Token merging in language model-based confusible disambiguation

Herman Stehouwer Menno van Zaanen

TiCC, Tilburg University, Tilburg

Abstract

In the context of confusible disambiguation (spelling correction that requires context), the synchronous back-off strategy combined with traditional n-gram language models performs well. However, when alternatives consist of a different number of tokens, this classification technique cannot be applied directly, because the computation of the probabilities is skewed. Previous work already showed that probabilities based on different order n-grams should not be compared directly.

In this article, we propose new probability metrics in which the size of the n is varied according to the number of tokens of the confusible alternative. This requires access to n-grams of variable length. Results show that the synchronous back-off method is extremely robust.

We discuss the use of suffix trees as a technique to store variable length n-gram information efficiently.

1 Introduction

When writing texts, people often use spelling checkers to tackle spelling mistakes. Most available spelling checkers concentrate on non-word errors only. Since these errors consist of character sequences that are not part of the language, these errors can be (relatively[1]) easily identified. For example, in English *woord* is is not part of the language, hence a non-word error. A possible correction would be *word*.

However, even when a text does not contain any non-word errors, there is no guarantee that the text is error-free. There are several types of spelling errors where the words themselves are part of the language, but are used incorrectly in their context. Note that these kinds of errors are much harder to recognize, as information from the context in which they occur is essential to recognize and correct these errors. In contrast, non-word errors can be recognized without context.

One class of such errors, called *confusibles*, consists of words that belong to the language, but are used incorrectly with respect to their local, sentential context. For example, *She owns to cars* contains the confusible *to*. Note that this word is a valid token and part of the language, but used incorrectly in the context. Considering the context, a correct and very likely alternative here would be the word *two*. Confusibles are grouped together in confusible sets, which are sets of words that are similar and often used incorrectly in context. *Too* is the third alternative in this particular confusible set.

A typical way of solving the problem of confusibles is to use a machine learning classifier that uses information from the context of the confusible and classifies the instance, resolving the ambiguity. In this article, we tackle this problem using a language model approach. In particular, we combine the language model approach with a synchronous back-off of higher-order n-grams to lower-order n-grams [12].

The research presented here is part of a larger project, which focusses on identifying and correcting context-sensitive spelling mistakes in general. For example, in this project, a class of pragmatically incorrect words (where words are incorrectly used within the document-wide context) is considered as well.

This article is organized as follows. In the next section, we briefly describe different approaches to confusible disambiguation, concentrating on how language models can be used in the context of confusible correction. We will also explain the synchronized back-off technique in more detail. In the following section (section 3), we will treat the problem of multiple token confusibles in more detail and we will elaborate on our need to merge the focus position of a sequence for alternatives of different length, and propose different

[1] Productive constructs such as compounds make this process harder, as simple word lists will not contain all words in the language.

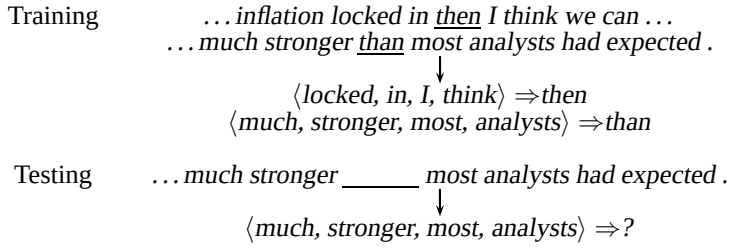

Figure 1: Instances are extracted from sentences during training. Testing classifies new instances, yielding values for the focus word position.

methods that allow us to do this. In section 4, we will discuss a practical solution to the technical issues by using suffix trees. Finally, we will end with a conclusion with some ideas on possible future work.

2 Approaches to confusible disambiguation

A typical approach to the problem of confusibles is to train a machine learning classifier to a specific confusible set. Most of the work in this area has concentrated on confusibles due to homophony (*to*, *too*, *two*) or similar spelling (*desert*, *dessert*). For instance, when word forms are homophonic, they tend to get confused often in writing (cf. the situation with *to*, *too*, and *two*, *affect* and *effect*, or *there*, *their*, and *they're* in English) [11, 15]. However, some research has also touched upon inflectional or derivational confusibles such as *I* versus *me* [4]. Typically, the confusible sets are very small (two or three elements) and the machine learner will only see training examples of the members of the confusible set. This approach is similar to that of accent restoration [1, 2, 3, 6, 9, 14, 16, 17].

The task of the machine learner is to decide, using features describing information from the context, which word taken from the confusible set really belongs in the position of the confusible. Using the example above, the classifier has to decide which word belongs on the position of the **X** in *She owns* **X** *cars*, where the possible answers for **X** are *to*, *too*, or *two*. We call **X**, the confusible that is under consideration, the *focus word*. Figure 1 illustrates the typical approach, when only words in the context are used.

2.1 Language model-based classification

Another way of looking at the problem of confusible disambiguation is to see it as a very specialized case of word prediction. The problem is then to predict which word belongs at a specific position. Using similarities between these cases, we can use techniques from the field of language modeling to solve the decision problem of finding the correct value in confusible sets.

Language models assign probabilities to sequences of words. Using this information, it is possible to predict the most likely word given a context. For example, we can use a language model to give us the probability for a sequence of n words $P_{LM}(w_1, \ldots, w_n)$. Based on this, it is possible to predict the most likely word w following a sequence of $n-1$ words, namely $\arg\max_w P_{LM}(w_1, \ldots, w_{n-1}, w)$. Obviously, a similar approach can be taken with w in the middle of the sequence. By limiting the possible values for w to those in the confusible set, we have designed a classifier based on the language model.

The advantage of the language model approach to confusible disambiguation is that the language model can handle all potential confusibles without any further training and tuning. With the language model it is possible to take the words from any confusible set and compute the probabilities of those words in the context. The element from the confusible set that has the highest probability is then selected. Since the language model can assign probabilities to all sequences of words, it is possible to define new confusible sets on the fly and let the language model disambiguate them without any further training. Obviously, this is not possible for a specialized machine learning classifier approach, where a classifier is fine-tuned to the features and classes are pre-defined for a specific confusible set.

The disadvantage of the generic (language model) classifier approach is that the accuracy is less than that of the specific (specialized machine learning classifier) approach, but previous works shows that the results are encouraging [12]. Since the specific classifiers are tuned to each specific confusible set, the weights for each of the features may be different for each set. For instance, there may be confusibles for

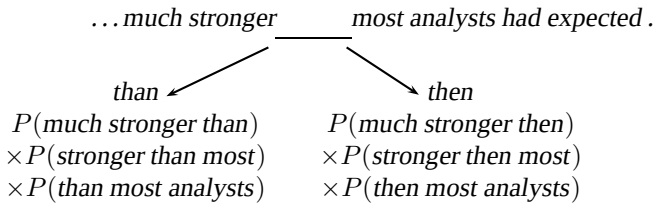

Figure 2: Computation of probabilities using the language model.

which the correct word is easily identified by words in a specific position. If a determiner, like *the*, occurs in the position directly before the confusible, *to* or *too* are very probably not the correct answers. The specific approach can take this into account by assigning specific weights to part-of-speech and position combinations, whereas the generic approach cannot do this explicitly for specific cases; the weights follow automatically from the training corpus and are the same for all words (in all situations).

Summarizing, there is no real difference between the classifier-based or language model-based disambiguation of confusibles. Both approaches classify instances. However, the main difference is the number of classes that can be classified into. The classifier-based approach limits the number of classes to the number of elements in the confusible set, whereas the language model-based approach allows for classification into any word in the language (that the model knows of). This makes the language model approach much more flexible.

2.2 Synchronized back-off in n-gram language models

The language model approach to confusible disambiguation described above relies heavily on the language model. In principle, any language model that provides probabilities given sequences can be used as components in a classifier. In previous work [12], we have used a relatively simple n-gram based classifier. An illustration of this approach can be found in figure 2.

The size of n defines the size of the context around the focus word. The example in figure 2 uses $n = 3$. This means that we do not need to compute the probability of the entire sentence, only the probabilities that will change depending on the value of the focus word will need to be computed. As usual, using larger n will give us more specific probabilities, but also introduces data sparseness problems.

There are several techniques that can be used to reduce the impact of data sparseness problems. Smoothing of probabilities or back-off methods are typically used. We have shown that the synchronous back-off technique works well in this setting.

Backoff techniques fall back on lower order n-grams when no occurrence of the higher order n-grams are found. In other words, if $P_n(w_0 \dots w_n) = 0$ then the language model uses $P_{n-1}(w_0 \dots w_{n-1}) \times P_{n-1}(w_1 \dots w_n)$. This process may continue until $n = 1$.

Applying this simple back-off strategy in the context of confusible disambiguation leads to the problem of unbalanced probabilities. For instance, it may be the case that when comparing different members of a confusible set, one of the alternatives has a non-zero tri-gram probability, while for the other alternative bi-gram probabilities are required (because their tri-gram probability is zero). It turns out that using these unbalanced probabilities yield undesirable results. Intuitively, this can be argued, since the different probabilities come from different probability spaces.

To resolve the problem of unbalanced probabilities, we have employed a back-off strategy called *synchronous back-off*. This method always uses probabilities of the same order n-gram model for each alternative in the confusible set. Whereas in the naive case, a position in two similar sequences may be computed using probabilities of different order language models, the synchronous back-off model only takes a lower order n-gram into account when all alternatives have zero probability. This is illustrated in figure 3.

3 Multiple tokens as alternatives

The language model approach to confusible disambiguation runs into problems when the alternatives in the confusible set (in particular when occurring on the focus position) contain a different number of tokens. For example, consider the case of *your* (one token) versus *you 're* (two tokens). The problem here is that for the first alternative, there is one token as focus word, whereas the other alternative has a two token focus word.

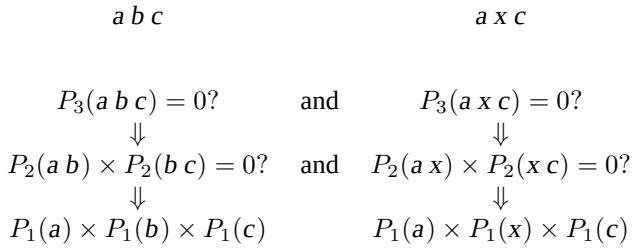

Figure 3: The probabilities of the two alternatives are computed using synchronous back-off. Only if both probabilities are zero are lower order probabilities taken into account.

Having different number of tokens as focus word has an impact on the computation of the probabilities of the sequences. For instance, when $n = 3$, the probabilities are computed based on two tokens before and after the focus word, combined with the focus word. In the case of *your*, the probabilities are based on five tokens, but in the case of *you 're* as the focus word, which contains two tokens itself, the probabilities are based on six tokens.

Experiments that compare continuous back-off (where each alternative is allowed to back off until a non-zero probability is found) against synchronous back-off already showed that comparing probabilities based on different order n-grams yields unsatisfactory results. Unfortunately, disambiguating a confusible set containing alternatives with a different number of tokens, this same situation arises. Effectively, different order n-grams will be compared.

3.1 Merging positions

Previous work resolved the problem of multiple tokens in alternatives by retokenizing the corpus, making sure that the order of the n-grams is the same for each alternative (essentially taking *you're* as one token).

The solution to the problem by merging tokens looses the information that *you're* also contains both tokens *you* and *'re*, which might be important for disambiguation.

Obviously, this approach equalizes the order of the n-grams of the alternatives, but it is not a practical solution, as for each situation with different tokens in the alternatives, retokenization is required.

Additionally, this problem also occurs when these multi-token "words" occur in non-focus word positions. Retokenizing the corpus implies that all occurrences of such words are modified. This results in a different probability distribution compared to the original, not retokenized version of the corpus.

3.2 Readjusting probabilities

From the previous section it may be clear that we would like to be able to merge tokens during testing only. This allows us to deal with different length confusible alternatives as if they all have the same size. However, we do not want to merge tokens by retokenizing the entire corpus as there may be many situations in which merging is needed.

Our solution to this problem is to retain the structural information that is present in the model, but to only adjust the counts that are required to compute the probabilities of the sequences (and hence the probabilities of the alternatives). This comes down to dynamically readjusting the probabilities when needed without retokenizing the corpus.

In situations where all alternatives contain one token, the probability of an alternative only depends on the direct context of the focus word. For instance, if we have the sequence $w = w_0 \dots w_m$ and w_f is the focus word, the probability of the sequence that is relevant for disambiguation is defined by $\Pi_{i=f-n+1}^{f} P_n(w_i \dots w_{i+n-1})$, which combines the probabilities of n-grams containing the focus word w_f.

We now extend this computation by considering the case of having a focus word consisting of two tokens: w_{f_0} and w_{f_1} (where $f_0 + 1 = f_1$). During the computation of the probabilities, these two tokens are treated as one. $\Pi_{i=f_0-n+1}^{f_1} P'_{n+1}(w_i \dots w_{i+n-1})$. This computation uses the same context as the one token alternative, but since an additional token is used, a higher order n-gram is needed. Note that this approach

is trivial to extend to alternatives with more than two tokens (which requires a proportional increase in the order of the n-grams).

So far, the approach is exactly the same, but if P' is P, then different order n-grams are compared, which is problematic, as discussed above. Clearly, different means of computing P' are needed. We will discuss four extensions and compare them against the case where $P = P'$ in section 5.

Typically, the probability of an n-gram is computed by $P_{\text{no discount}}(s) = \frac{\text{occ}(s)}{|C|-|s|+1}$, where occ returns the number of occurrences of n-gram s and $|C|$ is the number of tokens in the corpus. The denominator is number of n-grams of length $n = |s|$.

Since we are going to model the probability of a merged n-gram, we will use a similar formula, but discount the total number of n-grams in the corpus. Essentially, this comes down to pretending the corpus has been retokenized with respect to s. We will compute the probability according to $P_{\text{count}-\text{occ}}(s) = \frac{\text{occ}(s)}{|C|-|s|+1-\text{occ}(s)}$. This effectively increases the probability by pretending to reduce the size of the corpus by the number of times s occurred. This is like treating s as a single token.

This means of discounting can be taken further. We can also reduce the denominator by the number of occurrences of the uni-grams of the multi-token s. $P_{\text{count}-\min(\text{occ(sub-gram)})}(s) = \frac{\text{occ}(s)}{|C|-|s|+1-\min_{i=0}^{n}\text{occ}(s_i)}$ and $P_{\text{count}-\max(\text{occ(sub-gram)})}(s) = \frac{\text{occ}(s)}{|C|-|s|+1-\max_{i=0}^{n}\text{occ}(s_i)}$ describe this (where sub-gram is the set of words in the n-gram). As an extreme, we also propose to consider the idea of merging all bi-grams into uni-grams, resulting in $P_{\text{count}\times\frac{|\text{uni-gram}|}{|\text{bi-gram}|}}(s) = \frac{\text{occ}(s)}{(|C|-|s|+1)\times\frac{|\text{uni-gram}|}{|\text{bi-gram}|}}$.

We should note at this point that the different implementations of P' do not result in proper probabilities. The count numbers are modified and different probability spaces are combined in simple heuristic ways. However, this is not a major concern, as we are only interested in comparing the figures for different options in a set of alternatives to make a decision ($\arg\max$).

3.3 Impact

The problem of different probabilities in case of multiple token alternatives sketched here may only seem to be a minor problem with hardly any impact on the task. However, future work in this project concentrates on incorporating additional information in the language models. Details about incorporating more information will be discussed in section 6. For now, imagine how, for instance, part-of-speech information of the focus word can be incorporated. To get the probabilities right, merging tokens is preferred. However, this would also mean that the focus word will have multiple part-of-speech tags assigned to it. Starting from the part-of-speech information, keeping the tokens (and their part-of-speech information) separate is preferred, but this again introduces problems regarding the probability distributions.

Having access to a dynamic system that can readjust probabilities when needed is preferred. Essentially, this allows for easy switching between both views of the data. However, as shown above, variable order of n-grams (depending on the number of tokens in the focus word position). may be required to implement this. The next section describes a practical approach to storing and retrieving variable order n-grams based on suffix trees.

4 Practical issues

Our solution to the computation of multiple token focus word alternatives depends on access to variable length n-grams. Since our language model approach allows for dynamic creation of confusible sets, we cannot decide beforehand what the maximum required n is (apart from it being smaller than the size of the corpus). This calls for the dynamic computation of probabilities, leading to dynamic counting of n-grams in the corpus.

To keep the process of counting n-grams efficient, we store the corpus in a suffix tree. A suffix tree is a trie-based data structure that stores all suffixes of an input string in such a way that a suffix of the string can be found in linear time (in the length of the suffix). Note that suffix tree construction can also be done with a linear time algorithm (in the length of the string) [5, 13], which only needs to be done once. All suffixes occupy a single path from the root of the suffix tree to a leaf. For a simple example of a suffix tree based on the string *cacao* see figure 4.

Leaf nodes in suffix trees contain information on the begin position of the suffix leading to that specific leaf. This allows for finding the start position of the suffix that ends in that leaf. By extending this search,

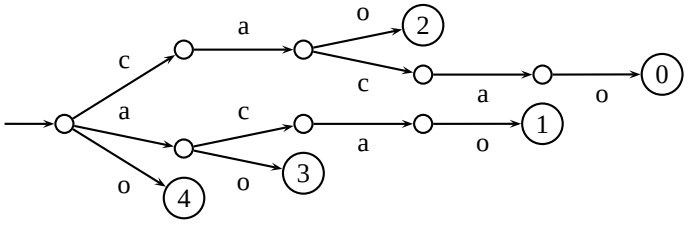

Figure 4: Suffix tree for *cacao*.

it is possible to find start positions of any sub-string. First, search the suffix tree using the symbols in the sub-string. Unless the sub-string is a suffix, this will lead to an internal node. Finding the position stored in a leaf node that can be reached from the internal node yields the start position of the sub-string.

Due to the way suffix trees are constructed, we can efficiently find the number of occurrences of sub-strings in the entire string. Starting from the root node, we find the node that belongs to the sub-string. From this node, we can count the number of leaves that it governs. This number is exactly the number of occurrences of the sub-string in the entire corpus.

Obviously, we can construct a suffix tree using tokens as elements (and not characters as was illustrated in figure 4). This allows us to efficiently identify the count of any n-gram of any order.

5 Results

To measure the impact of different means of adjusting the probabilities of the different order n-grams, we will run some experiments. First, we build a suffix tree given a training corpus, which will be used to generate n-gram counts. Next, we identify possible confusibles in the test corpus. We apply the language model classifier to see whether the correct alternatives are selected.

For training purposes, we used the Reuters news corpus RCV1 [7]. The Reuters corpus contains about 810,000 categorized newswire stories as published by Reuters in 1996 and 1997. This corpus contains around 130 million tokens.[2] For testing purposes, we used the Wall Street Journal part of the Penn Treebank corpus [10]. This well-known corpus contains articles from the Wall Street Journal in 1987 to 1989. We extract our test-instances from this corpus in the same way as we extract our training data from the Reuters corpus. Minor tokenization differences between the corpora are corrected for.

Both corpora are in the domain of English language news texts, so we expect them to have similar properties. However, they are different corpora and hence have slightly different properties. This means that there are also differences between the training and testing set. We have selected this division to create a more realistic setting. This should allow for a more to real-world use comparison than when both training and testing instances are extracted from the same corpus.

In this article, we will only concentrate on two confusible sets, {*your*, *you 're*} and {*their*, *they 're*, *there*}, that contain alternatives that have different number of tokens. This will not result in an impression on the performance of the language model approach of confusible disambiguation, but it will illustrate the impact the different ways of normalizing the probabilities of the different orders of n-grams.

Table 1 gives an idea of the order of magnitude of the occurrences of different tokens. The numbers indicate the number of occurrences of the tokens and token combinations. The figures for *of the* are provided as comparison. *of the* is the most frequent bi-gram in the corpus.

Looking at the counts of the multi-token alternatives (*you 're* and *they 're*) in table 1, we see that they are much lower compared to the single token alternatives. It may be clear that the uni-gram probabilities (where we take the multi-token alternatives as single tokens) will lead to a clear preference for the single token alternatives. The definition of P' should take this into account and assign more probability mass to the multi-token alternatives.

The results of the different probability computations are shown in table 2. The discount columns shows the count that is subtracted from the total count (which is also shown). The first entry shows the situation where the probability of larger order n-grams are treated as lower order (tri-grams used as bi-grams in this case). The second entry normalizes the total count by subtracting the number of occurrences of the bi-gram,

[2]Memory usage of a suffix tree is larger than that of a simple n-gram model, but it is not prohibitively large. The Reuters corpus, a 4GB plain text corpus, can be stored in a 16GB suffix tree, while optimizations making more efficient use of space are possible.

Total # tokens	132,887,136
# *your*	5,776
# *you 're*	1,935
# *you*	30,965
# *'re*	20,678
# *their*	167,227
# *there*	94,247
# *they 're*	2,672
# *they*	159,290
# *'re*	20,678
# *of the*	569,814
# *of*	2,616,495
# *the*	4,792,464

Table 1: Token counts from the Reuters RVC1 corpus.

Discount method	{*your*, *you 're*}		{*their*, *there*, *they 're*}	
	discount	accuracy	discount	accuracy
No discount	0/132M	79.6	0/132M	64.2
count $-$ occ	2K/132M	79.6	3K/132M	64.2
count $-$ $\min$ (occ(sub-gram))	21K/132M	79.6	21K/132M	64.2
count $-$ $\max$ (occ(sub-gram))	31K/132M	79.6	159K/132M	64.2
count $\times$ $(\frac{\lvert\text{uni-gram}\rvert}{\lvert\text{bi-gram}\rvert})$	127M/132M	79.6	127M/132M	63.7

Table 2: Results of the five different discounting schemes.

effectively using the probability as if the corpus was retokenized, treating the multi-token alternative as a single token. The next two entries reduce the total count by the number of occurrences of the elements of the bi-gram. Finally, discounting is taken into extreme, where for probability computation, the total count is reduced as if all bi-grams were uni-grams. Note that these discounting methods are only used when the focus word contains multiple tokens.

The results clearly show that discounting does not have any effect in classification accuracy. This means that the probabilities used to disambiguate between the different alternatives are extremely robust. Even discounting more than half the total count only has a marginal effect (in the {*their*, *there*, *they 're*} case).

The robustness of the system also indicates that we need to find other ways to improve results. Clearly, the token counts as used here do not easily allow modifications leading to improved results (the probabilities are too robust). Future extensions will require additional information not present in the counts per se.

6 Conclusion

In this article we have proposed several methods to allow alternatives in confusible sets to have an unequal number of tokens. This is essential in confusible sets such as {*your*, *you 're*}. Previous work has solved this in an ad hoc way by retokenizing the corpus, but the newly proposed methods allow for computation based on dynamic recounting, removing the explicit retokenization phase.

By varying the amount of discounting, which is the basis for the new probability metrics, shows that the synchronous back-off method is very robust. Previous research has already shown that synchronous back-off leads to good results.

On the practical side, we presented a suffix tree implementation of language models, allowing us to extract the counts needed for the probability computation of n-grams efficiently. Creation of the suffix tree is linear in corpus size, whereas finding the n-gram information is linear in the search string (the n-gram).

In the future, we would like to be able to deal with annotated tokens (such as part-of-speech), allowing for gradual back-off based on this additional information. The methods described here are essential for incorporating this information. Whereas concatenating tokens and retokenization is still possible, this is

more problematic for part-of-speech information.

Finally, it would be desirable to reduce the memory usage, for instance, using suffix arrays [5, 8] instead of suffix trees. Suffix arrays are known to have similar properties to suffix trees, but using less memory.

References

[1] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33. Association for Computational Linguistics, 2001.

[2] Y. Even-Zohar and D. Roth. A classification approach to word prediction. In *Proceedings of the First North-American Conference on Computational Linguistics*, pages 124–131, New Brunswick, NJ, 2000. ACL.

[3] A. R. Golding. A Bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the 3rd workshop on very large corpora, ACL-95*, 1995.

[4] A.R. Golding and D. Roth. A Winnow-Based Approach to Context-Sensitive Spelling Correction. *Machine Learning*, 34(1–3):107–130, 1999.

[5] Dan Gusfield. *Algorithms on Strings, Trees and Sequences*. University of Cambridge, Cambridge, 1997.

[6] J. H. Huang and D. W. Powers. Large scale experiments on correction of confused words. In *Australasian Computer Science Conference Proceedings*, pages 77–82, Queensland AU, 2001. Bond University.

[7] David D. Lewis, Yiming Yang, Tony G. Rose, G. Dietterich, Fan Li, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[8] Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. In *SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 319–327, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

[9] L. Mangu and E. Brill. Automatic rule acquisition for spelling correction. In *Proceedings of the International Conference on Machine Learning*, pages 187–194, 1997.

[10] M. Marcus, S. Santorini, and M. Marcinkiewicz. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[11] D. Sandra, F. Daems, and S. Frisson. Zo helder en toch zoveel fouten! wat leren we uit psycholinguïstisch onderzoek naar werkwoordfouten bij ervaren spellers? *Tijdschrift van de Vereniging voor het Onderwijs in het Nederlands*, 30(3):3–20, 2001.

[12] Herman Stehouwer and Menno van Zaanen. Language models for contextual error detection and correction. In *Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference*, pages 41–48, Athens, Greece, March 2009. Association for Computational Linguistics.

[13] Esko Ukkonen. On-line construction of suffix trees, 1995.

[14] A. Van den Bosch. Scalable classification-based word prediction and confusible correction. *Traitement Automatique des Langues*, 46(2):39–63, 2006.

[15] A. Van den Bosch and W. Daelemans. *Tussen Taal, Spelling en Onderwijs*, chapter Dat gebeurd mei niet: Computationele modellen voor verwarbare homofonen, pages 199–210. Academia Press, 2007.

[16] D. Wu, Z. Sui, and J. Zhao. An information-based method for selecting feature types for word prediction. In *Proceedings of the Sixth European Conference on Speech Communication and Technology, EUROSPEECH'99*, Budapest, 1999.

[17] D. Yarowsky. Decision lists for lexical ambiguity resolution: application to accent restoration in Spanish and French. In *Proceedings of the Annual Meeting of the ACL*, pages 88–95, 1994.