

Putting the *t* where it belongs: Solving a confusion problem in Dutch

Herman Stehouwer and Antal van den Bosch

Tilburg centre for Creative Computing, Tilburg University
{J.H.Stehouwer,Antal.vdnBosch}@uvt.nl

Abstract

A common Dutch writing error is to confuse a word ending in *-d* with a neighbor word ending in *-dt*. In this paper we describe the development of a machine-learning-based disambiguator that can determine which word ending is appropriate, on the basis of its local context. We develop alternative disambiguators, varying between a single monolithic classifier and having multiple confusable experts disambiguate between confusable pairs. Disambiguation accuracy of the best developed disambiguators exceeds 99%; when we apply these disambiguators to an external test set of collected errors, our detection strategy correctly identifies up to 79% of the errors.

1 Introduction

Learners of languages with alphabetic writing systems and overt morphology at some point face the task of learning the language’s derivational and inflectional paradigms. Mastering these paradigms is the key to producing grammatical sentences, with all the proper agreements and the proper choices of morpho-syntactic categories and their associated inflections. An important subtask to master is to choose the contextually appropriate form out of a set of paradigmatic alternatives. In the sentence “*He smiled broadly*”, the writer has to decide the proper forms are *He*, not *Him*; *smiled*, not another inflection of the same verb; and the adverbial form *broadly*, not the adjective *broad*. This task can be all the harder when the learner is a second-language learner whose native language uses different paradigms. A related task is faced by natural language generation systems, which need to generate the appropriate forms after having made their lexical choices.

In this paper we describe the development of a grammar checking module that could be integrated into a larger proofing system. We exemplify the approach using a notorious problem of paradigmatic choice in Dutch that takes learners some time to master. The problem, in its core a verb inflection task, still manages to confuse even experienced writers, witnessed by the many errors found on webpages, emails, and student reports. It serves as an illustration; as we will argue, the approach is easily transportable to similar morphological-paradigmatic confusability problems.

Making a forced choice in paradigmatic derivation or inflection, given a sentential context, can be straightforwardly cast as a classification task, which can be learned by a machine learning algorithm – much like selecting the appropriate sense for a polysemous word in word sense disambiguation systems. If a classifier can be trained to make the choice with zero error, then this classifier could be used as a grammar checker, applicable to any unseen text that contains a variant of the

derivation or inflection the classifier is trained on. On the basis of the context, the perfect classifier could decide on which form is appropriate, given the context; if the classifier's prediction contradicts the form actually used, then the form could be flagged as a grammatical error.

Taking this route, we distinguish two views on the same problem, which lead to somewhat different machine learning experiments. From one perspective the task can be seen as a morphological generation task in which, given a context, the appropriate morphological operation needs to be triggered. For example, in deciding whether it should be *word* or *wordt* in *hij [word] thuisgebracht (he is (being) brought home)*, a machine learner would be faced with the particular context *Hij . . . thuisgebracht*, the unresolved word *[word]*, and the task to classify this situation. Although this definition of the task seems straightforward (it does not presuppose any linguistic analysis of the data, as it operates just on surface cues found in wordforms), and the type of task appears quite central to natural language generation, there is not an abundance of literature approaching this task in this knowledge-free way. In (Yarowsky and Wicentowski 2000) a successful “minimally supervised” morphological analyzer is introduced which also works well on highly irregular forms. Similar work is described in (Schone and Jurafsky 2001), who focus on the inflection system of Dutch, English, and German. More generally, our definition is related in spirit to machine-learning approaches to morphological generation tasks at the word level, such as in past tense generation with English verbs (Mooney and Califf 1995) or diminutive inflection on Dutch nouns (Daelemans et al. 1997b), except that our predictive features come from the neighboring sentential context, not from the word itself.

Alternatively, and this is our second perspective, the task can be seen as a multitude of pairwise confusable disambiguation tasks, where each pair of alternatives is the domain of one classifier, henceforth *confusable expert*. For instance, there would be a separate confusable expert for deciding whether the word *houd* or the word *houdt* would be appropriate in a given context. This is in essence the same two-class task as the first formulation, but a machine learner performing one particular confusable disambiguation will only see training examples of the two alternate wordforms. Defined as such, this definition joins a long list of earlier approaches to context-based confusable set disambiguation or the related issue of accent restoration (Yarowsky 1994, Golding 1995, Mangu and Brill 1997, Wu et al. 1999, Even-Zohar and Roth 2000, Huang and Powers 2001, Banko and Brill 2001, Van den Bosch 2006). Most of this work has concentrated on a hand-selected set of notorious confusables due to homophony (*to, too, two*) or similar spelling (*desert, dessert*). Yet, some of it has also touched upon inflectional or derivational confusables such as *I* versus *me* (Golding and Roth 1999), the type of task we also target.

The novelty of the approach presented here resides in the fact that we combine the two approaches in order to achieve optimal results. The key difference between the two perspectives is the specificity of the classifiers. It can be expected that the monolithic classifier can leverage from the massive amounts of examples it can be trained on, while the confusable experts may draw some advantage out

of the fact that there may be specific lexical contextual markers for individual confusables (such as particular verbs that a confusable adverbial form will typically follow) that would go unnoticed by the monolithic classifier. By combining the two approaches, a best-of-both-worlds solution may be reached.

A powerful help to both approaches, and a strong point for the reusability of the approach in general, is that large amounts of labeled examples can be gathered freely from digital text corpora, without the need for annotation. The very occurrence of any word belonging to a confusable pair is effectively an example in context, labeled with one of the two possible outcomes. The only disadvantage of this “free lunch” is that if a text corpus in fact contains a writing error (e.g. a text may contain the incorrect *hij *word thuisgebracht*), this example will either turn up as a falsely labeled training example, potentially causing the resulting classifier to err later on, or as a falsely labeled test example, causing errors in statistics of false hits and misses. We will not be able to solve this, nor measure the size of the effect of this hidden noise. However, this hidden factor does not invalidate the target of our study, which is to measure the error detection capabilities of our trained classifiers on actual, manually collected cases of confusion.

The latter is the reason why our evaluation focuses on two aspects: (1) First we measure how accurately our classifiers can decide between the two outcomes of the task – at this point, we ignore whether our test data might actually contain corpus errors; (2) Second, we also confront our best-performing classifiers with pre-compiled lists of confusable errors in context, collected on the web, to see what portion of these errors are actually recognized as such, providing a sample-based error detection recall estimate.

It is unlikely that we would be able to develop perfect classifiers, but low error rates should be attainable. The goal of this paper is to test whether classifiers can be trained to classify at such low error rates that they can detect confusions well beyond the baseline level.

This paper is structured as follows. First, in Section 2 we introduce the task in some detail. Section 3 describes the systems developed. Section 4 reports on the experiments we performed; the results of the experiments are given in Section 5, and their wider implications are discussed in Section 7.

2 The confusion of Dutch words ending in *-d* versus *-dt*

The frequently occurring confusion of words ending in *-d* or *-dt* in Dutch is rooted in a verbal inflection choice, but extends to other forms as well. We first focus on the verbal inflection issue. A notorious problem in Dutch is the normal singular present tense *-t* inflection of verbs of which the stem ends in *-d*. The problematic issue is that the inflected form is pronounced the same as the form without the *-t*, as word-final *-d* and *-dt* are both devoiced and pronounced as */t/* in Dutch. So, the frequent verbs *word* and *wordt* (*become* and *becomes*) are both pronounced */wOrt/*. When wordforms are homophonic, they tend to get confused often in writing (cf. the situation with *to*, *too*, and *two*, or *there*, *their*, and *they're* in English) (Sandra et al. 2001, Van den Bosch and Daelemans 2007).

As a rule, the singular second-person and third-person forms of Dutch present-tense verbs receive a *-t* inflection: *ik loop* (I walk), *jij loopt* (you walk), *zij loopt* (she walks). Besides exceptions with high-frequency irregular verbs such as *zijn* (to be), an important subregularity is that when the second-person subject occurs after the verb, e.g. in questions, the *-t* inflection is not realized (compare *jij loopt* – you walk, to *loop jij?* – do you walk?) (Geerts et al. 1984).

Although this verbal inflection issue lies at the root of the *-d* versus *-dt* confusion problem, it also affects words with the same form that are not present-tense verbs. Verb forms ending in *-d* often occur with that same form as a singular noun (e.g. *strand* can occur as a noun, e.g. *het strand* – the beach, or as a verb, *ik strand* – I get stuck), and in a similar vein present-tense verbs in *-d* can have the same form as the past participle form of the same verb (e.g. *ik verwoord het* – I phrase it, versus *ik heb het verwoord* – I phrased it). Due to the homophonic confusion and due to cognitive errors of language learners and writers in general, also these forms may occasionally get an incorrect *-dt* ending.

3 System architecture

3.1 Example generation

We collect examples from large unannotated tokenized texts by (1) finding all words ending in *-dt*, and (2) finding all corresponding alternatives having the same character string up to *-dt*, ending in *-d*. We did not filter out nouns and past participles, to keep the method as general and knowledge-free as possible, as argued in the previous section.

For our experiments on the Dutch task we use the Twente News Corpus¹, which contains about 365 million tokens. The corpus is part-of-speech tagged with a state-of-the-art tagger for Dutch² at an estimated accuracy of 96.5%. Alternatively, the corpus is also tagged with unsupervised part-of-speech tags generated with Biemann’s Chinese Whispers algorithm (Biemann 2007) on which an MBT tagger was trained³. In the experiments described in more detail in Section 4, we compare three experimental conditions in which we either do not use tags, or use Biemann’s unsupervised tags, or the tags generated by the language-specific part-of-speech tagger. In the corpus we find 2,975,185 examples of *-d* versus *-dt* wordforms. Words ending in *-d* are in the majority (65%), which is partly due to the fact that the matching nouns and past participle forms all end in *-d*.

To turn the words in their sentential contexts into proper machine learning examples, a window with a limited focus of three words to the left, and three words to the right is imposed on the sentence, centering on the focus word. The focus word itself, containing the answer to the problem, is necessarily masked; we remove the information to be predicted by reducing the focus word to the shorter of the two forms. Figure 1 illustrates the conversion of an occurrence of a Dutch confus-

¹<http://wwwhome.cs.utwente.nl/~druid/TwNC/TwNC-main.html>

²<http://ilk.uvt.nl/tadpole>

³See <http://ilk.uvt.nl/mbt/> – MBT is also the tagger used in Tadpole.

Koningin Beatrix **wordt** vrijdag 70 jaar.

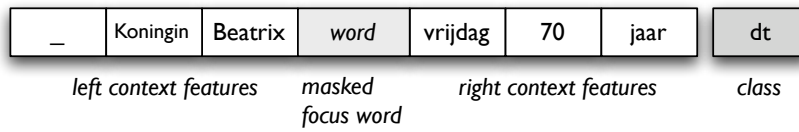


Figure 1: Generation of one Dutch windowed and labeled example.

able in context, to labeled examples. In the experimental conditions in which we also include part-of-speech information, the tags of all context words in the input window are included. However, the tag of the focus word is masked, as the focus tag reveals the outcome. Any feature that would unequivocally vote for the class suggested by the focus wordform would be counterproductive in a system that is to be used to detect that the focus wordform is in fact contextually inappropriate.

In order to be able to extract these local contexts we must first identify which words are part of the confusable problem. We used the following simple rule to build lists of confusable pairs: We select all words of the form $[stem-d]t$ for which we also find at least one occurrence of the word $[stem-d]$.

3.2 Combining classifiers

As our end goal is to develop a maximally accurate disambiguator for the two confusable tasks in order to detect actual writing errors, the question is which of the possible classifier architectures would be the most accurate. The most extreme outcomes may be on the one hand that the classifier trained on the generic morphological generation task is the best, possibly because it can be trained on most examples. On the other hand, it may turn out that an ensemble of all individual confusable experts offers the best performance. A downside of the latter architecture is that this would involve an architecture with thousands of classifiers. We decided to explore the space of combinations of generic classification with selections of word experts, under the assumption that there is an optimal selection of confusable experts that can be combined with a more generic classifier.

These ensemble systems act as gating systems: if the test word to be checked against the prediction of the system is handled by one of the selected confusable experts, the case is given to be classified to the appropriate expert; otherwise, it is passed on to be classified by the morphological generator, acting as a back-off classifier. Given such a gating architecture, the monolithic classifier can be trained in two ways: first, using all available training examples including the ones also handled by the selected confusable experts; and second, by using all training examples except those used by the specific confusable experts. The architecture of this system is visualized in a flowchart in Figure 2.

An important variable in this architecture is the criterion for selecting the con-

fusable experts. In our experiments, the performance of the confusable experts in terms of error rate on 10-fold cross-validation experiments on the training set determines their selection. If an expert performs better on average than the generic morphological generator on the same 10-fold cross-validation experiment, it is selected.

4 Experimental setup

To provide training and test examples, we randomly shuffle all sentences in the corpus, and then divide them over a 90% training corpus, and a 10% test corpus. Our main experiment involves the disambiguation of all cases of *-d* vs. *-dt* words, in the 10% test corpus, by the ensemble system with automatically selected confusable experts. We compare the ensemble system against three simpler systems:

1. A baseline system that predicts the overall most likely outcome; this amounts to always predicting *-d*;
2. A confusable expert baseline system that predicts for each confusable expert the most likely outcome, which may deviate from the overall most likely outcome (analogous to the “most frequent sense” baseline in word sense disambiguation);
3. The monolithic classifiers in isolation, i.e., the ensemble system without the confusable experts.

While the monolithic classifier has a total of 2,646,369 cases to train on (the number of *-dt* and *-d* tokens in the training corpus), some of the confusable experts will have but a handful of examples. The most frequent confusable pair, *word* versus *wordt* (*become* vs. *becomes*) is represented by 679,808 training examples.

The ensemble system (cf. Figure 2) offers one aforementioned experimental option that we decided to vary systematically, which is whether the monolithic classifier retains the examples of the confusable experts or not (henceforth referred to as the *Keep* and \neg *Keep* variants). If a confusable expert is selected in cross validation, it could be argued that the monolithic classifier does not need to be trained on examples of the particular word pair covered by the selected confusable expert anymore, since the monolithic classifier will not deal with that particular word pair. On the other hand, it could be argued that the monolithic classifier should keep all examples including those already covered by the selected confusable experts, since they might turn out useful training examples also for other words.

Adding to the *Keep* versus \neg *Keep* variants, we expand the experiments with the ensemble systems by three variants of using part-of-speech information, resulting in a 2×3 matrix of experiments. In the first variant, language-specific part-of-speech tags are included; in the third, unsupervised tags generated by Biemann’s Chinese Whispers (Biemann 2007) are incorporated in the feature vector. In the third variant, the windowed examples do not include part-of-speech information at all (such as depicted in Figure 1).

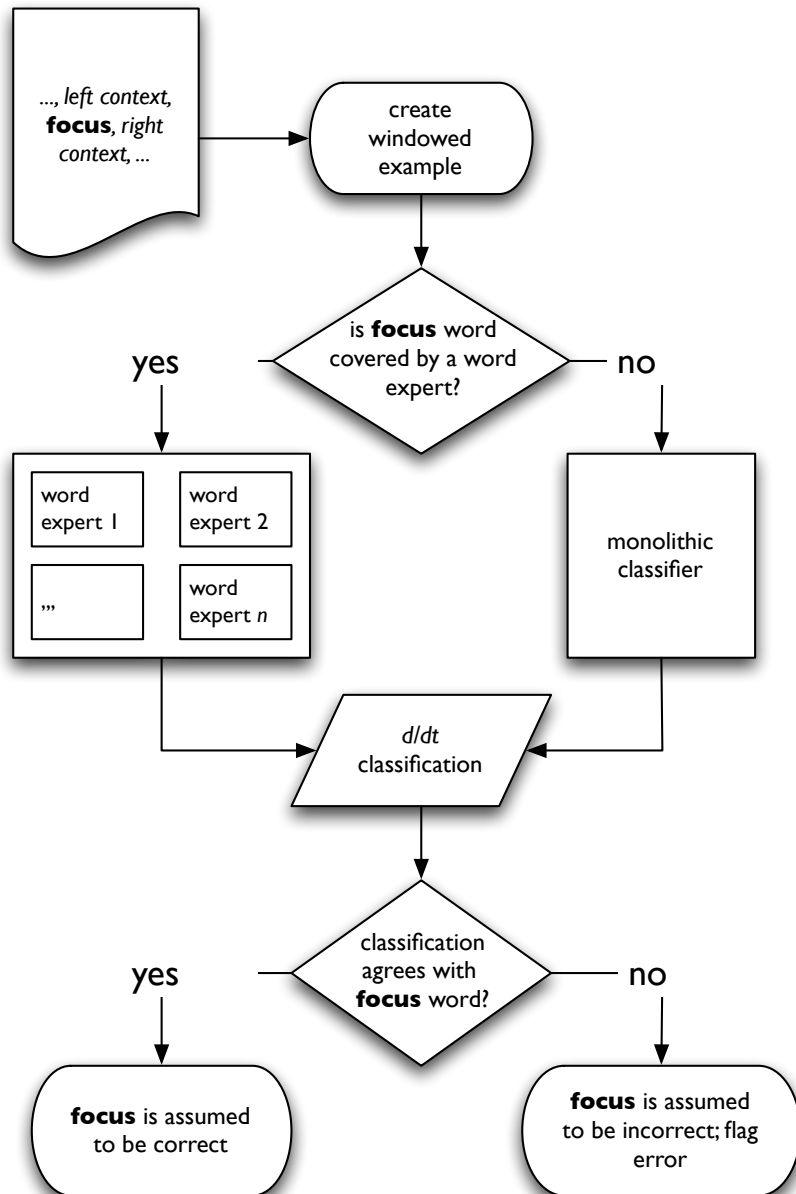


Figure 2: Flowchart of the grammar checking process with the ensemble system of confusable experts and the monolithic classifier. A potentially confused word in context, encoded as a windowed example, is classified by either a confusable expert or the monolithic classifier. If the classification does not agree with the original wordform in focus, an error is flagged.

| Baseline error | System | POS tagging | | |
|----------------|------------|-------------|--------------|-------------|
| | | supervised | unsupervised | no POS tags |
| 34.70 | Monolithic | 0.83 | 2.07 | 2.34 |
| 34.70 | Keep | 0.89 | 1.65 | 2.00 |
| 34.70 | ¬Keep | 0.94 | 2.08 | 2.00 |

Table 1: Baseline and system error rates (%) of the monolithic classifier and the ensemble classifiers with the monolithic classifier retaining the examples of the selected confusable experts (“Keep”) or not, under three POS tagging conditions. Boldfaced results mark the lowest error rate.

To allow for fast training and testing even with millions of examples, we used IGTre, a fast approximation of k -nearest neighbor classification (Daelemans et al. 1997a), as the core classifier in our experiments. IGTre 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 IGTre decision tree, the same feature is tested. Classification in IGTre 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 an end node is met, the outcome stored at the end node is generated as classification. If the last visited node is a non-ending node, but no outgoing arcs match with the next value to be tested, the most likely outcome stored at that last visited node is produced as the resulting classification.

IGTre is typically able to compress a large example set into a lean decision tree with high compression factors, in reasonably short time, comparable to other compression algorithms. More importantly, IGTre’s classification time depends only on the number of features ($O(f)$). Indeed, we observe high compression rates: trained on almost 2 million examples of the task, IGTre builds a tree containing a mere 46,466 nodes, with which it can classify 17 thousand examples per second on a current standard computing server.

5 Results

We evaluate the trained classifiers in several ways. First, we measure how accurately our classifiers can decide between the two outcomes of the task on the test set given a windowed local context. Second, we measure the capacity of our systems to detect errors in manually gathered list of confusions found on the web.

5.1 Disambiguation Error Rates

In order to ground our results, we first establish a naive baseline based on the majority outcome. The majority class baseline obtains an error rate of 34.79%, corresponding with the majority class occurring with 65.21% of all test instances.

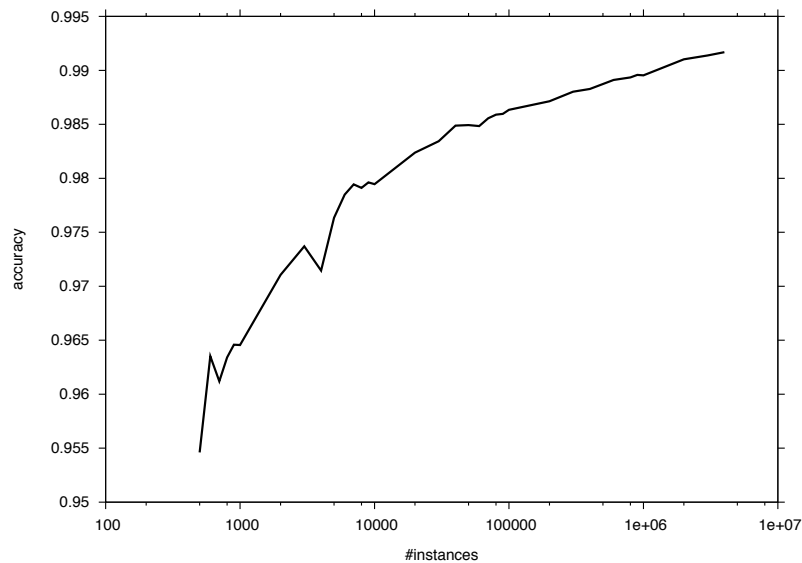


Figure 3: Learning curve for the monolithic classifier using the supervised POS tags as features. We plot the # of training instances used against the accuracy achieved. The x-axis is logarithmic.

The less naive baseline (mentioned to the left of “Keep” and “–Keep” in the same table) consists of a majority classifier for each word pair. The obtained error rate of this baseline is almost the same as that of the simpler baseline, however it is included here because, as we shall show later, it performs differently on the collected confusions.

Table 1 also reports the error rates achieved by the monolithic classifier in the line denoted with *Monolithic*. The monolithic classifier manages to attain an error rate of only 0.83% using supervised part-of-speech tags. This result is markedly better than the result obtained by the baseline (34.79%). Figure 3 shows the learning curve of this classifier. We can see that although classification accuracy improvement is slowing down with more training data, it would still continue to improve if we had more training data. The results of the monolithic classifier when using unsupervised part-of-speech tags or no part-of-speech tags are worse than the results with supervised part-of-speech tags (2.07% and 2.34% respectively). Still, they outperform the baseline by a wide margin.

Table 1 further displays the error rates measured for all of the 2×3 experiments, varying whether confusable expert data was kept or not (*Keep* vs. *–Keep*),

| Word | # of instances | Improvement (%) |
|--------------|----------------|-----------------|
| onderhoud(t) | 7040 | .4112 |
| spoed(t) | 1083 | .1994 |
| verbreed(t) | 394 | .1221 |
| ophoud(t) | 950 | .1198 |
| aftreed(t) | 255 | .1189 |
| strand(t) | 5359 | .1050 |
| vasthoud(t) | 884 | .1028 |
| aanmeld(t) | 163 | .0963 |
| bijhoud(t) | 238 | .0955 |
| vergoed(t) | 1524 | .0880 |

Table 2: The top-10 better performing confusable pairs for the combination system that keeps the examples for the confusable experts also as training material for the monolithic classifier. We list the word, the number of instances it has in the training set, and the average improvement over the monolithic classifier measured through cross-validation on the training set.

and varying the information on part-of-speech tags across supervised, unsupervised and none. The results show that combining the monolithic classifier with confusable experts (selected because of their superior performance over the monolithic classifier on heldout data) does not seem to work in our experiments. The best result has a slightly higher error rate than the monolithic classifier (0.89% versus 0.83%), despite the fact that it uses 61 confusable experts to arrive at this result. It appears that the monolithic classifier has an accuracy that is hard to surpass, and may be close to a ceiling performance – note that it is trained on nearly 3 million examples. Also, estimated performances on cross-validation experiments do not provide reliable clues with respect to the performance on unseen data, apparently.

In Table 2 we list the top ten selected confusable experts for our best performing combination system. These top improvements concern words in the medium to low frequencies. Some high-frequency words are also selected for the combination classifier, but these show a much more modest improvement. For example, *word* versus *wordt*, the most frequent confusable, is selected, however with an improvement average of only 0.3%.

Focusing on the usage of supervised, unsupervised, or no part-of-speech information, we observe the following. Overall, using supervised part-of-speech tags as features yields the lowest error rates, both for the monolithic classifiers and the combination systems. The use of unsupervised tags is not favored by our results, as the error rates of these variants hardly differ from those of the systems that do not employ part-of-speech information at all.

In sum, the lowest estimated error rate is 0.83%, produced by the monolithic classifier using supervised part-of-speech tags. This error rate appears low; yet, the question is whether these systems are accurate enough to actually spot cases of

| Baseline error | System | POS tagging | | |
|----------------|------------|-------------|--------------|-------------|
| | | supervised | unsupervised | no POS tags |
| 31 | Monolithic | 63 | 63 | 65 |
| 36 | Keep | 75 | 68 | 67 |
| 36 | ¬Keep | 79 | 63 | 65 |

Table 3: Baseline and system error detection accuracy (%) on manually collected errors, by the monolithic classifiers and the *Keep* and *¬Keep* systems, under three POS tagging conditions. Boldfaced results mark the highest detection accuracy.

confusions to a reasonable degree.

5.2 Error Detection Capability

To obtain an estimate of the error detection capabilities of all 2×3 system variants, we measure their error-detection performance on confusions as they occur in texts from the web. We gathered 525 errors, bootstrapping our search using simple two-word queries such as “*ik wordt*”, indicative of possible errors. We manually selected all genuine cases of errors from the search results, making sure that we sampled widely across confusable words and error patterns. All sentences with errors were tokenized and part-of-speech tagged, converted to windowed examples, and processed by the 2×3 systems, as well as by the most-frequent outcome baseline systems. Correct error detection occurs when the system predicts a different outcome than actually present in the test example; if the system agrees with the error, the system has not detected it.

Table 3 lists the error detection accuracies on the collected errors. Comparing Table 3 to Table 1, one observation is that the highest error detection accuracy, 79%, is not obtained by the system with the lowest error rate. In fact, the system that uses supervised part-of-speech information and does *not* keep the training data of the confusable experts in the training data of the monolithic classifier, is the best error detector. The utility of using unsupervised part-of-speech data is not apparent from the results obtained. Again, we do not see evidence that unsupervised part-of-speech information could replace supervised part-of-speech information.

Overall, we observe that our classifiers perform markedly better than the most-frequent baselines; all systems more than double the accuracy of the baseline system (31% for the most-frequent outcome baseline, 36% for the confusable expert baseline). In terms of error reduction over the baseline, our best system is able to reduce the error by 70%.

As an additional appraisal of our approach, we compared the detection capacities of our classifiers against that of the grammar checker in the Dutch proofing tools included in the commercial word processor Microsoft Word 2003. Tested on the 525 errors, the Word spelling checker spots 88 errors, or 17%, and does not raise an alarm with the remaining errors. Our approach, with 63%–79%, clearly

outperforms the Word grammar checker with respect to d/dt-error detection.

Yet, the 79% detection accuracy is considerably lower than the 99% accuracy obtained on test data, attained by the best classifiers. The discrepancy between these numbers can be explained as follows. The 99% accuracy score concerns the disambiguation of all cases of d/dt words occurring in context in newspaper text, most of which can be expected to be correct. Newspaper articles constitute professionally text that tends to be proofread and checked before publication—it has been mentioned that in texts published by the Associated Press service, 1 in 2000 words are incorrectly spelled (Church and Gale 1991). In contrast, our external test set contains only errors, occurring in text that is mostly non-professionally written, and often contains other grammar and spelling errors as well. Of the 525 errors in context, 79% can be detected correctly; it would seem that these 79% occur in contexts that resembles a context seen earlier in the newspaper text training material, while there is a 21% portion of cases where the context does not provide clues that the d/dt form in focus is actually inappropriate.

6 Conclusion

We presented an approach to detecting a class of confusable errors, namely those related to choices in the writing process between two similar words which differ only in their ending. We exemplified the approach on distinguishing between Dutch homophonic words ending in *-d* and their counterpart ending in *-dt*, the latter typically marking a second-person or third-person present-tense verbal inflection, such as *word* versus *wordt* (*become* versus *becomes*).

The reasoning underlying our approach is that by training classifiers on large amounts of confusable cases that can disambiguate between confusable alternatives at high accuracy, we effectively produce grammar checking subsystems that can pinpoint errors in text. By virtue of being very accurate, a discrepancy between the classifier and the actual confusable word as it occurs in running text, may well signal that the word in the text is contextually inappropriate. The question is whether we are able to train classifiers with such high accuracy (or low error rate) that they indeed can pinpoint errors with high accuracy as well. On the test problem we attained an error rate of under 1%. When applied to a manually gathered list of confusable errors found on the web, the systems are able to detect over 70% of the errors (at best 79%), markedly better than naive most-frequent outcome baselines which only correct 35% of the errors, and also better than Microsoft Word 2003 which detected only 17% of the errors.

Our systems consist of a combination of selected confusable experts combined with a back-off monolithic classifier. The selection of confusable experts is performed automatically, based on superior performance over the monolithic classifier in a cross-validation experiment on training data. In a 2×3 experimental matrix, we varied (1) whether the monolithic classifier retains or loses the training examples of the selected confusable experts, and (2) which part-of-speech information is used as input features: supervised, unsupervised, or no tags at all. On the target task, the detection of errors, we found that the best system was the ensemble

system composed of selected confusable experts, where the monolithic back-off classifier did not retain the training examples of the selected confusable experts, and supervised part-of-speech tags were used in the input feature vector. We did not observe that using unsupervised tags be a proxy for having supervised tags; hence, for now we have to assume that our approach assumes the presence of a language-specific part-of-speech tagger.

To conclude the paper we discuss the novelty and generality of the approach, and we critically review the issue of evaluation. First, the core novelty of our approach lies in the fact that our systems use an ensemble architecture. In the ensemble, some classifiers are confusable-specific (the confusable experts), while one monolithic classifier generically solves the problem. Our systems are gating systems (cf. Figure 2) that use the monolithic classifier as a back-off classifier for solving those cases not covered by the selected confusable experts. Earlier approaches either focused on the monolithic solution (Yarowsky & Wicentowski 2000) or on individual confusable expert submodules (Golding & Roth 1999). Our results on error detection show that the monolithic approach can be improved by adding a selection of confusable experts in the ensemble.

In sum, we believe the proposed approach to be usable in proofing tools.

7 Discussion

Concerning the generality of the approach, the proposed method applies to confusable cases in which the writer is forced to choose among a set of paradigmatic alternatives for derivation or inflection, and in which the alternatives have a similar but discernable surface form that uniquely identifies at least one of the two alternatives. A word ending in *-dt* will almost exclusively be a present-tense singular verb form, or a typo of a word that should end in *-d*. This identification enables the knowledge-free extraction of these cases and their counterparts from large text corpora; no annotation effort is needed.

Intrinsic to our approach is the inclusion of “leaked” examples of words with other part-of-speech tags than just the main tag associated with the underlying inflection, e.g. in our case, past participles and nouns ending in *-d*. These words however can also be, and judging from web queries, are, frequently misspelled with *-dt* at the end. These occurrences will in principle also be corrected by our system.

Similar cases to which this method could be applied straightforwardly include confusable sets with two or more than two outcomes, as long as they keep the property of having marked alternative word endings of which at least one uniquely identifies a morphological inflection or derivation. If one is marked, then the counterparts can be extracted automatically as well. Some random examples that fit the bill would be Dutch diminutive inflection *-tje* vs. *-je*, *-etje*, *-pje*, and *-kje* (Daelemans et al. 1997a); English gerund/infinitive inflection *-ing* versus forms without *-ing*; *-y* versus *-ily*, and *-ation* versus *-ization*.

A final point concerns evaluation. On the larger issue of spelling correction, it has been argued that the precision and recall of error detection and cor-

rection (which are not the same, but in our case of two-way confusable outcomes, are conflated) constitute the best evaluation of a spelling correction system (Reynaert 2005). A spelling error detector should not produce false hits (detect an error when there is none) nor produce misses (fail to detect an error where there is one). Precision and recall are the appropriate measurements for these two aspects. However, to estimate precision and recall reliably, the test would involve processing a (preferably large) corpus of free text in which all actual errors are detected in advance. As these corpora are not available as yet, we have to downgrade our evaluation to recall estimates such as the one presented in this paper, in which a precompiled list of errors in context is presented to the error detector. We do not know yet the precision of our systems, i.e., the relative amount of cases it disagrees with a wordform in free text in which it actually flags an error correctly. Estimates on professionally written, edited text such as the text we train our classifiers on, indicate that this precision is only between 2% and 10% – but this is double-checked text that is supposed to be devoid of errors already. If in journalistic texts only 1 in 2000 words are incorrect (Church & Gale 1991), then a 99% correct classifier would raise about 20 alarms in a 2000-word text, while there would only be one error in the text; the precision could in that case be 5% at best.

In general we recommend that our method be applied to text that is written outside of professional context, such as a lot of the material on the world-wide web (Ringlstetter et al. 2006).

References

- Banko, M. and E. Brill (2001), Scaling to very very large corpora for natural language disambiguation, *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 26–33.
- Biemann, Christian (2007), *Unsupervised and Knowledge-free Natural Language Processing in the Structure Discovery Paradigm*, PhD thesis, Leipzig University.
- Church, Kenneth Ward and William A. Gale (1991), Probability scoring for spelling correction, *Statistics and Computing* **1**(2), 93–103.
- Daelemans, W., A. Van den Bosch, and A. Weijters (1997a), IGTREE: using trees for compression and classification in lazy learning algorithms, *Artificial Intelligence Review* **11**, 407–423.
- Daelemans, W., P. Berck, and S. Gillis (1997b), Data mining as a method for linguistic analysis: Dutch diminutives, *Folia Linguistica* **XXXI**(1–2), 57–75.
- Even-Zohar, Y. and D. Roth (2000), A classification approach to word prediction, *Proceedings of the First North-American Conference on Computational Linguistics*, ACL, New Brunswick, NJ, pp. 124–131.
- Geerts, G., W. Haeseryn, J. de Rooij, and M. van der Toorn (1984), *Algemene Nederlandse Spraakkunst*, Wolters-Noordhoff, Groningen and Wolters, Leuven.

- Golding, A. R. (1995), A Bayesian hybrid method for context-sensitive spelling correction, *Proceedings of the 3rd workshop on very large corpora, ACL-95*.
- Golding, A.R. and D. Roth (1999), A Winnow-Based Approach to Context-Sensitive Spelling Correction, *Machine Learning* **34**(1–3), 107–130.
- Huang, J. H. and D. W. Powers (2001), Large scale experiments on correction of confused words, *Australasian Computer Science Conference Proceedings*, Bond University, Queensland AU, pp. 77–82.
- Mangu, L. and E. Brill (1997), Automatic rule acquisition for spelling correction, *Proceedings of the International Conference on Machine Learning*, pp. 187–194.
- Mooney, R. J. and M. E. Califf (1995), Induction of first-order decision lists: Results on learning the past tense of English verbs, *Journal of Artificial Intelligence Research* **3**, 1–24.
- Quinlan, J.R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA.
- Reynaert, M. (2005), *Text-induced spelling correction*, PhD thesis, Tilburg University.
- Ringlstetter, C., K. Schultz, and S. Mihov (2006), Orthographic errors in web pages: Toward cleaner web corpora, *Computational Linguistics* **32**(3), 295–340.
- Sandra, D., F. Daems, and S. Frisson (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.
- Schone, P. and D. Jurafsky (2001), Knowledge-free induction of inflectional morphologies.
- Van den Bosch, A. (2006), Scalable classification-based word prediction and confusable correction, *Traitement Automatique des Langues* **46**(2), 39–63.
- Van den Bosch, A. and W. Daelemans (2007), Dat gebeurt mei niet: Computationale modellen voor verwarbare homofonen.
- Wu, D., Z. Sui, and J. Zhao (1999), An information-based method for selecting feature types for word prediction, *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, *Proceedings of the Annual Meeting of the ACL*, pp. 88–95.
- Yarowsky, D. and R. Wicentowski (2000), Minimally supervised morphological analysis by multimodal alignment, *Proceedings of ACL-2000*, Morgan Kaufmann, San Francisco, CA, pp. 207–216.