Language models for contextual error detection and correction

Herman Stehouwer
Tilburg Centre for Creative Computing
Tilburg University
Tilburg, The Netherlands
j.h.stehouwer@uvt.nl

Menno van Zaanen
Tilburg Centre for Creative Computing
Tilburg University
Tilburg, The Netherlands
mvzaanen@uvt.nl

Abstract

The problem of identifying and correcting confusibles, i.e. context-sensitive spelling errors, in text is typically tackled using specifically trained machine learning classifiers. For each different set of confusibles, a specific classifier is trained and tuned.

In this research, we investigate a more generic approach to context-sensitive confusible correction. Instead of using specific classifiers, we use one generic classifier based on a language model. This measures the likelihood of sentences with different possible solutions of a confusible in place. The advantage of this approach is that all confusible sets are handled by a single model. Preliminary results show that the performance of the generic classifier approach is only slightly worse that that of the specific classifier approach.

1 Introduction

When writing texts, people often use spelling checkers to reduce the number of spelling mistakes in their texts. Many spelling checkers concentrate on non-word errors. These errors can be easily identified in texts because they consist of character sequences that are not part of the language. For example, in English *woord* is is not part of the language, hence a non-word error. A possible correction would be *word*.

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 required 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 would be the word *two*. Confusibles are grouped together in confusible sets. Confusible sets are sets of words that are similar and often used incorrectly in context. *Too* is the third alternative in this particular confusible set.

The research presented here is part of a larger project, which focusses on context-sensitive spelling mistakes in general. Within this project all classes of context-sensitive spelling errors are tackled. For example, in addition to confusibles, a class of pragmatically incorrect words (where words are incorrectly used within the document-wide context) is considered as well. In this article we concentrate on the problem of confusibles, where the context is only as large as a sentence.

2 Approach

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*). However, some research has also touched upon inflectional or derivational confusibles such as *I* versus *me* (Golding and Roth, 1999). 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) (Sandra et al., 2001; Van den Bosch and Daelemans, 2007).

Most work on confusible disambiguation using machine learning concentrates on hand-selected sets of notorious confusibles. The confusible sets are typically 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 approaches used in accent restoration (Yarowsky, 1994; Golding, 1995; Mangu and Brill, 1997; Wu et al., 1999; Even-Zohar and Roth, 2000; Banko and Brill, 2001; Huang and Powers, 2001; Van den Bosch, 2006).

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*.

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 problem of selecting the best alternative from confusible sets. We will investigate this approach in this article.

Language models assign probabilities to sequences of words. Using this information, it is possible to predict the most likely word in a certain context. If a language model gives us the probability for a sequence of n words $P_{LM}(w_1, \ldots, w_n)$, we can use this to predict the most likely word w following a sequence of $n-1$ words $\arg\max_w P_{LM}(w_1, \ldots, w_{n-1}, w)$. Obviously, a similar approach can be taken with w in the middle of the sequence.

Here, we will use a language model as a classifier to predict the correct word in a context. Since a language model models the entire language, it is different from a regular machine learning classifier trained on a specific set of confusibles. The advantage of this 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 according to the language model is then selected. Since the language model assigns 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 of a specific confusible set.

The expected disadvantage of the generic (language model) classifier approach is that the accuracy is expected to be less than that of the specific (specialized machine learning classifier) approach. 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 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.

In this article, we will investigate whether it is possible to build a confusible disambiguation system that is generic for all sets of confusibles using language models as generic classifiers and investigate in how far this approach is useful for solving the confusible problem. We will compare these generic classifiers against specific classifiers that are trained for each confusible set independently.

3 Results

To measure the effectiveness of the generic classifier approach to confusible disambiguation, and to compare it against a specific classifier approach we have implemented several classification systems. First of these is a majority class baseline system, which selects the word from the confusible set that occurs most often in the training data.[1] We have also implemented several generic classifiers based on different language models. We compare these against two machine learning classifiers. The machine learning classifiers are trained separately for each different experiment, whereas

[1] This baseline system corresponds to the simplest language model classifier. In this case, it only uses n-grams with $n = 1$.

the parameters and the training material of the language model are kept fixed throughout all the experiments.

3.1 System description

There are many different approaches that can be taken to develop language models. A well-known approach is to use n-grams, or Markov models. These models take into account the probability that a word occurs in the context of the previous $n-1$ words. The probabilities can be extracted from the occurrences of words in a corpus. Probabilities are computed by taking the relative occurrence count of the n words in sequence.

In the experiments described below, we will use a tri-gram-based language model and where required this model will be extended with bi-gram and uni-gram language models. The probability of a sequence is computed as the combination of the probabilities of the tri-grams that are found in the sequence.

Especially when n-grams with large n are used, data sparseness becomes an issue. The training data may not contain any occurrences of the particular sequence of n symbols, even though the sequence is correct. In that case, the probability extracted from the training data will be zero, even though the correct probability should be non-zero (albeit small). To reduce this problem we can either use back-off or smoothing when the probability of an n-gram is zero. In the case of back-off, the probabilities of lower order n-grams are taken into account when needed. Alternatively, smoothing techniques (Chen and Goodman, 1996) redistribute the probabilities, taking into account previously unseen word sequences.

Even though the language models provide us with probabilities of entire sequences, we are only interested in the n-grams directly around the confusible when using the language models in the context of confusible disambiguation. The probabilities of the rest of the sequence will remain the same whichever alternative confusible is inserted in the focus word position. Figure 1 illustrates that the probability of for example $P(\textit{analysts had expected})$ is irrelevant for the decision between *then* and *than* because it occurs in both sequences.

The different language models we will consider here are essentially the same. The differences lie in how they handle sequences that have zero probability. Since the probabilities of the n-grams are multiplied, having a n-gram probability of zero results in a zero probability for the entire sequence. There may be two reasons for an n-gram to have probability zero: there is not enough training data, so this sequence has not been seen yet, or this sequence is not valid in the language.

When it is known that a sequence is not valid in the language, this information can be used to decide which word from the confusible set should be selected. However, when the sequence simply has not been seen in the training data yet, we cannot rely on this information. To resolve the sequences with zero probability, we can use smoothing. However, this assumes that the sequence is valid, but has not been seen during training. The other solution, back-off, tries not to make this assumption. It checks whether subsequences of the sequence are valid, i.e. have non-zero probabilities. Because of this, we will not use smoothing to reach non-zero probabilities in the current experiments, although this may be investigated further in the future.

The first language model that we will investigate here is a linear combination of the different n-grams. The probability of a sequence is computed by a linear combination of weighted n-gram probabilities. We will report on two different weight settings, one system using uniform weighting, called *uniform linear*, and one where unigrams receive weight 1, bi-grams weight 138, and tri-grams weight 437.[2] These weights are normalized to yield a final probability for the sequence, resulting in the second system called *weighted linear*.

The third system uses the probabilities of the different n-grams separately, instead of using the probabilities of all n-grams at the same time as is done in the linear systems. The *continuous back-off* method uses only one of the probabilities at each position, preferring the higher-level probabilities. This model provides a step-wise back-off. The probability of a sequence is that of the tri-grams contained in that sequence. However, if the probability of a trigram is zero, a back-off to the probabilities of the two bi-grams of the sequence is used. If that is still zero, the uni-gram probability at that position is used. Note that this uni-gram probability is exactly what the baseline system

[2] These weights are selected by computing the accuracy of all combinations of weights on a held out set.

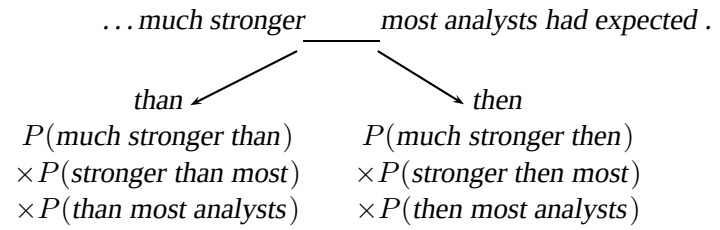

Figure 1: Computation of probabilities using the language model.

uses. With this approach it may be the case that the probability for one word in the confusible set is computed based on tri-grams, whereas the probability of another word in the set of confusibles is based on bi-grams or even the uni-gram probability. Effectively, this means that different kinds of probabilities are compared. The same weights as in the weighted linear systems are used.

To resolve the problem of unbalanced probabilities, a fourth language model, called *synchronous back-off*, is proposed. Whereas in the case of the continuous back-off model, two words from the confusible set may be computed using probabilities of different level n-grams, the synchronous back-off model uses probabilities of the same level of n-grams for all words in the confusible set, with n being the highest value for which at least one of the words has a non-zero probability. For instance, when word a has a tri-gram probability of zero and word b has a non-zero tri-gram probability, b is selected. When both have a zero tri-gram probability, a back-off to bi-grams is performed for both words. This is in line with the idea that if a probability is zero, the training data is sufficient, hence the sequence is not in the language.

To implement the specific classifiers, we used the TiMBL implementation of a k-NN classifier (Daelemans et al., 2007). This implementation of the k-NN algorithm is called *IB1*. We have tuned the different parameter settings for the k-NN classifier using Paramsearch (Van den Bosch, 2004), which resulted in a k of 35.[3] To describe the instances, we try to model the data as similar as possible to the data used by the generic classifier approach. Since the language model approaches use n-grams with $n = 3$ as the largest n, the features for the specific classifier approach use words one and two positions left and right of the focus word. The focus word becomes the class that needs to be predicted. We show an example of both training and testing in figure 2. Note that the features for the machine learning classifiers could be expanded with, for instance, part-of-speech tags, but in the current experiments only the word forms are used as features.

In addition to the k-NN classifier, we also run the experiments using the IGTree classifier, which is denoted *IGTree* in the rest of the article, which is also contained in the TiMBL distribution. IGTree is a fast, trie based, approximation of k-nearest neighbor classification (Knuth, 1973; Daelemans et al., 1997). IGTree allows for fast training and testing even with millions of examples. IGTree compresses a set of labeled examples into a decision tree structure similar to the classic C4.5 algorithm (Quinlan, 1993), except that throughout one level in the IGTree decision tree, the same feature is tested. Classification in IGTree is a simple procedure in which the decision tree is traversed from the root node down, and one path is followed that matches the actual values of the new example to be classified. If a leaf is found, the outcome stored at the leaf of the IGTree is returned as the classification. If the last node is not a leaf node, but there are no outgoing arcs that match a feature-value combination of the instance, the most likely outcome stored at that node is produced as the resulting classification. This outcome is computed by collating the outcomes of all leaf nodes that can be reached from the node.

IGTree is typically able to compress a large example set into a lean decision tree with high compression factors. This is done in reasonably short time, comparable to other compression algorithms. More importantly, IGTree's classification time depends only on the number of features ($O(f)$). Indeed, in our experiments we observe high compression rates. One of the unique characteristics of IGTree compared to basic k-NN is its resemblance to smoothing of a basic language

[3] We note that k is handled slightly differently in TiMBL than usual, k denotes the number of closest distances considered. So if there are multiple instances that have the same (closest) distance they are all considered.

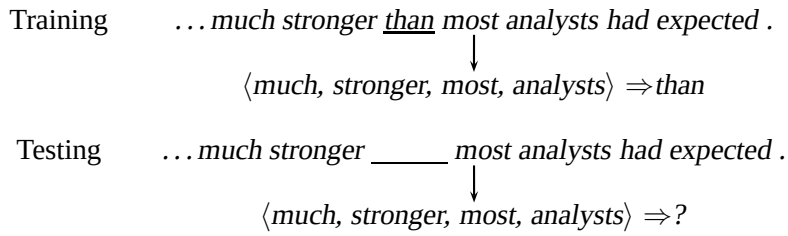

Figure 2: During training, a classified instance (in this case for the confusible pair {*then*, *than*}) are generated from a sentence. During testing, a similar instance is generated. The classifier decides what the corresponding class, and hence, which word should be the focus word.

model (Zavrel and Daelemans, 1997), while still being a generic classifier that supports any number and type of features. For these reasons, IGTree is also included in the experiments.

3.2 Experimental settings

The probabilities used in the language models of the generic classifiers are computed by looking at occurrences of n-grams. These occurrences are extracted from a corpus. The training instances used in the specific machine learning classifiers are also extracted from the same data set. For training purposes, we used the Reuters news corpus RCV1 (Lewis et al., 2004). 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.

For testing purposes, we used the Wall Street Journal part of the Penn Treebank corpus (Marcus et al., 1993). 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. There are minor tokenization differences between the corpora. The data is corrected for these differences.

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 are slightly different. 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.

For the specific experiments, we selected a number of well-known confusible sets to test the different approaches. In particular, we look at {*then, than*}, {*its, it's*}, {*your, you're*}, {*their, there, they're*}. To compare the difficulty of these problems, we also selected two words at random and used them as a confusible set.

The random category consists of two words that where randomly selected from all words in the Reuters corpus that occurred more than a thousand times. The words that where chosen, and used for all experiments here are *refugees* and *effect*. They occur around 27 thousand times in the Reuters corpus.

3.3 Empirical results

Table 1 sums up the results we obtained with the different systems. The baseline scores are generally very high, which tells us that the distribution of classes in a single confusible set is severely skewed, up to a ten to one ratio. This also makes the task hard. There are many examples for one word in the set, but only very few training instances for the other(s). However, it is especially important to recognize the important aspects of the minority class.

The results clearly show that the specific classifier approaches outperform the other systems. For instance, on the first task ({*then, than*}) the classifier achieves an accuracy slightly over 98%, whereas the language model systems only yield around 96%. This is as expected. The classifier is trained on just one confusible task and is therefore able to specialize on that task.

Comparing the two specific classifiers, we see that the accuracy achieved by IB1 and IGTree is quite similar. In general, IGTree performs a bit worse than IB1 on all confusible sets, which is as expected. However, in general it is possible for IGTree to outperform IB1 on certain tasks. In our experience this mainly happens on tasks where the usage of IGTree, allowing for more compact internal representations, allows one to use much more training data. IGTree also leads to improved

	{*then, than*}	{*its, it's*}	{*your, you're*}	{*their, there, they're*}	random
Baseline	82.63	92.42	78.55	68.36	93.16
IB1	98.01	98.67	96.36	97.12	97.89
IGTree	97.07	96.75	96.00	93.02	95.79
Uniform linear	68.27	50.70	31.64	32.72	38.95
Weighted linear	94.43	92.88	93.09	93.25	88.42
Continuous back-off	81.49	83.22	74.18	86.01	63.68
Synchronous back-off	96.42	94.10	92.36	93.06	87.37
Number of cases	2,458	4,830	275	3,053	190

Table 1: This table shows the performance achieved by the different systems, shown in accuracy (%). The *Number of cases* denotes the number of instances in the testset.

performance in cases where the features have a strong, absolute ordering of importance with respect to the classification problem at hand.

The generic language model approaches perform reasonably well. However, there are clear differences between the approaches. For instance the weighted linear and synchronous back-off approaches work well, but uniform linear and continuous back-off perform much worse. Especially the synchronous back-off approach achieves decent results, regardless of the confusible problem.

It is not very surprising to see that the continuous back-off method performs worse than the synchronous back-off method. Remember that the continuous back-off method always uses lower level n-grams when zero probabilities are found. This is done independently of the probabilities of the other words in the confusible set. The continuous back-off method prefers n-grams with larger n, however it does not penalize backing off to an n-gram with smaller n. Combine this with the fact that n-gram probabilities with large n are comparatively lower than those for n-grams with smaller n and it becomes likely that a bi-gram contributes more to the erroneous option than the correct tri-gram does to the correct option. Tri-grams are more sparse than bi-grams, given the same data.

The weighted linear approach outperforms the uniform linear approach by a large margin on all confusible sets. It is likely that the contribution from the n-grams with large n overrules the probabilities of the n-grams with smaller n in the uniform linear method. This causes a bias towards the more frequent words, compounded by the fact that bi-grams, and uni-grams even more so, are less sparse and therefore contribute more to the total probability.

We see that the both generic and specific classifier approaches perform consistently across the different confusible sets. The synchronous back-off approach is the best performing generic classifier approach we tested. It consistently outperforms the baseline, and overall performs better than the weighted linear approach.

The experiments show that generic classifiers based on language model can be used in the context of confusible disambiguation. However, the n in the different n-grams is of major importance. Exactly which n grams should be used to compute the probability of a sequence requires more research. The experiments also show that approaches that concentrate on n-grams with larger n yield more encouraging results.

4 Conclusion and future work

Confusibles are spelling errors that can only be detected within their sentential context. This kind of errors requires a completely different approach compared to non-word errors (errors that can be identified out of context, i.e. sequences of characters that do not belong to the language). In practice, most confusible disambiguation systems are based on machine learning classification techniques, where for each type of confusible, a new classifier is trained and tuned.

In this article, we investigate the use of language models in the context of confusible disambiguation. This approach works by selecting the word in the set of confusibles that has the highest probability in the sentential context according to the language model. Any kind of language model can be used in this approach.

The main advantage of using language models as generic classifiers is that it is easy to add new sets of confusibles without retraining or adding additional classifiers. The entire language is mod-

eled, which means that all the information on words in their context is inherently present.

The experiments show that using generic classifiers based on simple n-gram language models yield slightly worse results compared to the specific classifier approach, where each classifier is specifically trained on one confusible set. However, the advantage of the generic classifier approach is that only one system has to be trained, compared to different systems for each confusible in the specific classifier case. Also, the exact computation of the probabilities using the n-grams, in particular the means of backing-off, has a large impact on the results.

As future work, we would like to investigate the accuracy of more complex language models used as classifiers. The n-gram language models described here are relatively simple, but more complex language models could improve performance. In particular, instead of back-off, smoothing techniques could be investigated to reduce the impact of zero probability problems (Chen and Goodman, 1996). This assumes that the training data we are currently working with is not enough to properly describe the language.

Additionally, language models that concentrate on more structural descriptions of the language, for instance, using grammatical inference techniques (de la Higuera, 2005), or models that explicitly take long distance dependencies into account (Griffiths et al., 2005) can be investigated. This leads to much richer language models that could, for example, check whether there is already a verb in the sentence (which helps in cases such as {*its, it's*}).

A different route which we would also like to investigate is the usage of a specific classifier, such as TiMBL's IGTree, as a language model. If a classifier is trained to predict the next word in the sentence or to predict the word at a given position with both left and right context as features, it can be used to estimate the probability of the words in a confusible set, just like the language models we have looked at so far. Another type of classifier might estimate the perplexity at a position, or provide some other measure of "surprisedness". Effectively, these approaches all take a model of the entire language (as described in the training data) into account.

References

Banko, M. and Brill, E. (2001). 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.

Chen, S. and Goodman, J. (1996). An empirical study of smoothing techniques for language modelling. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 310–318. ACL.

Daelemans, W., Van den Bosch, A., and Weijters, A. (1997). IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

Daelemans, W., Zavrel, J., Van der Sloot, K., and Van den Bosch, A. (2007). TiMBL: Tilburg Memory Based Learner, version 6.1, reference guide. Technical Report ILK 07-07, ILK Research Group, Tilburg University.

de la Higuera, C. (2005). A bibliographical study of grammatical inference. *Pattern Recognition*, 38(9):1332 – 1348. Grammatical Inference.

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

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

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

Griffiths, T. L., Steyvers, M., Blei, D. M., and Tenenbaum, J. B. (2005). Integrating topics and syntax. In *In Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press.

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

Knuth, D. E. (1973). *The art of computer programming*, volume 3: Sorting and searching. Addison-Wesley, Reading, MA.

Lewis, D. D., Yang, Y., Rose, T. G., Dietterich, G., Li, F., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

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

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

Quinlan, J. (1993). C4.5*: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Sandra, D., Daems, F., and Frisson, S. (2001). 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.

Van den Bosch, A. (2004). Wrapped progressive sampling search for optimizing learning algorithm parameters. In Verbrugge, R., Taatgen, N., and Schomaker, L., editors, *Proceedings of the Sixteenth Belgian-Dutch Conference on Artificial Intelligence*, pages 219–226, Groningen, The Netherlands.

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

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

Wu, D., Sui, Z., and Zhao, J. (1999). 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.

Yarowsky, D. (1994). 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.

Zavrel, J. and Daelemans, W. (1997). Memory-based learning: Using similarity for smoothing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 436–443.