GenERRate: Generating Errors for Use in Grammatical Error Detection

Jennifer Foster
National Centre for Language Technology
School of Computing
Dublin City University, Ireland
jfoster@computing.dcu.ie

Øistein E. Andersen
Computer Laboratory
University of Cambridge
United Kingdom
oa223@cam.ac.uk

Abstract

This paper explores the issue of automatically generated ungrammatical data and its use in error detection, with a focus on the task of classifying a sentence as grammatical or ungrammatical. We present an error generation tool called GenERRate and show how GenERRate can be used to improve the performance of a classifier on learner data. We describe initial attempts to replicate Cambridge Learner Corpus errors using GenERRate.

1 Introduction

In recent years automatically generated ungrammatical data has been used in the training and evaluation of error detection systems, in evaluating the robustness of NLP tools and as negative evidence in unsupervised learning. The main advantage of using such artificial data is that it is cheap to produce. However, it is of little use if it is not a realistic model of the naturally occurring and expensive data that it is designed to replace. In this paper we explore the issues involved in generating synthetic data and present a tool called GenERRate which can be used to produce many different kinds of syntactically noisy data. We use the tool in two experiments in which we attempt to train classifiers to distinguish between grammatical and ungrammatical sentences. In the first experiment, we show how GenERRate can be used to improve the performance of an existing classifier on sentences from a learner corpus of transcribed spoken utterances. In the second experiment we try to produce a synthetic error corpus that is inspired by the Cambridge Learner Corpus (CLC)[1], and we evaluate the difference between a classifier's performance when trained on this data and its performance when trained on original CLC material. The results of both experiments provide pointers on how to improve GenERRate, as well as highlighting some of the challenges associated with automatically generating negative evidence.

The paper is organised as follows: In Section 2, we discuss the reasons why artificial ungrammatical data has been used in NLP and we survey its use in the field, focussing mainly on grammatical error detection. Section 3 contains a description of the GenERRate tool. The two classification experiments which use GenERRate are described in Section 4. Problematic issues are discussed in Section 5 and avenues for future work in Section 6.

2 Background

2.1 Why artifical error data is useful

Before pointing out the benefit of using *artificial* negative evidence in grammatical error detection, it is worth reminding ourselves of the benefits of employing negative evidence, be it artificial or naturally occurring. By grammatical error detection, we mean either the task of distinguishing the grammatical from the ungrammatical at the sentence level or more local targeted error detection, involving the identification, and possibly also correction, of particular types of errors. Distinguishing grammatical utterances from ungrammatical ones involves the use of a binary classifier or a grammaticality scor-

[1] http://www.cambridge.org/elt/corpus/learner_corpus2.htm

Proceedings of the NAACL HLT Workshop on Innovative Use of NLP for Building Educational Applications, pages 82–90,
Boulder, Colorado, June 2009.

ing model. Examples are Andersen (2006; 2007), Okanohara and Tsujii (2007), Sun et al. (2007) and Wagner et al. (2007). In targeted error detection, the focus is on identifying the common errors made either by language learners or native speakers (depending on the application). For ESL applications, this includes the detection of errors involving articles (Han et al., 2006; De Felice and Pulman, 2008; Gamon et al., 2008), prepositions (De Felice and Pulman, 2008; Gamon et al., 2008; Tetreault and Chodorow, 2008), verb forms (Lee and Seneff, 2008b), mass/count noun confusions (Brockett et al., 2006) and word order (Metcalf and Meurers, 2006).

The presence of a pattern in a corpus of well-formed language is positive evidence that the pattern is well-formed. The presence of a pattern in an corpus of ill-formed language is negative evidence that the pattern is erroneous. Discriminative techniques usually lead to more accurate systems than those based on one class alone. The use of the two types of evidence can be seen at work in the system described by Lee and Seneff (2008b): Verb phrases are parsed and their parse trees are examined. If the parse trees resemble the "disturbed" trees that statistical parsers typically produce when an incorrect verb form is used, the verb phrase is considered a likely candidate for correction. However, to avoid overcorrection, positive evidence in the form of Google n-gram statistics is also employed: a correction is only applied if its n-gram frequency is higher than that of the original uncorrected n-gram.

The ideal situation for a grammatical error detection system is one where a large amount of labelled positive *and* negative evidence is available. Depending on the aims of the system, this labelling can range from simply marking a sentence as ungrammatical to a detailed description of the error along with a correction. If an error detection system employs machine learning, the performance of the system will improve as the training set size increases (up to a certain point). For systems which employ learning algorithms with large feature sets (e.g. maximum entropy, support vector machines), the size of the training set is particularly important so that overfitting is avoided. The collection of a large corpus of ungrammatical data requires a good deal of manual effort. Even if the annotation only involves marking the sentence as correct/incorrect, it still requires that the sentence be read and a grammaticality judgement applied to it. If more detailed annotation is applied, the process takes even longer. Some substantially-sized annotated error corpora do exist, e.g. the Cambridge Learner Corpus, but these are not freely available.

One way around this problem of lack of availability of suitably large error-annotated corpora is to introduce errors into sentences automatically. In order for the resulting error corpus to be useful in an error detection system, the errors that are introduced need to resemble those that the system aims to detect. Thus, the process is not without some manual effort: knowing what kind of errors to introduce requires the inspection of real error data, a process similar to error annotation. Once the error types have been specified though, the process is fully automatic and allows large error corpora to be compiled. If the set of well-formed sentences into which the errors are introduced is large and varied enough, it is possible that this will result in ungrammatical sentence structures which learners produce but which have not yet been recorded in the smaller naturally occurring learner corpora. To put it another way, the same type of error will appear in lexically and syntactically varied *contexts*, which is potentially advantageous when training a classifier.

2.2 Where artificial error data has been used

Artificial errors have been employed previously in targeted error detection. Sjöbergh and Knutsson (2005) introduce split compound errors and word order errors into Swedish texts and use the resulting artificial data to train their error detection system. These two particular error types are chosen because they are frequent errors amongst non-native Swedish speakers whose first language does not contain compounds or has a fixed word order. They compare the resulting system to three Swedish grammar checkers, and find that their system has higher recall at the expense of lower precision. Brockett et al. (2006) introduce errors involving mass/count noun confusions into English newswire text and then use the resulting parallel corpus to train a phrasal SMT system to perform error correction. Lee and Seneff (2008b) automatically introduce verb form errors (subject–verb agreement errors, complementation errors and errors in a main verb after an auxiliary) into well-

formed text, parse the resulting text and examine the parse trees produced.

Both Okanohara and Tsujii (2007) and Wagner et al. (2007) attempt to learn a model which discriminates between grammatical and ungrammatical sentences, and both use synthetic negative data which is obtained by distorting sentences from the British National Corpus (BNC) (Burnard, 2000). The methods used to distort the BNC sentences are, however, quite different. Okanohara and Tsujii (2007) generate ill-formed sentences by sampling a probabilistic language model and end up with "pseudo-negative" examples which resemble machine translation output more than they do learner texts. Indeed, machine translation is one of the applications of their resulting discriminative language model. Wagner et al. (2007) introduce grammatical errors of the following four types into BNC sentences: context-sensitive spelling errors, agreement errors, errors involving a missing word and errors involving an extra word. All four types are considered equally likely and the resulting synthetic corpus contains errors that look like the kind of slips that would be made by native speakers (e.g. repeated adjacent words) as well as errors that resemble learner errors (e.g. missing articles). Wagner et al. (2009) report a drop in accuracy for their classification methods when applied to real learner texts as opposed to held-out synthetic test data, reinforcing the earlier point that artificial errors need to be tailored for the task at hand (we return to this in Section 4.1).

Artificial error data has also proven useful in the automatic evaluation of error detection systems. Bigert (2004) describes how a tool called Missplel is used to generate context-sensitive spelling errors which are then used to evaluate a context-sensitive spelling error detection system. The performance of general-purpose NLP tools such as part-of-speech taggers and parsers in the face of noisy ungrammatical data has been automatically evaluated using artificial error data. Since the features of machine-learned error detectors are often part-of-speech n-grams or word–word dependencies extracted from parser output (De Felice and Pulman, 2008, for example), it is important to understand how part-of-speech taggers and parsers react to particular grammatical errors. Bigert et al. (2005) introduce artificial context-sensitive spelling errors into error-free Swedish text and then evaluate parsers and a part-of-speech tagger on this text using their performance on the error-free text as a reference. Similarly, Foster (2007) investigates the effect of common English grammatical errors on two widely-used statistical parsers using distorted treebank trees as references. The procedure used by Wagner et al. (2007; 2009) is used to introduce errors into the treebank sentences.

Finally, negative evidence in the form of automatically distorted sentences has been used in unsupervised learning. Smith and Eisner (2005a; 2005b) generate negative evidence for their *contrastive estimation* method by moving or removing a word in a sentence. Since the aim of this work is not to detect grammatical errors, there is no requirement to generate the kind of negative evidence that might actually be produced by either native or non-native speakers of a language. The negative examples are used to guide the unsupervised learning of a part-of-speech tagger and a dependency grammar.

We can conclude from this survey that synthetic error data *is* useful in a variety of NLP applications, including error detection and evaluation of error detectors. In Section 3, we describe an automatic error generation tool, which has a modular design and is flexible enough to accommodate the generation of the various types of synthetic data described above.

3 Error Generation Tool

GenERRate is an error generation tool which accepts as input a corpus and an *error analysis* file consisting of a list of errors and produces an error-tagged corpus of syntactically ill-formed sentences. The sentences in the input corpus are assumed to be grammatically well-formed. GenERRate is implemented in Java and will be made available to download for use by other researchers.[2]

3.1 Supported Error Types

Error types are defined in terms of their corrections, that is, in terms of the operations (*insert, delete, substitute* and *move*) that are applied to a well-formed sentence to make it ill-formed. As well as being a popular classification scheme in the field of error analysis (James, 1998), it has the advantage of

[2] http://www.computing.dcu.ie/~jfoster/resources/generrate.html

being theory-neutral. This is important in this context since it is hoped that GenERRate will be used to create negative evidence of various types, be it L2-like grammatical errors, native speaker slips or more random syntactic noise. It is hoped that GenERRate will be easy to use for anyone working in linguistics, applied linguistics, language teaching or computational linguistics.

The inheritance hierarchy in Fig. 1 shows the error types that are supported by GenERRate. We briefly describe each error type.

Errors generated by removing a word

- **DeletionError:** Generated by selecting a word at random from the sentence and removing it.
- **DeletionPOSError:** Extends DeletionError by allowing a specific POS to be specified.
- **DeletionPOSWhereError:** Extends DeletionPOSError by allowing left and/or right context (POS tag or *start*/*end*) to be specified.

Errors generated by inserting a word

- **InsertionError:** Insert a random word at a random position. The word is chosen either from the sentence itself or from a word list, and this choice is also random.
- **InsertionFromFileOrSentenceError:** This differs from the InsertionError in that the decision of whether to use the sentence itself or a word list is not made at random but supplied in the error type specification.
- **InsertionPOSError:** Extends InsertionFromFileOrSentenceError by allowing the POS of the new word to be specified.
- **InsertionPOSWhereError:** Analogous to the DeletionPOSWhereError, this extends InsertionPOSError by allowing left and/or right context to be specified.

Errors generated by moving a word

- **MoveError:** Generated by randomly selecting a word in the sentence and moving it to another position, randomly chosen, in the sentence.
- **MovePOSError:** A word tagged with the specified POS is randomly chosen and moved to a randomly chosen position in the sentence.
- **MovePOSWhereError:** Extends MovePOSError by allowing the change in position to be specified in terms of direction and number of words.

```
subst,word,an,a,0.2
subst,NNS,NN,0.4
subst,VBG,TO,0.2
delete,DT,0.1
move,RB,left,1,0.1
```

Figure 2: GenERRate Toy Error Analysis File

Errors generated by substituting a word

- **SubstError:** Replace a random word by a word chosen at random from a word list.
- **SubstWordConfusionError:** Extends SubstError by allowing the POS to be specified (same POS for both words).
- **SubstWordConfusionNewPOSError:** Similar to SubstWordConfusionError, but allows different POSs to be specified.
- **SubstSpecificWordConfusionError:** Replace a specific word with another (e.g. *be*/*have*).
- **SubstWrongFormError:** Replace a word with a different form of the same word. The following changes are currently supported: noun number (e.g. *word*/*words*), verb number (*write*/*writes*), verb form (*writing*/*written*), adjective form (*big*/*bigger*) and adjective/adverb (*quick*/*quickly*). Note that this is the only error type which is language-specific. At the moment, only English is supported.

3.2 Input Corpus

The corpus that is supplied as input to GenERRate must be split into sentences. It does not have to be part-of-speech tagged, but it will not be possible to generate many of the errors if it is not. GenERRate has been tested using two part-of-speech tagsets, the Penn Treebank tagset (Santorini, 1991) and the CLAWS tagset (Garside et al., 1987).

3.3 Error Analysis File

The error analysis file specifies the errors that GenERRate should attempt to insert into the sentences in the input corpus. A toy example with the Penn tagset is shown in Fig. 2. The first line is an instance of a *SubstSpecificWordConfusion* error. The second

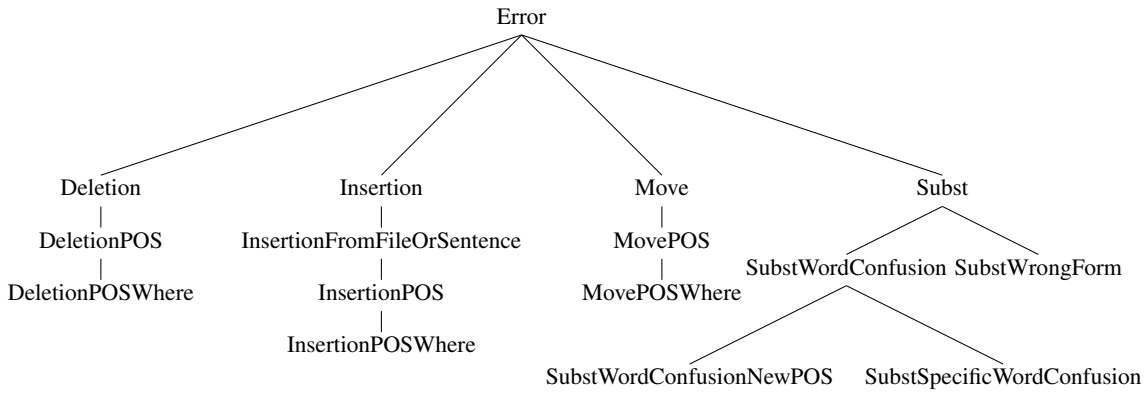

Figure 1: GenERRate Error Types

and third are instances of the *SubstWrongFormError* type. The fourth is a *DeletionPOSError*, and the fifth is a *MovePOSWhereError*. The number in the final column specifies the desired proportion of the particular error type in the output corpus and is optional. However, if it is present for one error type, it must be present for all. The overall size of the output corpus is supplied as a parameter when running GenERRate.

3.4 Error Generation

When frequency information is not supplied in the error analysis file, GenERRate iterates through each error in the error analysis file and each sentence in the input corpus, tries to insert an error of this type into the sentence and writes the resulting sentence to the output file together with a description of the error. GenERRate includes an option to write the sentences into which an error could not be inserted and the reason for the failure to a log file. When the error analysis file *does* include frequency information, a slightly different algorithm is used: for each error, GenERRate selects sentences from the input file and attempts to generate an instance of that error until the desired number of errors has been produced or all sentences have been tried.

4 Classification Experiments

We describe two experiments which involve the use of GenERRate in a binary classification task in which the classifiers attempt to distinguish between grammatically well-formed and ill-formed sentences or, more precisely, to distinguish between sentences in learner corpora which have been annotated as erroneous and their corrected counterparts. In the first experiment we use GenERRate to create ungrammatical training data using information about error types gleaned from a subset of a corpus of transcribed spoken utterances produced by ESL learners in a classroom environment. The classifier is one of those described in Wagner et al. (2007). In the second experiment we try to generate a CLC-inspired error corpus and we use one of the simplest classifiers described in Andersen (2006). Our aim is not to improve classification performance, but to test the GenERRate tool, to demonstrate how it can be used and to investigate differences between synthetic and naturally occurring datasets.

4.1 Experiments with a Spoken Language Learner Corpus

Wagner et al. (2009) train various classifiers to distinguish between BNC sentences and artificially produced ungrammatical versions of BNC sentences (see §2). They report a significant drop in accuracy when they apply these classifiers to real learner data, including the sentences in a corpus of transcribed spoken utterances. The aim of this experiment is to investigate to what extent this loss in accuracy can be reduced by using GenERRate to produce a more realistic set of ungrammatical training examples.

The spoken language learner corpus contains over 4,000 transcribed spoken sentences which were produced by learners of English of all levels and with a variety of L1s. The sentences were produced in a classroom setting and transcribed by the teacher. The transcriptions were verified by the students. All

of the utterances have been marked as erroneous.

4.1.1 Setup

A 200-sentence held-out section of the corpus is analysed by hand and a GenERRate error analysis file containing 89 errors is compiled. The most frequent errors are those involving a change in noun or verb number or an article deletion. GenERRate then applies this error analysis file to 440,930 BNC sentences resulting in the same size set of synthetic examples ("new-ungram-BNC"). Another set of synthetic sentences ("old-ungram-BNC") is produced from the same input using the error generation procedure used by Wagner et al. (2007; 2009). Table 1 shows examples from both sets.

Two classifiers are then trained, one on the original BNC sentences and the old-ungram-BNC sentences, and the other on the original BNC sentences and the new-ungram-BNC sentences. Both classifiers are tested on 4,095 sentences from the spoken language corpus (excluding the held-out section). 310 of these sentences are corrected, resulting in a small set of grammatical test data. The classifier used is the POS n-gram frequency classifier described in Wagner et al. (2007).[3] The features are the frequencies of the least frequent n-grams (2–7) in the input sentence. The BNC (excluding those sentences that are used as training data) is used as reference data to compute the frequencies. Learning is carried out using the *Weka* implementation of the J48 decision tree algorithm.[4]

4.1.2 Results

The results of the experiment are displayed in Table 2. The evaluation measures used are precision, recall, total accuracy and accuracy on the grammatical side of the test data. Recall is the same as accuracy on the ungrammatical side of the test data.

The results are encouraging. There is a significant increase in accuracy when we train on the new-ungram-BNC set instead of the old-ungram-BNC set. This increase is on the ungrammatical side of the test data, i.e. an increase in recall, demonstrating that by analysing a small set of data from our test domain, we can automatically create more effective training data. This is useful in a scenario where a small-to-medium-sized learner corpus is available but which is not large enough to be split into a training/development/test set. These results seem to indicate that reasonably useful training data can be created with minimum effort. Of course, the accuracy is still rather low but we suspect that some of this difference can be explained by domain effects — the sentences in the training data are BNC written sentences (or distorted versions of them) whereas the sentences in the learner corpus are transcribed spoken utterances. Re-running the experiments using the spoken language section of the BNC as training data might yield better results.

4.2 A CLC-Inspired Corpus

We investigate to what extent it is possible to create a large error corpus inspired by the CLC using the current version of GenERRate. The CLC is a 30-million-word corpus of learner English collected from University of Cambridge ESOL exam papers at different levels. Approximately 50% of the CLC has been annotated for errors and corrected.

4.2.1 Setup

We attempt to use GenERRate to insert errors into corrected CLC sentences. In order to do this, we need to create a CLC-specific error analysis file. In contrast to the previous experiment, we do this *automatically* by extracting erroneous POS trigrams from the error-annotated CLC sentences and encoding them as GenERRate errors. This results in approximately 13,000 errors of the following types: DeletionPOSWhereError, InsertionPOSWhereError, MovePOSWhereError, SubstWordConfusionError, SubstWordConfusionNewPOSError, SubstSpecificWordConfusionError and SubstWrongFormError. Frequencies are extracted, and errors occurring only once are excluded.

Three classifiers are trained. The first is trained on corrected CLC sentences (the grammatical section of the training set) and original CLC sentences (the ungrammatical section). The second classifier is trained on corrected CLC sentences and the sentences that are generated from the corrected CLC

[3] Wagner et al. (2009) report accuracy figures in the range 55–70% for their various classifiers (when tested on synthetic test data), but the best performance is obtained by combining parser-output and n-gram POS frequency features using decision trees in a voting scheme.

[4] http://www.cs.waikato.ac.nz/ml/weka/

Old-Ungram-BNC	New-Ungram-BNC
Biogas ***production production*** *is growing rapidly*	*Biogas* ***productions*** *is growing rapidly*
Emil ***as*** *courteous and helpful*	*Emil courteous and* ***was*** *helpful*
I knows *what makes you tick*	*I know what* ***make*** *you tick*
He did n't bother to lift his eyes from the ***task hand***	*He did n't bother* ***lift*** *his eyes from the task at hand*

Table 1: Examples from two synthetic BNC sets

Training Data	Precision	Recall	Accuracy	Accuracy on Grammatical
BNC/old-ungram-BNC	95.5	37.0	39.8	76.8
BNC/new-ungram-BNC	94.9	51.6	52.4	63.2

Table 2: Spoken Language Learner Corpus Classification Experiment

sentences using GenERRate (we call these "faux-CLC"). The third is trained on corrected CLC sentences and a 50/50 combination of CLC and faux-CLC sentences. In all experiments, the grammatical section of the training data contains 438,150 sentences and the ungrammatical section 454,337. The classifiers are tested on a held-out section of the CLC containing 43,639 corrected CLC sentences and 45,373 original CLC sentences. To train the classifiers, the *Mallet* implementation of Naive Bayes is used.[5] The features are word unigrams and bigrams, as well as part-of-speech unigrams, bigrams and trigrams. Andersen (2006) experimented with various learning algorithms and, taking into account training time and performance, found Naive Bayes to be optimal. The POS-tagging is carried out by the RASP system (Briscoe and Carroll, 2002).

4.2.2 Results

The results of the CLC classification experiment are presented in Table 3. There is a 6.2% drop in accuracy when we move from training on original CLC sentences to artificially generated sentences. This is somewhat disappointing since it means that we have not completely succeeded in replicating the CLC errors using GenERRate. Most of the accuracy drop is on the ungrammatical side, i.e. the correct/faux model classifies more incorrect CLC sentences as correct than the correct/incorrect model. This drop in accuracy occurs because some frequently occurring error types are not included in the error analysis file. One reason for the gap in coverage is the failure of the part-of-speech tagset to make some important distinctions. The corrected CLC sentences which were used to generate the faux-CLC set were tagged with the CLAWS tagset, and although more fine-grained than the Penn tagset, it does not, for example, make a distinction between mass and count nouns, a common source of error. Another important reason for the drop in accuracy are the recurrent spelling errors which occur in the incorrect CLC test set but not in the faux-CLC test set. It is promising, however, that much of the performance degradation is recovered when a mixture of the two types of ungrammatical training data is used, suggesting that artificial data could be used to augment naturally occurring training sets

[5] `http://mallet.cs.umass.edu/`

5 Limitations of GenERRate

We present three issues that make the task of generating synthetic error data non-trivial.

5.1 Sophistication of Input Format

The experiments in §4 highlight coverage issues with GenERRate, some of which are due to the simplicity of the supported error types. When linguistic context is supplied for deletion or insertion errors, it takes the form of the POS of the words immediately to the left and/or right of the target word. Lee and Seneff (2008a) analysed preposition errors made by Japanese learners of English and found that a greater proportion of errors in argument prepositional phrases (*look at him*) involved a deletion than those in adjunct PPs (*came at night*). The only way for such a distinction to be encoded in a GenERRate error analysis file is to allow *parsed* input to be accepted. This brings with it the problem, however, that parsers are less accurate than POS-taggers. Another possible improvement would be to make use

Training Data	Precision	Recall	Accuracy	Accuracy on Grammatical
Held-Out Test Data				
Correct/Incorrect CLC	69.7	42.6	61.3	80.8
Correct/Faux CLC	62.0	30.7	55.1	80.5
Correct/Incorrect+Faux CLC	69.7	38.2	60.0	82.7

Table 3: CLC Classification Experiment

of WordNet synsets in order to choose the new word in substitution errors.

5.2 Covert Errors

A covert error is an error that results in a syntactically well-formed sentence with an interpretation different from the intended one. Covert errors are a natural phenomenon, occurring in real corpora. Lee and Seneff (2008b) give the example *I am preparing for the exam* which has been annotated as erroneous because, given its context, it is clear that the person meant to write *I am prepared for the exam*. The problems lie in deciding what covert errors should be handled by an error detection system and how to create synthetic data which gets the balance right.

When to avoid: Covert errors can be produced by GenERRate as a result of the sparse linguistic context provided for an error in the error analysis file. An inspection of the new-ungram-BNC set shows that some error types are more likely to result in covert errors. An example is the *SubstWrongFormError* when it is used to change a noun from singular to plural. This results in the sentence *But there was no sign of Benny's father* being changed to the well-formed but more implausible *But there was no sign of Benny's **fathers***. The next version of GenERRate should include the option to change the form of a word *in a certain context*.

When not to avoid: In the design of GenERRate, particularly in the design of the *SubstWrongFormError* type, the decision was made to exclude tense errors because they are likely to result in covert errors, e.g. *She walked home → She **walks** home*. But in doing so we also avoid generating examples like this one from the spoken language learner corpus: *When I was a high school student, I **go** to bed at one o'clock*. These tense errors are common in L2 data and their omission from the faux-CLC training set is one of the reasons why the performance of this model is inferior to the real-CLC model.

5.3 More complex errors

The learner corpora contain some errors that are corrected by applying more than one transformation. Some are handled by the *SubstWrongFormError* type (*I spend a long time **to fish** →I spend a long time **fishing***) but some are not (*She is one of **reason** I became interested in English → She is one of **the reasons** I became interested in English*).

6 Conclusion

We have presented GenERRate, a tool for automatically introducing syntactic errors into sentences and shown how it can be useful for creating synthetic training data to be used in grammatical error detection research. Although we have focussed on the binary classification task, we also intend to test GenERRate in targeted error detection. Another avenue for future work is to explore whether GenERRate could be of use in the automatic generation of language test items (Chen et al., 2006, for example). Our immediate aim is to produce a new version of GenERRate which tackles some of the coverage issues highlighted by our experiments.

Acknowledgments

This paper reports on research supported by the University of Cambridge ESOL Examinations. We are very grateful to Cambridge University Press for giving us access to the Cambridge Learner Corpus and to James Hunter from Gonzaga College for supplying us with the spoken language learner corpus. We thank Ted Briscoe, Josef van Genabith, Joachim Wagner and the reviewers for their very helpful suggestions.

References

Øistein E. Andersen. 2006. Grammatical error detection. Master's thesis, Cambridge University.

Øistein E. Andersen. 2007. Grammatical error detection using corpora and supervised learning. In Ville Nurmi and Dmitry Sustretov, editors, *Proceedings of the 12th Student Session of the European Summer School for Logic, Language and Information*, Dublin.

Johnny Bigert, Jonas Sjöbergh, Ola Knutsson, and Magnus Sahlgren. 2005. Unsupervised evaluation of parser robustness. In *Proceedings of the 6th CICling*, Mexico City.

Johnny Bigert. 2004. Probabilistic detection of context-sensitive spelling errors. In *Proceedings of the 4th LREC*, Lisbon.

Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd LREC*, Las Palmas.

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st COLING and the 44th ACL*, Sydney.

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

Chia-Yin Chen, Liou Hsien-Chin, and Jason S. Chang. 2006. Fast — an automatic generation system for grammar tests. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, Sydney.

Rachele De Felice and Stephen G. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd COLING*, Manchester.

Jennifer Foster. 2007. Treebanks gone bad: Parser evaluation and retraining using a treebank of ungrammatical sentences. *International Journal on Document Analysis and Recognition*, 10(3-4):129–145.

Michael Gamon, Jianfeng Gao, Chris Brockett, Alexandre Klementiev, William B. Dolan, Dmitriy Belenko, and Lucy Vanderwende. 2008. Using contextual speller techniques and language modelling for ESL error correction. In *Proceedings of the 3rd IJCNLP*, Hyderabad.

Roger Garside, Geoffrey Leech, and Geoffrey Sampson, editors. 1987. *The Computational Analysis of English: a Corpus-Based Approach*. Longman, London.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129.

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

John Lee and Stephanie Seneff. 2008a. An analysis of grammatical errors in non-native speech in English. In *Proceedings of the 2008 Spoken Language Technology Workshop*, Goa.

John Lee and Stephanie Seneff. 2008b. Correcting misuse of verb forms. In *Proceedings of the 46th ACL*, Columbus.

Vanessa Metcalf and Detmar Meurers. 2006. Towards a treatment of word order errors: When to use deep processing – and when not to. Presentation at the NLP in CALL Workshop, CALICO 2006.

Daisuke Okanohara and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *Proceedings of the 45th ACL*, Prague.

Beatrice Santorini. 1991. Part-of-speech tagging guidelines for the Penn Treebank project. Technical report, University of Pennsylvania, Philadelphia, PA.

Jonas Sjöbergh and Ola Knutsson. 2005. Faking errors to avoid making errors. In *Proceedings of RANLP 2005*, Borovets.

Noah A. Smith and Jason Eisner. 2005a. Contrastive Estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd ACL*, Ann Arbor.

Noah A. Smith and Jason Eisner. 2005b. Guiding unsupervised grammar induction using contrastive estimation. In *Proceedings of the IJCAI Workshop on Grammatical Inference Applications*, Edinburgh.

Guihua Sun, Xiaohua Liu, Gao Cong, Ming Zhou, Zhongyang Xiong, John Lee, and Chin-Yew Lin. 2007. Detecting erroneous sentences using automatically mined sequential patterns. In *Proceedings of the 45rd ACL*, Prague.

Joel R. Tetreault and Martin Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd COLING*, Manchester.

Joachim Wagner, Jennifer Foster, and Josef van Genabith. 2007. A comparative evaluation of deep and shallow approaches to the automatic detection of common grammatical errors. In *Proceedings of the joint EMNLP/CoNLL*, Prague.

Joachim Wagner, Jennifer Foster, and Josef van Genabith. 2009. Judging grammaticality: Experiments in sentence classification. *CALICO Journal*. Special Issue on the 2008 Automatic Analysis of Learner Language CALICO Workshop. To Appear.