A Comparative Evaluation of Deep and Shallow Approaches to the Automatic Detection of Common Grammatical Errors

Joachim Wagner, Jennifer Foster, and Josef van Genabith*
National Centre for Language Technology
School of Computing, Dublin City University, Dublin 9, Ireland
{jwagner, jfoster, josef}@computing.dcu.ie

Abstract

This paper compares a deep and a shallow processing approach to the problem of classifying a sentence as grammatically well-formed or ill-formed. The deep processing approach uses the XLE LFG parser and English grammar: two versions are presented, one which uses the XLE directly to perform the classification, and another one which uses a decision tree trained on features consisting of the XLE's output statistics. The shallow processing approach predicts grammaticality based on n-gram frequency statistics: we present two versions, one which uses frequency thresholds and one which uses a decision tree trained on the frequencies of the rarest n-grams in the input sentence. We find that the use of a decision tree improves on the basic approach only for the deep parser-based approach. We also show that combining both the shallow and deep decision tree features is effective. Our evaluation is carried out using a large test set of grammatical and ungrammatical sentences. The ungrammatical test set is generated automatically by inserting grammatical errors into well-formed BNC sentences.

1 Introduction

This paper is concerned with the task of predicting whether a sentence contains a grammatical error. An accurate method for carrying out automatic grammaticality judgements has uses in the areas of computer-assisted language learning and grammar checking. Comparative evaluation of existing error detection approaches has been hampered by a lack of large and commonly used evaluation error corpora. We attempt to overcome this by automatically creating a large error corpus, containing four different types of frequently occurring grammatical errors. We use this corpus to evaluate the performance of two approaches to the task of automatic error detection. One approach uses low-level detection techniques based on POS n-grams. The other approach is a novel parser-based method which employs deep linguistic processing to discriminate grammatical input from ungrammatical. For both approaches, we implement a basic solution, and then attempt to improve upon this solution using a decision tree classifier. We show that combining both methods improves upon the individual methods.

N-gram-based approaches to the problem of error detection have been proposed and implemented in various forms by Atwell(1987), Bigert and Knutsson (2002), and Chodorow and Leacock (2000) amongst others. Existing approaches are hard to compare since they are evaluated on different test sets which vary in size and error density. Furthermore, most of these approaches concentrate on one type of grammatical error only, namely, context-sensitive or real-word spelling errors. We implement a vanilla n-gram-based approach which is tested on a very large test set containing four different types of error.

The idea behind the parser-based approach to error detection is to use a broad-coverage hand-crafted precision grammar to detect ungrammatical sen-

*Also affiliated to IBM CAS, Dublin.

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 112–121, Prague, June 2007. ©2007 Association for Computational Linguistics

tences. This approach exploits the fact that a precision grammar is designed, in the traditional generative grammar sense (Chomsky, 1957), to distinguish grammatical sentences from ungrammatical sentences. This is in contrast to treebank-based grammars which tend to massively overgenerate and do not generally aim to discriminate between the two. In order for our approach to work, the coverage of the precision grammars must be broad enough to parse a large corpus of grammatical sentences, and for this reason, we choose the XLE (Maxwell and Kaplan, 1996), an efficient and robust parsing system for Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982) and the ParGram English grammar (Butt et al., 2002) for our experiments. This system employs robustness techniques, some borrowed from Optimality Theory (OT) (Prince and Smolensky, 1993), to parse extra-grammatical input (Frank et al., 1998), but crucially still distinguishes between optimal and suboptimal solutions.

The evaluation corpus is a subset of an ungrammatical version of the British National Corpus (BNC), a 100 million word balanced corpus of British English (Burnard, 2000). This corpus is obtained by automatically inserting grammatical errors into the original BNC sentences based on an analysis of a manually compiled "real" error corpus.

This paper makes the following contributions to the task of automatic error detection:

1. A novel deep processing XLE-based approach
2. An effective and novel application of decision tree machine learning to both shallow and deep approaches
3. A novel combination of deep and shallow processing
4. An evaluation of an n-gram-based approach on a wider variety of errors than has previously been carried out
5. A large evaluation error corpus

The paper is organised as follows: in Section 2, we describe previous approaches to the problem of error detection; in Section 3, a description of the error corpus used in our evaluation experiments is presented, and in Section 4, the two approaches to error detection are presented, evaluated, combined and compared. Section 5 provides a summary and suggestions for future work.

2 Background

2.1 Precision Grammars

A precision grammar is a formal grammar designed to distinguish ungrammatical from grammatical sentences. This is in contrast to large treebank-induced grammars which often accept ungrammatical input (Charniak, 1996). While high coverage is required, it is difficult to increase coverage without also increasing the amount of ungrammatical sentences that are accepted as grammatical by the grammar. Most publications in grammar-based automatic error detection focus on locating and categorising errors and giving feedback. Existing grammars are re-used (Vandeventer Faltin, 2003), or grammars of limited size are developed from scratch (Reuer, 2003).

The ParGram English LFG is a hand-crafted broad-coverage grammar developed over several years with the XLE platform (Butt et al., 2002). The XLE parser uses OT to resolve ambiguities (Prince and Smolensky, 1993). Grammar constraints resulting in rare constructions can be marked as "dispreferred" and constraints resulting in common ungrammatical constructions can be marked as "ungrammatical". The use of constraint ordering and marking increases the robustness of the grammar, while maintaining the grammatical / ungrammatical distinction (Frank et al., 1998). The English Resource Grammar (ERG) is a precision Head-Driven Phrase Structure Grammar (HPSG) of English (Copestake and Flickinger, 2000; Pollard and Sag, 1994). Its coverage is not as broad as the XLE English grammar. Baldwin et al. (2004) propose a method to identify gaps in the grammar. Blunsom and Baldwin (2006) report ongoing development.

There has been previous work using the ERG and the XLE grammars in the area of computer-assisted language learning. Bender et al. (2004) use a version of the ERG containing mal-rules to parse ill-formed sentences from the SST corpus of Japanese learner English (Emi et al., 2004). They then use the semantic representations of the ill-formed input to generate well-formed corrections. Khader et al. (2004) study whether the ParGram English LFG can be used for computer-assisted language learning by adding additional OT marks for ungrammatical constructions observed in a learner corpus. However, the evaluation is preliminary, on only 50 test items.

2.2 N-gram Methods

Most shallow approaches to grammar error detection originate from the area of real-word spelling error correction. A real-word spelling error is a spelling or typing error which results in a token which is another valid word of the language in question.

The (to our knowledge) oldest work in this area is that of Atwell (1987) who uses a POS tagger to flag POS bigrams that are unlikely according to a reference corpus. While he speculates that the bigram frequency should be compared to how often the same POS bigram is involved in errors in an error corpus, the proposed system uses the raw frequency with an empirically established threshold to decide whether a bigram indicates an error. In the same paper, a completely different approach is presented that uses the same POS tagger to consider spelling variants that have a different POS. In the example sentence *I am very **hit*** the POS of the spelling variant *hot/JJ* is added to the list NN-VB-VBD-VBN of possible POS tags of *hit*. If the POS tagger chooses *hit/JJ*, the word is flagged and the correction *hot* is proposed to the user. Unlike most n-gram-based approaches, Atwell's work aims to detect grammar errors in general and not just real-word spelling errors. However, a complete evaluation is missing.

The idea of disambiguating between the elements of confusion sets is related to word sense disambiguation. Golding (1995) builds a classifier based on a rich set of context features. Mays et al. (1991) apply the noisy channel model to the disambiguation problem. For each candidate correction S' of the input S the probability $P(S')P(S|S')$ is calculated and the most likely correction selected. This method is re-evaluated by Wilcox-O'Hearn et al. (2006) on WSJ data with artificial real-word spelling errors.

Bigert and Knutsson (2002) extend upon a basic n-gram approach by attempting to match n-grams of low frequency with similar n-grams in order to reduce overflagging. Furthermore, n-grams crossing clause boundaries are not flagged and the similarity measure is adapted in the case of phrase boundaries that usually result in low frequency n-grams.

Chodorow and Leacock (2000) use a mutual information measure in addition to raw frequency of n-grams. Apart from this, their ALEK system employs other extensions to the basic approach, for example frequency counts from both generic and word-specific corpora are used in the measures. It is not reported how much each of these contribute to the overall performance.

Rather than trying to implement all of the previous n-gram approaches, we implement the basic approach which uses rare n-grams to predict grammaticality. This property is shared by all previous shallow approaches. We also test our approach on a wider class of grammatical errors.

3 Ungrammatical Data

In this section, we discuss the notion of an artificial error corpus (Section 3.1), define the type of ungrammatical language we are dealing with (Section 3.2), and describe our procedure for creating a large artificial error corpus derived from the BNC (Section 3.3).

3.1 An Artificial Error Corpus

In order to meaningfully evaluate a shallow versus deep approach to automatic error detection, a large test set of ungrammatical sentences is needed. A corpus of ungrammatical sentences can take the form of a learner corpus (Granger, 1993; Emi et al., 2004), i. e. a corpus of sentences produced by language learners, or it can take the form of a more general error corpus comprising sentences which are not necessarily produced in a language-learning context and which contain competence and performance errors produced by native and non-native speakers of the language (Becker et al., 1999; Foster and Vogel, 2004; Foster, 2005). For both types of error corpus, it is not enough to collect a large set of sentences which are likely to contain an error - it is also necessary to examine each sentence in order to determine whether an error has actually occurred, and, if it has, to note the nature of the error. Thus, like the creation of a treebank, the creation of a corpus of ungrammatical sentences requires time and linguistic knowledge, and is by no means a trivial task.

A corpus of ungrammatical sentences which is large enough to be useful can be created automatically by inserting, deleting or replacing words in grammatical sentences. These transformations should be linguistically realistic and should, therefore, be based on an analysis of naturally produced grammatical errors. Automatically generated error corpora have been used before in natural language processing. Bigert (2004) and Wilcox-O'Hearn et al. (2006), for example, automatically introduce spelling errors into texts. Here, we generate a large error corpus by automatically inserting four different kinds of grammatical errors into BNC sentences.

3.2 Commonly Produced Grammatical Errors

Following Foster (2005), we define a sentence to be ungrammatical if all the words in the sentence are well-formed words of the language in question, but the sentence contains one or more error. This error can take the form of a performance slip which can occur due to carelessness or tiredness, or a competence error which occurs due to a lack of knowledge of a particular construction. This definition includes real-word spelling errors and excludes non-word spelling errors. It also excludes the abbreviated informal language used in electronic communication. Using the above definition as a guideline, a 20,000 word corpus of ungrammatical English sentences was collected from a variety of written texts including newspapers, academic papers, emails and website forums (Foster and Vogel, 2004; Foster, 2005). The errors in the corpus were carefully analysed and classified in terms of how they might be corrected using the three word-level correction operators: insert, delete and substitute. The following frequency ordering of the three word-level correction operators was found:
substitute (48%) $>$ *insert* (24%) $>$ *delete* (17%) $>$ *combination* (11%)
Stemberger (1982) reports the same ordering of the substitution, deletion and insertion correction operators in a study of native speaker spoken language slips. Among the grammatical errors which can be corrected by substituting one word for another, the most common errors are real-word spelling errors and agreement errors. In fact, 72% of all errors fall into one of the following four classes:

1. missing word errors:
   *What **are** the subjects?* $>$ *What the subjects?*
2. extra word errors:
   Was that in the summer? $>$ *Was that in the summer **in**?*
3. real-word spelling errors:
   *She could **not** comprehend.* $>$ *She could **no** comprehend.*
4. agreement errors:
   She steered Melissa round a corner. $>$ *She steered Melissa round a **corners**.*

A similar classification was adopted by Nicholls (1999), having analysed the errors in a learner corpus. Our research is currently limited to the four error types given above, i. e. missing word errors, extra word errors, real-word spelling errors and agreements errors. However, it is possible for it to be extended to handle a wider class of errors.

3.3 Automatic Error Creation

The error creation procedure takes as input a part-of-speech-tagged corpus of sentences which are assumed to be well-formed, and outputs a corpus of ungrammatical sentences. The automatically introduced errors take the form of the four most common error types found in the manually created corpus, i. e. missing word errors, extra word errors, real-word spelling errors and agreement errors. For each sentence in the original tagged corpus, an attempt is made to automatically produce four ungrammatical sentences, one for each of the four error types. Thus, the output of the error creation procedure is, in fact, four error corpora.

3.3.1 Missing Word Errors

In the manually created error corpus of Foster (2005), missing word errors are classified based on the part-of-speech (POS) of the missing word. 98% of the missing parts-of-speech come from the following list (the frequency distribution in the error corpus is given in brackets):
det (28%) $>$ *verb (23%)* $>$ *prep (21%)* $>$ *pro (10%)* $>$ *noun (7%)* $>$ *"to" (7%)* $>$ *conj (2%)*
We use this information when introducing missing word errors into the BNC sentences. For each sentence, all words with the above POS tags are noted. One of these is selected and deleted. The above frequency ordering is respected so that, for example, missing determiner errors are produced more often than missing pronoun errors. No ungrammatical

sentence is produced if the original sentence contains just one word or if the sentence contains no words with parts-of-speech in the above list.

3.3.2 Extra Word Errors

We introduce extra word errors in the following three ways:

1. Random duplication of any token within a sentence: *That's the way* ***we we*** *learn here.*
2. Random duplication of any POS within a sentence: *There* ***it he*** *was.*
3. Random insertion of an arbitrary token into the sentence: *Joanna drew* ***as*** *a long breadth.*

Apart from the case of duplicate tokens, the extra words are selected from a list of tagged words compiled from a random subset of the BNC. Again, our procedure for inserting extra words is based on the analysis of extra word errors in the 20,000 word error corpus of Foster (2005).

3.3.3 Real-Word Spelling Errors

We classify an error as a real-word spelling error if it can be corrected by replacing the erroneous word with another word with a Levenshtein distance of one from the erroneous word, e.g. *the* and *they*. Based on the analysis of the manually created error corpus (Foster, 2005), we compile a list of common English real-word spelling error word pairs. For each BNC sentence, the error creation procedure records all tokens in the sentence which appear as one half of one of these word pairs. One token is selected at random and replaced by the other half of the pair. The list of common real-word spelling error pairs contains such frequently occurring words as *is* and *a*, and the procedure therefore produces an ill-formed sentence for most input sentences.

3.3.4 Agreement Errors

We introduce subject-verb and determiner-noun number agreement errors into the BNC sentences. We consider both types of agreement error equally likely and introduce the error by replacing a singular determiner, noun or verb with its plural counterpart, or vice versa. For English, subject-verb agreement errors can only be introduced for present tense verbs, and determiner-noun agreement errors can only be introduced for determiners which are marked for number, e.g. demonstratives and the indefinite article. The procedure would be more productive if applied to a morphologically richer language.

3.3.5 Covert Errors

James (1998) uses the term *covert error* to describe a genuine language error which results in a sentence which is syntactically well-formed under some interpretation different from the intended one. The prominence of covert errors in our automatically created error corpus is estimated by manually inspecting 100 sentences of each error type. The percentage of grammatical structures that are inadvertently produced for each error type and an example of each one are shown below:

- Agreement Errors, 7%
  Mary's staff ***include*** *Jones,Smith and Murphy* > *Mary's staff* ***includes*** *Jones,Smith and Murphy*
- Real-Word Spelling Errors, 10%
  And ***then****?* > *And* ***them****?*
- Extra Word Errors, 5%
  in defiance of the free rider prediction > *in defiance of the free rider* ***near*** *prediction*
- Missing Word Errors, 13%
  She steered ***Melissa*** *round a corner* > *She steered round a corner*

The occurrence of these *covert errors* can be reduced by fine-tuning the error creation procedure but they can never be completely eliminated. Indeed, they should not be eliminated from the test data, because, ideally, an optimal error detection system should be sophisticated enough to flag syntactically well-formed sentences containing covert errors as potentially ill-formed.[1]

4 Error Detection Evaluation

In this section we present the error detection evaluation experiments. The experimental setup is explained in Section 4.1, the results are presented in Section 4.2 and they are analysed in Section 4.3.

[1] An example of this is given in the XLE User Documentation (`http://www2.parc.com/isl/groups/nltt/xle/doc/`). The authors remark that an ungrammatical reading of the sentence *Lets go to the store* in which *Lets* is missing an apostrophe, is preferable to the grammatical yet implausible analysis in which *Lets* is a plural noun.

4.1 Experimental Setup

4.1.1 Test Data and Evaluation Procedure

The following steps are carried out to produce training and test data for this experiment:

1. Speech material, poems, captions and list items are removed from the BNC. 4.2 million sentences remain. The order of sentences is randomised.
2. For the purpose of cross-validation, the corpus is split into 10 parts.
3. Each part is passed to the 4 automatic error insertion modules described in Section 3.3, resulting in 40 additional sets of varying size.
4. The first 60,000 sentences of each of the 50 sets, i. e. 3 million sentences, are parsed with XLE.[2]
5. N-gram frequency information is extracted for the first 60,000 sentences of each set. An additional 20,000 is extracted as held-out data.
6. 10 sets with mixed error types are produced by joining a quarter of each respective error set.
7. For each error type (including mixed errors) and cross-validation set, the 60,000 grammatical and 60,000 ungrammatical sentences are joined.
8. Each cross-validation run uses one set out of the 10 as test data (120,000 sentences) and the remaining 9 sets for training (1,080,000 sentences).

The experiment is a standard binary classification task. The methods classify the sentences of the test sets as grammatical or ungrammatical. We use the standard measures of precision, recall, f-score and accuracy (Figure 1). True positives are understood to be ungrammatical sentences that are identified as such. The baseline precision and accuracy is 50% as half of the test data is ungrammatical. If 100% of the test data is classified as ungrammatical, recall will be 100% and f-score $2/3$. Recall shows the accuracy we would get if the grammatical half of the test data was removed. Parametrised methods are first optimised for accuracy and then the other measures are taken. Therefore, f-scores below the artificial $2/3$ baseline are meaningful.

[2]We use the XLE command *parse-testfile* with *parse-literally* set to 1, *max xle scratch storage* set to 1,000 MB, *time-out* to 60 seconds, and the XLE English LFG. Skimming is not switched on and fragments are.

Measure	Formula
precision	$tp/(tp+fp)$
recall	$tp/(tp+fn)$
f-score	$2pr*re/(pr+re)$
accuracy	$(tp+tn)/(tp+tn+fp+fn)$

Figure 1: Evaluation measures: tp = true positives, fp = false positives, tn = true negatives, fn = false negatives, pr = precision, re = recall

4.1.2 Method 1: Precision Grammar

According to the XLE documentation, a sentence is marked with a star (*) if its optimal solution uses a constraint marked as ungrammatical. We use this star feature, parser exceptions and zero number of parses to classify a sentence as ungrammatical.

4.1.3 Method 2: POS N-grams

In each cross-validation run, the full data of the remaining 9 sets of step 2 of the data generation (see Section 4.1.1) is used as a reference corpus of $0.9\times4,200,000 = 3,800,000$ assumedly grammatical sentences. The reference corpora and data sets are POS tagged with the IMS TreeTagger (Schmidt, 1994). Frequencies of POS n-grams ($n = 2,\ldots,7$) are counted in the reference corpora. A test sentence is flagged as ungrammatical if it contains an n-gram below a fixed frequency threshold. Method 2 has two parameters: n and the frequency threshold.

4.1.4 Method 3: Decision Trees on XLE Output

The XLE parser outputs additional statistics for each sentence that we encode in six features:

- An integer indicating starredness (0 or 1) and various parser exceptions (-1 for time out, -2 for exceeded memory, etc.)
- The number of optimal parses[3]
- The number of unoptimal parses
- The duration of parsing
- The number of subtrees
- The number of words

[3]The use of preferred versus dispreferred constraints are used to distinguish optimal parses from unoptimal ones.

Training data for the decision tree learner is composed of $9 \times 60,000 = 540,000$ feature vectors from grammatical sentences and $9 \times 15,000 = 135,000$ feature vectors from ungrammatical sentences of each error type, resulting in equal amounts of grammatical and ungrammatical training data.

We choose the weka implementation of machine learning algorithms for the experiments (Witten and Frank, 2000). We use a J48 decision tree learner with the default model.

4.1.5 Method 4: Decision Trees on N-grams

Method 4 follows the setup of Method 3. However, the features are the frequencies of the rarest n-grams ($n = 2, \ldots, 7$) in the sentence. Therefore, the feature vector of one sentence contains 6 numbers.

4.1.6 Method 5: Decision Trees on Combined Feature Sets

This method combines the features of Methods 3 and 4 for training a decision tree.

4.2 Results

Table 1 shows the results for Method 1, which uses XLE starredness, parser exceptions[4] and zero parses to classify grammaticality. Table 2 shows the results for Method 2, the basic n-gram approach. Table 3 shows the results for Method 3, which classifies based on a decision tree of XLE features. The results for Method 4, the n-gram-based decision tree approach, are shown in Table 4. Finally, Table 5 shows the results for Method 5 which combines n-gram and XLE features in decision trees.

In the case of Method 2, we first have to find optimal parameters. As only very limited integer values for n and the threshold are reasonable, an exhaustive search is feasible. We considered $n = 2, \ldots, 7$ and frequency thresholds below 20,000. Separate held-out data (400,000 sentences) is used in order to avoid overfitting. Best accuracy is achieved with 5-grams and a threshold of 4. Table 2 reports results with these parameters.

[4]XLE parsing (see footnote 2 for configuration) runs out of time for 0.7 % and out of memory for 2.5 % of sentences, measured on training data of the first cross-validation run, i. e. 540,000 grammatical sentence and 135,000 of each error type. 14 sentences of 3 million caused the parser to terminate abnormally.

Error type	Pr.	Re.	F-Sc.	Acc.
Agreement	66.2	64.6	65.4	65.8
Real-word	63.5	57.3	60.3	62.2
Extra word	64.4	59.7	62.0	63.4
Missing word	59.2	47.8	52.9	57.4
Mixed errors	63.5	57.3	60.3	62.2

Table 1: Classification results with XLE starredness, parser exceptions and zero parses (Method 1)

Error type	Pr.	Re.	F-Sc.	Acc.
Agreement	58.6	51.7	55.0	57.6
Real-word	64.0	64.9	64.5	64.2
Extra word	64.8	67.3	66.0	65.4
Missing word	57.2	48.8	52.7	56.1
Mixed errors	61.5	58.2	59.8	60.8

Table 2: Classification results with 5-gram and frequency threshold 4 (Method 2)

The standard deviation of results across cross-validation runs is below 0.006 on all measures, except for Method 4. Therefore we only report average percentages. The highest observed standard deviation is 0.0257 for recall of Method 4 on agreement errors.

For Methods 3, 4 and 5, the decision tree learner optimises accuracy and, in doing so, chooses a trade-off between precision and recall.

4.3 Analysis

Both Method 1 (Table 1) and Method 2 (Table 2) achieve above baseline accuracy for all error types. However, Method 1, which uses the XLE starred feature, parser exceptions and zero parses to determine whether or not a sentence is grammatical, slightly outperforms Method 2, which uses the fre-

Error type	Pr.	Re.	F-Sc.	Acc.
Agreement	67.0	79.3	72.6	70.1
Real-word	63.4	67.6	65.4	64.3
Extra word	63.0	66.4	64.7	63.7
Missing word	59.7	57.8	58.7	59.4
Mixed errors	63.4	67.8	65.6	64.4

Table 3: Classification results with decision tree on XLE output (Method 3)

Error type	Pr.	Re.	F-Sc.	Acc.
Agreement	61.2	53.8	57.3	59.9
Real-word	65.3	64.3	64.8	65.1
Extra word	66.4	67.4	66.9	66.7
Missing word	59.1	49.2	53.7	57.5
Mixed errors	63.3	58.7	60.9	62.3

Table 4: Classification results with decision tree on vectors of frequency of rarest n-grams (Method 4)

Error type	Pr.	Re.	F-Sc.	Acc.
Agreement	67.1	75.2	70.9	69.2
Real-word	65.8	70.7	68.1	67.0
Extra word	65.9	71.2	68.5	67.2
Missing word	61.2	58.0	59.5	60.6
Mixed errors	65.2	68.8	66.9	66.0

Table 5: Classification results with decision tree on joined feature set (Method 5)

quency of POS 5-grams to detect an error. The XLE deep-processing approach is better than the n-gram-based approach for agreement errors (f-score +10.4). Examining the various types of agreement errors, we can see that this is especially the case for singular subjects followed by plural copula verbs (recall +37.7) and determiner-noun number mismatches (recall +23.6 for singular nouns and +18.0 for plural nouns), but not for plural subjects followed by singular verbs (recall -24.0). The relatively poor performance of Method 2 on agreement errors involving determiners could be due to the lack of agreement marking on the Penn Treebank determiner tag used by TreeTagger.

Method 1 is outperformed by Method 2 for real-word spelling and extra word errors (f-score -4.2, -4.0). Unsurprisingly, Method 2 has an advantage on those real-word spelling errors that change the POS (recall -8.8 for Method 1). Both methods perform poorly on missing word errors. For both methods there are only very small differences in performance between the various missing word error subtypes (identified by the POS of the deleted word).

Method 3, which uses machine learning to exploit all the information returned by the XLE parser, improves performance from Method 1, the basic XLE method, for all error types.[5] The general improvement comes from an improvement in recall, meaning that more ungrammatical sentences are actually flagged as such without compromising precision. The improvement is highest for agreement errors (f-score +7.2). Singular subject with plural copula errors (e. g. *The man are*) peak at a recall of 91.0. The Method 3 results indicate that information on the number of solutions (optimal and unoptimal), the number of subtrees, the time taken to parse the sentence and the number of words can be used to predict grammaticality. It would be interesting to investigate this approach with other parsers.

Method 4, which uses a decision tree with n-gram-based features, confirms the results of Method 2. The decision trees' root nodes are similar or even identical (depending on cross-validation run) to the decision rule of Method 2 (5-gram frequency below 4). However, the 10 decision trees have between 1,111 and 1,905 nodes and draw from all features, even bigrams and 7-grams that perform poorly on their own. The improvements are very small though and they are not significant according the criterion of non-overlapping cross-validation results. The main reason for the evaluation of Method 4 is to provide another reference point for comparison of the final method.

The overall best results are those for Method 5, the combined XLE, n-gram and machine-learning-based method, which outperforms the next best method, Method 3, on all error types apart from agreement errors (f-score -1.7, +2.7, +3.8, +0.8). For agreement errors, it seems that the relatively poor results for n-grams have a negative effect on the relatively good results for the XLE. Figure 2 shows that the performance is almost constant on ungrammatical data in the important sentence length range from 5 to 40. However, there is a negative correlation of accuracy and sentence length for grammatical sentences. Very long sentences of any kind tend to be classified as ungrammatical, except for missing word errors which remain close to the 50% baseline of coin-flipping.

For all methods, missing word errors are the worst-performing, particularly in recall (i. e. the ac-

[5]The +0.3 increase in average accuracy for extra word errors is not clearly significant as the results of cross-validation runs overlap.

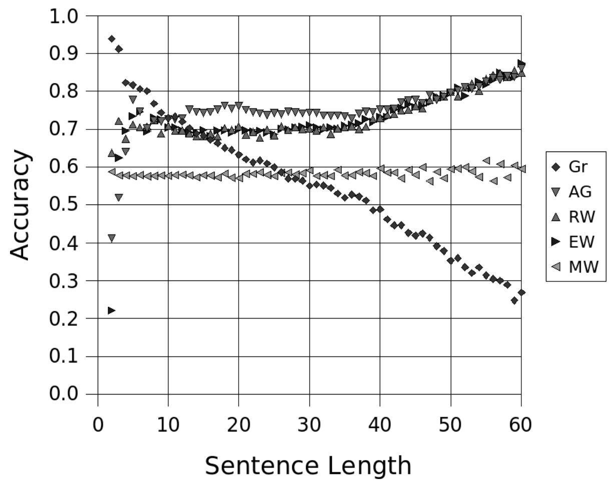

Figure 2: Accuracy by sentence length for Method 5 measured on separate grammatical and ungrammatical data: Gr = Grammatical, AG = Agreement, RW = Real-Word, EW = Extra Word, MW = Missing Word

curacy on ungrammatical data alone). This means that the omission of a word is less likely to result in the sentence being flagged as erroneous. In contrast, extra word errors perform consistently and relatively well for all methods.

5 Conclusion and Future Work

We evaluated a deep processing approach and a POS n-gram-based approach to the automatic detection of common grammatical errors in a BNC-derived artificial error corpus. The results are broken down by error type. Together with the deep approach, a decision tree machine learning algorithm can be used effectively. However, extending the shallow approach with the same learning algorithm gives only small improvements. Combining the deep and shallow approaches gives an additional improvement on all but one error type.

Our plan is to investigate why all methods perform poorly on missing word errors, to extend the error creation procedure so that it includes a wider range of errors, to try the deep approach with other parsers, to integrate additional features from state-of-the-art shallow techniques and to repeat the experiments for languages other than English.

Acknowledgements

This work is supported by the IRCSET Embark Initiative (basic research grant SC/02/298 and postdoctoral fellowship P/04/232). The training and test data used in this reseach is based on the British National Corpus (BNC), distributed by Oxford University Computing Services on behalf of the BNC Consortium. We thank Djamé Seddah for helping us to run the XLE parsing on the SFI/HEA Irish Centre for High-End Computing (ICHEC) and the authors wish to acknowledge ICHEC for the provision of computational facilities and support.

References

Eric Atwell. 1987. How to detect grammatical errors in a text without parsing it. In *Proceedings of the 3rd EACL*, pages 38–45, Morristown, NJ.

Timothy Baldwin, John Beavers, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Beauty and the beast: What running a broad-coverage precision grammar over the BNC taught us about the grammar - and the corpus. In *Pre-Proceedings of the International Conference on Linguistic Evidence: Empirical, Theoretical and Computational Perspectives*, pages 21–26.

Markus Becker, Andrew Bredenkamp, Berthold Crysmann, and Judith Klein. 1999. Annotation of error types for German news corpus. In *Proceedings of the ATALA Workshop on Treebanks*, Paris, France.

Emily M. Bender, Dan Flickinger, Stephan Oepen, and Timothy Baldwin. 2004. Arboretum: Using a precision grammar for grammar checking in CALL. In *Proceedings of the InSTIL/ICALL Symposium: NLP and Speech Technologies in Advanced Language Learning Systems*, Venice, Italy.

Johnny Bigert and Ola Knutsson. 2002. Robust error detection: a hybrid approach combining unsupervised error detection and linguistic knowledge. In *Proceedings RO-MAND-02*, Frascati, Italy.

Johnny Bigert. 2004. Probabilistic detection of context-sensitive spelling errors. In *Proceedings of LREC-04*, volume Five, pages 1633–1636, Lisbon, Portugal.

Phil Blunsom and Timothy Baldwin. 2006. Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of EMNLP-06*, pages 164–171, Sydney.

Lou Burnard. 2000. User reference guide for the British national corpus. Technical report, Oxford University Computing Services.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7, Morristown, NJ, USA.

Eugene Charniak. 1996. Tree-bank grammars. Technical Report CS-96-02, Department of Computer Science, Brown University.

Martin Chodorow and Claudia Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proceedings of NAACL-00*, pages 140–147, San Francisco, CA.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of LREC-02*, Athens, Greece.

Izumi Emi, Kiyotaka Uchimoto, and Hitoshi Isahara. 2004. The overview of the SST speech corpus of Japanese learner English and evaluation through the experiment on automatic detection of learners' errors. In *Proceedings of LREC-04*, volume Four, pages 1435–1439, Lisbon, Portugal.

Jennifer Foster and Carl Vogel. 2004. Good reasons for noting bad grammar: Constructing a corpus of ungrammatical language. In Stephan Kepser and Marga Reis, editors, *Pre-Proceedings of the International Conference on Linguistic Evidence: Empirical, Theoretical and Computational Perspectives*, pages 151–152, Tübingen, Germany.

Jennifer Foster. 2005. *Good Reasons for Noting Bad Grammar: Empirical Investigations into the Parsing of Ungrammatical Written English*. Ph.D. thesis, University of Dublin, Trinity College, Dublin, Ireland.

Anette Frank, Tracy Holloway King, Jonas Kuhn, and John Maxwell. 1998. Optimality theory style constraint ranking in large-scale LFG grammars. In *Proceedings of LFG-98*, Brisbane, Australia.

Andrew R. Golding. 1995. A Bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 39–53, Boston, MA.

Sylviane Granger. 1993. International corpus of learner English. In J. Aarts, P. de Haan, and N.Oostdijk, editors, *English Language Corpora: Design, Analysis and Exploitation*, pages 57–71. Rodopi, Amsterdam.

Carl James. 1998. *Errors in Language Learning and Use: Exploring Error Analysis*. Addison Wesley Longman.

Ron Kaplan and Joan Bresnan. 1982. Lexical Functional Grammar: a formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press.

Rafiq Abdul Khader, Tracy Holloway King, and Miriam Butt. 2004. Deep CALL grammars: The LFG-OT experiment. http://ling.uni-konstanz.de/pages/home/butt/dgfs04call.pdf.

John Maxwell and Ron Kaplan. 1996. An Efficient Parser for LFG. In *Proceedings of LFG-96*, Grenoble.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management*, 23(5):517–522.

D. Nicholls. 1999. The Cambridge learner corpus – error coding and analysis. In *Summer Workshop on Learner Corpora*, Tokyo, Japan.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications.

Alan Prince and Paul Smolensky. 1993. *Optimality Theory*. MIT Press, Cambridge, Massachusetts.

Veit Reuer. 2003. *PromisD - Ein Analyseverfahren zur antizipationsfreien Erkennung und Erklärung von grammatischen Fehlern in Sprachlehrsystemen*. Ph.D. thesis, Humboldt-Universität zu Berlin, Berlin, Germany.

Helmut Schmidt. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, Manchester, England.

J.P. Stemberger. 1982. Syntactic errors in speech. *Journal of Psycholinguistic Research*, 11(4):313–45.

Anne Vandeventer Faltin. 2003. *Syntactic Error Diagnosis in the context of Computer Assisted Language Learning*. Ph.D. thesis, Université de Genève.

L. Amber Wilcox-O'Hearn, Graeme Hirst, and Alexander Budanitsky. 2006. Real-word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model. http://ftp.cs.toronto.edu/pub/gh/WilcoxOHearn-etal-2006.pdf.

Ian H. Witten and Eibe Frank. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers.