we introduced a new approach to automatic correction of context-dependent dt-mistakes, one of the most frequent spelling errors in the Dutch language. By learning a neural network to estimate the probability distribution of a verb's suffix conditioned on its stem and the context in which it occurs, we were able to build a spelling correction system that achieves state-of-the-art performance on three different benchmarking datasets. The method does not require annotated training examples and only relies on basic preprocessing tools to tokenize the text and identify verbs, which enables training on millions of examples. Furthermore, we proposed a method to determine which words in a sentence cause the system to make corrections, which can be a valuable way of providing feedback to the user. We still see room for further improvement for verb suffixes that occur less frequently (*-den*, *-ten*, *-de*, *-te*, *-e*). A strategy where the more frequent suffix classes are subsampled, may help dealing with these cases.

Acknowledgements

The authors would like to thank Artuur Leeuwenberg and the anonymous reviewers for their helpful feedback. This research was funded by the SCATE project (IWT-SBO 130047). Ivan Vulić is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909). Marie-Francine Moens is partially supported by the ERC Advanced Grant 2017 CALCULUS (Horizon 2020 - 788506).

References

Atwell, Eric and Stephen Elliott (1987), Dealing with ill-formed English text, The Computational Analysis of English, *The Computational Analysis of English: A Corpus-Based Approach.*

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015), Neural Machine Translation by

Jointly Learning to Align and Translate, *International Conference on Learning Representations (ICLR)*, pp. 1–15. https://arxiv.org/abs/1409.0473.

Britz, Denny, Anna Goldie, Thang Luong, and Quoc Le (2017), Massive Exploration of Neural Machine Translation Architectures, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)* pp. 1442—-1451, Association for Computational Linguistics, Copenhagen, Denmark. http://aclweb.org/anthology/D17-1151.

Church, Kenneth W and William A Gale (1991), Probability scoring for spelling correction, *Statistics and Computing* **1** (2), pp. 93–103. https://doi.org/10.1007/BF01889984.

Daelemans, Walter and Antal van den Bosch (2007), Dat gebeurd mei niet: computationele modellen voor verwarbare homofonen, *Tussen taal, spelling en onderwijs : essays bij het emeritaat van Frans Daems. Gent : Academia Press*, pp. 199—-210.

Daelemans, Walter, Antal Van Den Bosch, and Ton Weijters (1997), IGTree: Using trees for compression and classification in lazy learning algorithms, *Artificial Intelligence Review* **11**, pp. 407–423.

D'Hertefelt, Margot, Lieve De Wachter, and Serge Verlinde (2014), Writing Aid Dutch. Supporting students' writing skills by means of a string and pattern matching based web application, *Proceedings of the 6th International Conference on Computer Supported Education*, pp. 486–491.

Dt-fout in Het Journaal (2013). https://www.nieuwsblad.be/cnt/dmf20130213%5C_033.

Gal, Yarin and Zoubin Ghahramani (2016), A Theoretically Grounded Application of Dropout in Recurrent Neural Networks, *Advances in Neural Information Processing Systems 29*, pp. 1019–1027. http://papers.nips.cc/paper/6241-a-theoretically-grounded-application-of-dropout-in-recurrent-neural-networks.pdf.

Gale, William A and Kenneth W Church (1990), Estimation Procedures for Language Context: Poor Estimates are Worse than None, *in* Momirović, Konstantin and Vesna Mildner, editors, *Compstat*, Physica-Verlag HD, Heidelberg, pp. 69–74.

Gale, William A, Kenneth W Church, and David Yarowsky (1995), Discrimination decisions for 100,000-dimensional spaces, *Annals of Operations Research* **55** (2), pp. 323–344. https://doi.org/10.1007/BF02030865.

Gheuens, Koen (2012), Spelling op het internet; de chaos becijferd, *Levende Talen Tijdschrift* **13** (1), pp. 26–35.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011), Deep sparse rectifier neural networks, *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* **15**, pp. 315–323.

Golding, Andrew R (1996), A Bayesian hybrid method for context-sensitive spelling correction, *Proceedings of the Third Workshop on Very Large Corpora* **3**, pp. 15. http://arxiv.org/abs/cmp-lg/9606001.

Golding, Andrew R, Dan Roth, Claire Cardie, and Raymond Mooney (1999), A Winnow-Based Approach to Context-Sensitive Spelling Correction*, *Machine Learning* **34**, pp. 107–130.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016), *Deep Learning*, MIT Press.

Graves, Alex and Jürgen Schmidhuber (2005), Framewise phoneme classification with bidirectional LSTM networks, *Proceedings of the International Joint Conference on Neural Networks*, Vol. 4, pp. 2047–2052.

Hochreiter, Sepp and Jürgen Schmidhuber (1997), Long Short-Term Memory, *Neural Computation* **9** (8), pp. 1735–1780. http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735.

Houthuys, Astrid (2016), 'De Schrijfassistent': uw nieuwe personal schrijfcoach. http://www.standaard.be/cnt/dmf20161029%5C_02546399.

Jarrett, Kevin, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun (2009), What is the best multi-stage architecture for object recognition?, *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2146–2153.

Kingma, Diederik P. and Jimmy Lei Ba (2015), Adam: a Method for Stochastic Optimization, *International Conference on Learning Representations 2015* pp. 1–15. https://arxiv.org/pdf/1412.6980.pdf.

Koehn, Philipp (2005), Europarl : A Parallel Corpus for Statistical Machine Translation, *MT Summit* **11**, pp. 79—-86. http://mt-archive.info/MTS-2005-Koehn.pdf.

Laevaert, Yannick (2017), Automatische detectie en correctie van dt-fouten met recurrente neurale netwerken, Master Thesis. KU Leuven. Faculteit Ingenieurswetenschappen, Leuven.

Mangu, Lidia and Eric Brill (1997), Automatic Rule Acquisition for Spelling Correction, *Proceedings of the Fourteenth International Conference on Machine Learning* pp. 187–194. http://dl.acm.org/citation.cfm?id=645526.657126.

Mays, Eric, Fred J. Damerau, and Robert L. Mercer (1991), Context based spelling correction, *Information Processing & Management* **27** (5), pp. 517–522, Pergamon. https://www.sciencedirect.com/science/article/pii/030645739190066U.

Nair, Vinod and Geoffrey E Hinton (2010), Rectified Linear Units Improve Restricted Boltzmann Machines, *Proceedings of the 27th International Conference on Machine Learning* (3), pp. 807–814.

OpenTaal (2014), Grammaticacontrole met LanguageTool. https://www.opentaal.org/begrippenlijst/216-grammaticacontrole-met-languagetool.

Paleis maakt dt-fout in kerstboodschap (2017). https://www.nieuwsblad.be/cnt/dmf20171211%5C_03236553.

Reynaert, Martin (2008), All, and only, the errors: More complete and consistent spelling and OCR-error correction evaluation, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Schmid, Helmut (1994), Probabilistic part-of-speech tagging using decision trees, *Proceedings of the International Conference on New Methods in Language Processing.*

Schuster, Mike and Kuldip K. Paliwal (1997), Bidirectional recurrent neural networks, *IEEE Transactions on Signal Processing* **45** (11), pp. 2673–2681. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=650093.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014), Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *Journal of Machine Learning Research* **15**, pp. 1929–1958.

Stehouwer, Herman and Antal Van den Bosch (2009), Putting the t where it belongs: Solving a confusion problem in Dutch, *Computational Linguistics in the Netherlands 2007: Selected Papers from the 18th CLIN Meeting* pp. 21–36. http://ilk.uvt.nl/downloads/pub/papers/CLIN07-putting-the-t.pdf.

Van den Bosch, Antal, Bertjan Busser, Sander Canisius, and Walter Daelemans (2007), An efficient memory-based morphosyntactic tagger and parser for Dutch, *Selected Papers of CLIN 2007.*

Verhaert, Nina and Dominiek Sandra (2016), Homofoondominantie veroorzaakt dt-fouten tijdens het spellen en maakt er ons blind voor tijdens het lezen, *Levende Talen Tijdschrift* **17** (4), pp. 37—-46.

Vosse, Theo (1992), Detecting and correcting morpho-syntactic errors in real texts, *Proceedings of the third conference on Applied natural language processing*, Association for Computational Linguistics, pp. 111–118.

VRT lanceert online schrijfhulp (2016). https://www.vrt.be/nl/over-de-vrt/nieuws/2018/05/17/vrt-lanceert-online-schrijfhulp/.

Yarowsky, David (1994), Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. http://arxiv.org/abs/cmp-lg/9406034.