High-Order Sequence Modeling for Language Learner Error Detection

Michael Gamon
Microsoft Research
One Microsoft Way
Redmond, WA 98052
mgamon@microsoft.com

Abstract

We address the problem of detecting English language learner errors by using a discriminative high-order sequence model. Unlike most work in error-detection, this method is agnostic as to specific error types, thus potentially allowing for higher recall across different error types. The approach integrates features from many sources into the error-detection model, ranging from language model-based features to linguistic analysis features. Evaluation results on a large annotated corpus of learner writing indicate the feasibility of our approach on a realistic, noisy and inherently skewed set of data. High-order models consistently outperform low-order models in our experiments. Error analysis on the output shows that the calculation of precision on the test set represents a lower bound on the real system performance.

1. Introduction

Systems for automatic detection and correction of errors in native writing have been developed for many decades. Early in the development of these systems, the approach was exclusively based on knowledge engineering. Hand-crafted grammars would analyze a sentence and would contain special mechanisms for rule or constraint relaxation that allow ungrammatical sentences to produce a parse, while at the same time indicating that a grammatical error is present. More recently, data-driven methods have assumed prominence and there has been an emerging area of research into the challenge of detecting and correcting errors in learner language (for an overview see Leacock et al. 2010). Data-driven methods offer the familiar set of advantages: they can be more flexible than a manually maintained set of rules and they tend to cope better with noisy input. Drawbacks include the inability to handle linguistically more complex errors that involve long distance dependencies such as subject-verb agreement. Learner errors as a target for error detection and correction pose a particular challenge but also offer some unique opportunities. The challenge lies in the density of errors (much higher than in native writing), the variety of errors (a superset of typical native errors) and the generally more non-idiomatic writing. On the other hand, the availability of annotated corpora, often comprised of manually corrected learner essays or scripts, provides a big advantage for the evaluation and training of data-driven systems.

Data-driven systems for English learner error detection and correction typically target a specific set of error types and contain a machine learned component for each error type. For example, such a system may have a classifier that determines the correct choice of preposition given the lexical and syntactic part-of-speech (POS) context and hence can aid the learner with the notoriously difficult problem of identifying an appropriate preposition. Similarly, a classifier can be used to predict the correct choice of article in a given context. Such targeted systems have the advantage that they often achieve relatively high precision at, of course, the cost of recall. However, while there are a few major learner error categories, such as prepositions and articles, there is also a long tail of content word and other errors that is not amenable to a targeted approach.

In this paper, we depart from the error-specific paradigm and explore a sequence modeling approach to general error detection in learner writing. This approach is completely agnostic as to the error type. It attempts to predict the location of an error in a sentence based on observations gathered from a supervised training phase on an error-annotated learner corpus. Features used here are based on an *n*-gram language model, POS tags, simple string features that indicate token length and capitalization, and linguistic analysis by a constituency parser. We train and evaluate the method on a sizeable subset of the corpus. We show the contribution of the different feature types and perform a manual error analysis to pinpoint shortcomings of the system and to get a more accurate idea of the system's precision.

2. Related work

Error-specific approaches comprise the majority of recent work in learner error detection. Two of the most studied error types in learner English are preposition and article errors since they make up a large percentage of errors in learner writing (16% and 13% respectively in the *Cambridge Learner Corpus*, without considering spelling and punctuation errors). The most widely used approach for detecting and correcting these errors is classification, with lexical and POS features gleaned from a window around the potential preposition/article site in a sentence. Some recent work includes Chodorow et al. (2007), De Felice and Pulman (2008), Gamon (2010), Han et al. (2010), Izumi et al. (2004), Tetreault and Chodorow (2008), Rozovskaya and Roth (2010a, 2010b). Gamon et al. (2008) and Gamon (2010) used a language model in addition to a classifier and combined the classifier output and language model scores in a meta-classifier. These error-specific methods achieve high precision (up to 80-90% on some corpora) but only capture highly constrained error types such as preposition and determiner errors.

There has also been research on error-detection methods that are not designed to identify a specific error type. The basic idea behind these error-agnostic approaches is to identify an error where there is a particularly unlikely sequence compared to the patterns found in a large well-formed corpus. Atwell (1986) used low-likelihood sequences of POS tags as indicators for the presence of an error. Sjöbergh (2005) used a chunker to detect unlikely chunks in native Swedish writing compared to the chunks derived from a large corpus of well-formed Swedish writing. Bigert and Knutsson (2002) employed a statistical method to identify a variety of errors in Swedish writing as rare sequences of morpho-syntactic tags. They significantly reduced false positives by using additional methods to determine whether the unexpected sequence is due to phrase or sentence boundaries or due to rare single tags. Chodorow and Leacock (2000) utilized mutual information and chi-square statistics to identify typical contexts for a small set of targeted words from a large well-formed corpus. Comparing these statistics to the ones found in a novel sentence, they could identify unlikely contexts for the targeted words that were often good indicators of the presence of an error. Sun et al. (2007) mined for patterns that consist of POS tags and function words. The patterns are of variable length and can also contain gaps. Patterns were then combined in a classifier to distinguish correct from erroneous sentences. Wagner et al. (2007) combined parse probabilities from a set of statistical parsers and POS tag *n*-gram probabilities in a classifier to detect ungrammatical sentences. Okanohara and Tsujii (2007) differed from the previous approaches in that they directly used discriminative language models to distinguish correct from incorrect sentences, without the direct modeling of error-indicating patterns. Park and Levy (2011) use a noisy channel model with a base language model and a set of error-specific noise models for error detection and correction.

In contrast to previous work, we cast the task as a sequence modeling problem. This provides a flexible framework in which multiple statistical and linguistic signals can be combined and calibrated by supervised learning. The approach is error-agnostic and can easily be extended with additional statistical or linguistic features.

3. Error detection by sequence modeling

Errors consist of a sub-sequence of tokens in a longer token sequence. They can be identified by a combination of internal and contextual features, the latter requiring a notion of Markov window (a window around a token in which relevant information is likely to be found). This is similar to tasks such as named entity recognition (NER) or part-of-speech tagging, where sequence modeling has proven to be successful.

We choose a Maximum Entropy Markov Model (MEMM, McCallum et al. 2000) as the modeling technique. In NER, the annotation convention uses three labels for a token “O” (outside of NE), “B” (beginning of NE), and “I” (inside of NE). For our purpose we reduced the set of labels to just “O” and “I” since most of the errors are relatively short.

Conditional Random Fields (Lafferty et al. 2001) are considered to be superior to MEMMs in learning problems affected by label bias (Bottou 1991). In our scheme, however, there are only two states “O” and “I”, and both states can transition to each other. Since there are no states with asymmetric transition properties that would introduce a bias towards states with fewer transitions, label bias is not a problem for us.

Figure 1 shows the structure of our MEMM with a Markov order of five (the diagram only shows the complete set of arcs for the last state). The input sentence contains the token sequence *the past year I was stayed …* with the error *was stayed*. Instead of using the tokens themselves as observations, we chose to use POS tags assigned by an automatic tagger (Toutanova et al. 2003). This choice was motivated by data sparseness. Learning a model that observes individual lexical items and predicts a sequence of error/non-error tags would be ideal, but given the many different error types and triggering contexts for an error, such a model would require much more training data. A large set of features that serve as constraints on the state transition models are extracted for each state. These features are described in Section 5.

Note that the model structure would lend itself to a factorial conditional random field (McCallum et al. 2003) which allows the joint labeling of POS tags and state labels. This would, however, require training data that is labeled for both errors and POS tags.

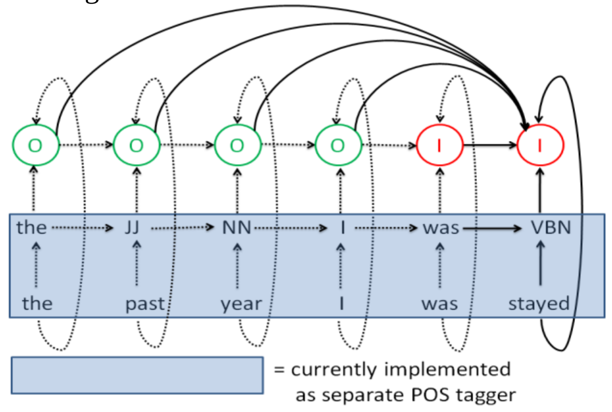

Figure 1: MEMM model for error detection, the full set of dependencies is only shown for the last state.

4. Detecting errors in the Cambridge Learner Corpus

The learner corpus used to train and evaluate the system is the *Cambridge Learner Corpus* (CLC). It consists of essays (scripts) written as part of the University of Cambridge *English for Speakers of Other Languages* (ESOL) examinations. The corpus contains about 30 million words of learner English. All errors are annotated and include, when possible, a single suggested correction. Errors are categorized into 87 error types.

We performed a number of preprocessing steps on the data. On the assumption that learners have access to a spell checker, errors that were marked as spelling errors were corrected based on the annotations. Confused words (*their/there*) were treated in the same way, given that they are corrected by a modern proofing tool such as the one in Microsoft Word. In addition, British English spelling conventions were changed to those of American English. Sentences containing errors that had no suggested rewrite were eliminated. Finally, only lexical errors are covered in this work. For punctuation and capitalization we removed the error annotations, retaining the original (erroneous) punctuation and capitalization.

We grouped the remaining 60 error classifications into eight categories: Content word, Inflectional morphology, Noun phrase errors, Preposition errors, Multiple errors, Other errors involving content words, Other errors involving function words and Derivational morphology. The distribution of error categories is shown in Table 1.

Error Class	**Freq**	**Pct**
Content word insertion, deletion or choice	185,201	21%
Inflectional morphology and agreement of content words	157,660	18%
Noun phrase formation: Determiners and quantifiers	130,829	15%
Preposition error	124,902	14%
Multiple: Adjacent and nested annotations	113,615	13%
Other content word errors	79,596	9%
Other function word errors: anaphors and conjunctions	65,034	7%
Derivational morphology of content words	39,213	4%

Table 1: Error types in the CLC.

The *multiple* error class includes any combination of error types where the error annotations are either nested or adjacent. The other categories are more focused: the errors are of a particular class and their adjacent context is correct, although there may be another error annotation a single token away. *Content word* errors involve the insertion, deletion and substitution of nouns, verbs, adjectives and adverbs. Further analysis of this error category on a random sample of 200 instances reveals that the majority (72%) of content word errors involve substitutions, while deletions account for 10% of the errors and insertions for 18%. Most substitutions (63%) involve the wrong choice of a word that is somewhat semantically related to the correct choice. *Inflectional morphology* includes all inflection errors for content words as well as subject-verb agreement errors. The inflectional errors include many cases of what might be considered spelling errors, for example **dieing/dying*. Similarly, the *derivational morphology* errors include all derivational errors for content words – and also include many errors that may be considered as spelling errors. *Noun formation* errors include all annotations involving determiners and quantifiers: inflection, derivation, countability, word form and noun-phrase-internal agreement. *Preposition* errors include all annotations that involve prepositions: insertion, deletion, substitution and a non-preposition being used in place of a preposition. There are two other categories: those involving the remaining function words (anaphors and conjunctions) and those involving remaining content words (collocation, idiom, negative formation, argument structure, word order, etc.).

It is important to highlight the challenges inherent in this data set. First of all, the problem is highly skewed since only 7.3% of tokens in the test set are involved in an error. Second, since we included correct learner sentences in the development and test sets in the proportion they occur in the overall corpus, only 47% of sentences in the test set contain error annotations, greatly increasing the likelihood of false positives.

5. Features

5.1 Language model features

The language model (LM) features comprise a total of 29 features. Each of these features is calculated from *n*-gram probabilities observed at and around the current token. All LM features are based on scores from a 7-gram language model with absolute discount smoothing built from the Gigaword corpus (Gao et al. 2001, Nguyen et al. 2007).

We group the language model features conceptually into five categories: basic features, ratio features, drop features, entropy delta features and miscellaneous. All probabilities are log probabilities, and *n* in the *n*-grams ranges from 1 to 5. All features are calculated for each token *w* of the tokens $w_0...w_i$ in a sentence.

Basic LM features consist of two features: the unigram probability of *w* and the average *n*-gram probability of all *n*-grams in the sentence that contain *w*.

Ratio features are based on the intuition that errors can be characterized as involving tokens that have a very low ratio of higher order *n*-gram probabilities to lower order *n*-gram probabilities. In other words, these are tokens that are part of an unlikely combination of otherwise likely smaller *n*-grams. These features are calculated as the ratio of the average *x*-gram probability of all *x*-grams containing *w* to the average *y*-gram probability of all *y*-grams containing *w*. The values for *x* and *y* are: 5 and 1, 4 and 1, 3 and 1, 2 and 1, 5 and 4, 4 and 3, 3 and 2.

Drop features measure either the drop or increase in *n*-gram probability across token *w*. For example, the *bigram drop* at w_i is the delta between the bigram probability of the bigram starting at *i-1* to the bigram probability of the bigram starting at *i*. Drop features are calculated for *n*-grams with $2 \leq n \leq 5$.

Entropy delta features offer another way to look at the changes of *n*-gram probability across a token *w*. *Forward entropy* for w_i is defined as the entropy of the string $w_i...w_n$ where *n* is the index of the last token in the sentence. We calculate the entropy of an *n*-gram as the language model probability of string $w_i...w_n$ divided by the number of tokens in that string. *Backward entropy* is calculated analogously for $w_0...w_i$. For *n*-grams with $1 \leq n \leq 5$, we also calculate, at each index *i* into the token array, the delta between the *n*-gram entropy of the *n*-gram starting at *i* and the *n*-gram starting at *i+1* (*forward sliding entropy*). Similarly the delta between the *n*-gram entropy of the *n*-gram starting at *i* and the *n*-gram starting at *i-1* (*backward sliding entropy*) is calculated.

There are four miscellaneous language model features. Three of them, *minimum ratio to random*, *average ratio to random*, and *overall ratio to random* address the fact that a "good" *n*-gram is likely to have a much higher probability than an *n*-gram with the same tokens in random order. For all *n*-grams where $2 \leq n \leq 5$ we calculate the ratio between the *n*-gram probability and the sum of the unigram probabilities. For a token w_i we produce the *minimum ratio to random* (the minimum ratio of all *n*-grams including *w*) and the *average ratio to random* (the average of all ratios of the *n*-grams including *w*). *Overall ratio to random* is obtained by looping through each *n*-gram where $2 \leq n \leq 5$ that includes w_i and summing the *n*-gram probabilities (*sum1*) as well as the unigram probabilities of all unigrams in these *n*-grams (*sum2*). The ratio feature is then *sum1/sum2*. The final feature addresses the intuition that an erroneous word may cause *n*-grams that contain the word to be less likely than adjacent but non-overlapping *n*-grams. *Overlap to adjacent ratio* is the sum of probabilities of *n*-grams including w_i, divided by the sum of probabilities of *n*-grams that are adjacent to w_i but do not include it.

Note that this use of a host of language model features is substantially different from using a single language model score on hypothesized error and potential correction to filter out unlikely correction candidates as in Gamon et al. (2008) and Gamon (2010).

5.2 String features

String features capture information about the characters in a token and the tokens in a sentence. Two binary features indicate whether a token is capitalized (initial capitalization or all capitalized), one feature indicates the token length in characters and one feature measures the number of tokens in the sentence.

5.3 Linguistic Analysis features

Each sentence is linguistically analyzed by a PCFG-LA parser (Petrov et al., 2006) trained on the Penn Treebank (Marcus et al., 1993). A number of features are extracted from the constituency tree to assess the syntactic complexity of the whole sentence, the syntactic complexity of the local environment of a token, and simple constituency information for each token. These features are: label of the parent and grandparent node, number of sibling nodes, number of siblings of the parent, presence of a governing head node, label of the governing head node, and length of path to the root. An additional feature indicates whether the POS tag assigned by the parser does not match the tag assigned by the POS tagger, which may indicate a tagging error.

6. Experiments

6.1 Design

For our experiments we use three different mutually exclusive random subsets of CLC. 50K sentences are used for training of the models (larger data sets exceeded the capabilities of our MEMM trainer). In this set, we only include sentences that contain at least one annotated error. We also experimented using a mix of error-free and erroneous sentences, but the resulting models turned out to be extremely skewed towards always predicting the majority state "O" (no error). 20K sentences (including both erroneous and correct sentences) are used for parameter tuning and testing, respectively.

Each token in the data is annotated with one of the states "O" or "I". Performance is measured on a per token basis, i.e. each mismatch between the predicted state and the annotated state is counted as an error, each match is counted as a correct prediction.

We use the development set to tune two parameters: the size of the Markov window and a prior to prevent overfitting. The latter is a Gaussian prior (or quadratic regularizer) where the mean is fixed to zero and the variance is left as a free parameter. We perform a grid search to find values for the parameters that optimize the model's *F1* score on the development data.

In order to be able to report precision and recall curves, we use a technique similar to the one described in Minkov et al. (2010): we introduce an artificial feature with a constant value at training time. At test time we perform multiple runs, modifying the weight on the artificial feature. This weight variation influences the model's prior propensity to assign each of the two states, allowing us to measure a precision/recall tradeoff.

6.2 Performance of feature sets

Figure 2 illustrates the performance of three different feature sets and combinations. The baseline is using only language model features and standard POS tags, which tops out at about 20% precision. Adding the string features discussed in the previous section, and *partially lexicalized* (*PL*) POS tags, where we used POS tags for content word tokens and the lexicalized token for function words, we get a small but consistent improvement. We obtain the best performance when all features are used, including the linguistic analysis features (DepParse). We found that a high-order model with a Markov window size of 14 performed best for all experiments with a top *F1* score. *F1* at lower orders was significantly worse. Training time for the best models was less than one hour.

6.3 Predicting error types

In our next experiment, we tried to determine how the sequence modeling approach performs for individual error types. Here we trained eight different models, one for each of the error types in Table 1. As in the previous experiments, the development and test files contained error-free sentences. The optimal Markov window size ranged from 8 to 15. Note that our general sequence model described in the previous sections does not recognize different error types, so it was necessary to train one model per error type for the experiments in this section.

Figure 3 shows the results from this series of experiments. We omit the results for *other content word error, other function word* and *multiple errors* in this graph since these relatively ill-defined error classes performed rather poorly. As Figure 3 illustrates, derivational errors and preposition errors achieve by far the best results. The fact that the individual precision never reaches the level of the general sequence model (Figure 2) can be attributed to the much smaller overall set of errors in each of the eight training sets. In Figure 4 we compare the sequence modeling results for prepositions with results from the preposition component of the current version of the system described in Gamon (2010) on the same test set. That system consists of a preposition-specific classifier, a language model and a meta-classifier that combines evidence from the classifier and the language model. The sequence model approach outperforms the classifier of that system, but the full system including language model and meta-classifier achieves much higher precision than the sequence modeling approach.

6.4 Learning curve experiments

An obvious question that arises is how much training data we need for an error detection sequence model, i.e. how does performance degrade as we decrease the amount of training data from the 50K error-annotated sentences that were used in the previous experiments. To this end we produced random subsets of the training data in 20% increments. For each of these training sets, we determined the resulting *F1* score by first performing parameter tuning on the development set and then measuring precision and recall of the best model on the test set. Results are shown in Figure 5: at 20% of training data, precision starts to increase at the cost of recall. At 80% of the training data, recall starts to trend up as well. This upward trend of both precision and recall indicates that increasing the amount of training data is likely to further improve results.

6.5 Error analysis

The precision values obtained in our experiments are low, but they are also based on the strictest possible measure of accuracy: an error prediction is only counted as correct if it exactly matches a location and annotation in the CLC. A manual analysis of 400 randomly selected sentences containing “false positives”, where the system had 29% precision and 10% recall, by the strictest calculation, showed that 14% of the “false positives” identified an error that was either not annotated in CLC or was an error type not covered by the system such as punctuation or case (recall from Section 4 that for these errors we removed the error annotations but retained the original string). An additional 16% were adjacent to an error annotation. 12% had error annotations within 2-4 tokens from the predicted error. Foreign language and other unknown proper names comprised an additional 6%. Finally, 9% were due to tokenization problems or all-upper case input that throws off the POS tagger. Thus the precision reported in Figure 2 through Figure 6 is really a lower bound. 30% of the “false positives” either identify, or are adjacent to, an error.

Sentence length has a strong influence on the accuracy of the sequence model. For sentences less than 7 tokens long, average precision is approximately 7%, whereas longer sentences average at 29% precision. This observation fits with the fact that high-order models perform best in the task, i.e. the more context a model can access, the more reliable its predictions are. Shorter sentences are also less likely to contain an error: only 12% of short sentences contain an error, as opposed to 46% of sentences of seven tokens or longer.

For sentences that are at least 7 tokens long, error predictions on the first and last two tokens (the last token typically being punctuation) have an average precision of 22% as compared to an average of 30% at all other positions. Other unreliable error predictions include those involving non-alphabetic characters (quotes, parentheses, symbols, numbers) with 1% precision and proper name tags with 10% precision. Many of the predictions on NNP tags identify, by and large, unknown or foreign names (*Cricklewood*, *Cajamarca*). Ignoring system flags on short sentences, symbols and NNP tags would improve precision with little cost to recall.

We also experimented with a precision/recall metric that is less harsh but at the same time realistic for error detection. For this "soft metric" we count correct and incorrect predictions at the error level instead of the token level. An error is defined as a consecutive sequence of *n* error tags, where $n \geq 1$.

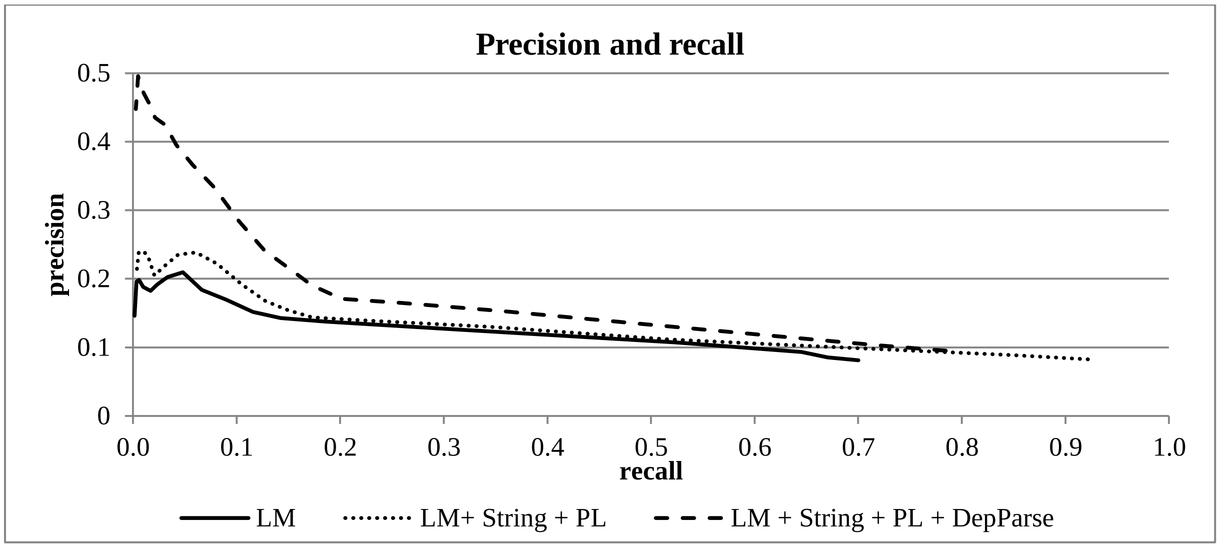

Figure 2: Precision and recall of different feature sets.

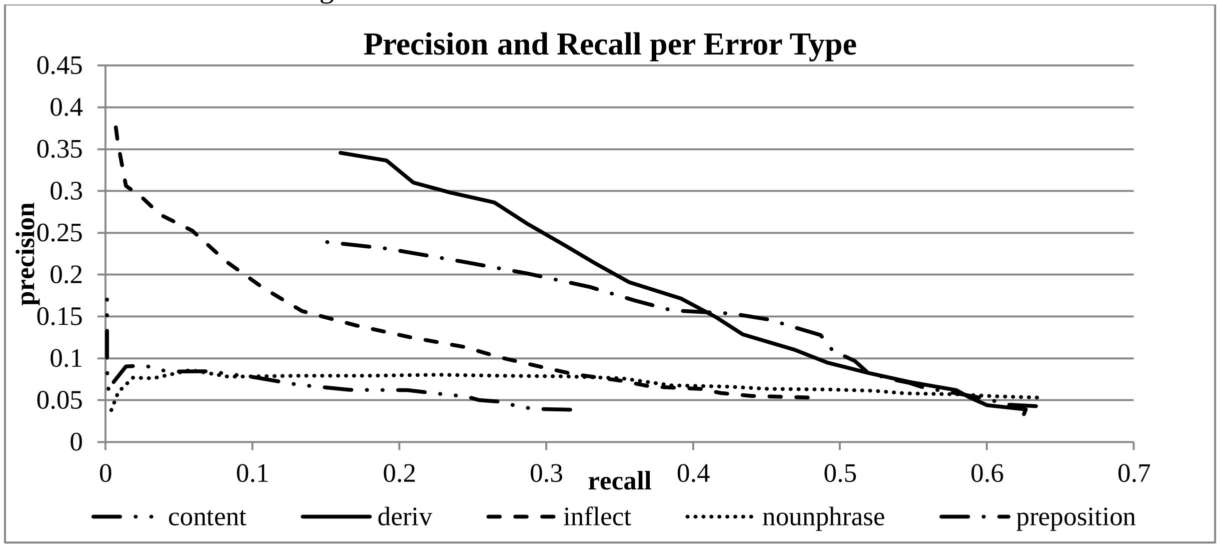

Figure 3: Precision and recall of different error models.

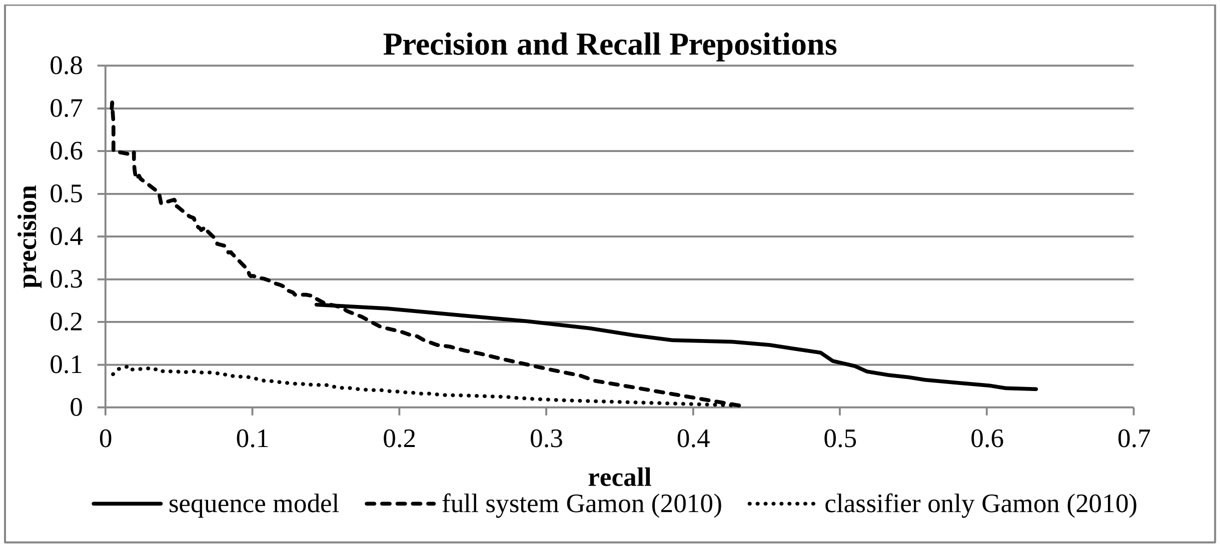

Figure 4: Preposition precision and recall.

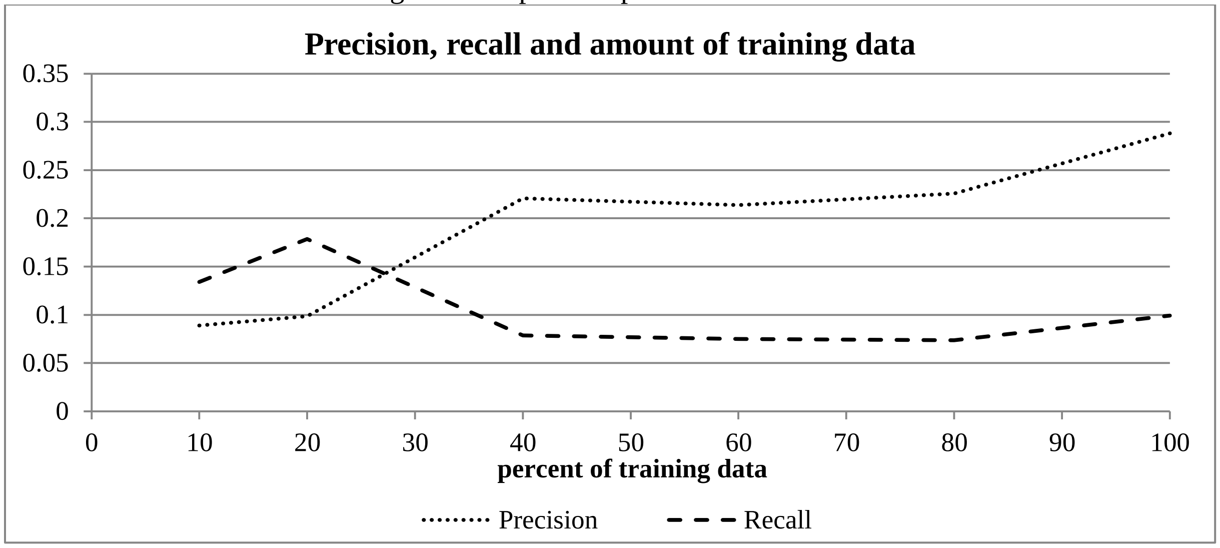

Figure 5: Learning curve.

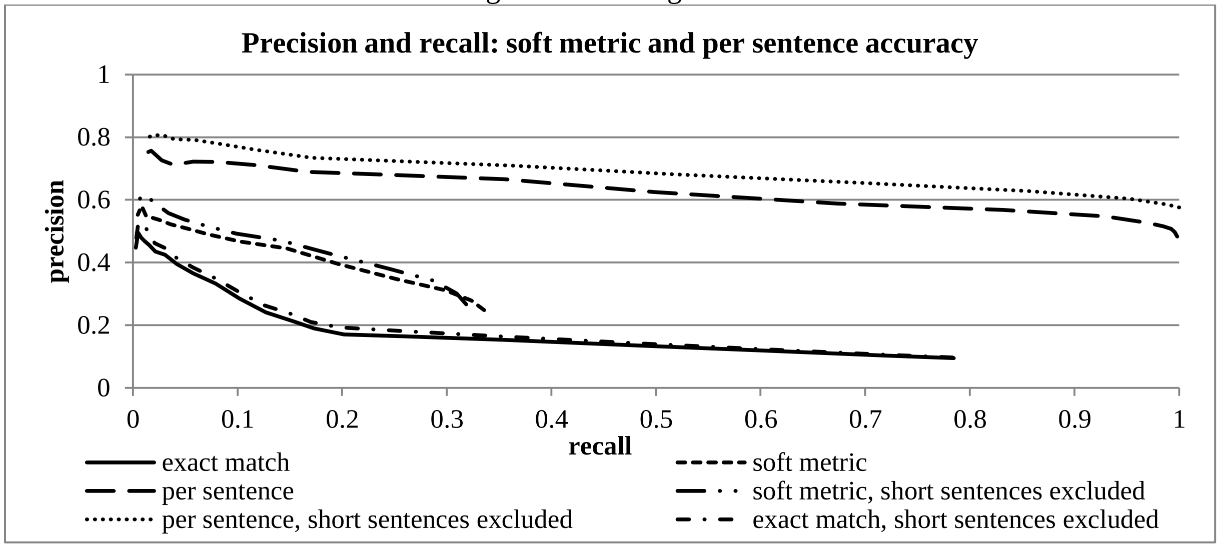

Figure 6: Precision and recall for adjacent annotated error

A predicted error counts as being correct with respect to an annotated error if the following two criteria are met:

a) At least one predicted error token is part of an annotated error or is directly adjacent to an annotated error
b) No more than two predicted error tokens fall outside the annotated error.

Criterion (a) establishes that predicted and annotated error are overlapping or at least directly adjacent. Criterion (b) ensures that the predicted error is “local” enough to the annotated error and does not include too much irrelevant context, but it still allows an annotated error to be flanked by predicted error tokens. Figure 6 illustrates the precision/recall characteristics of the best model when using this soft metric as compared to the strict metric. We also included a “per sentence” metric in Figure 6, where we measure precision and recall at the level of identifying a sentence as containing an error or not, in other words when using the model as a detector for ungrammatical sentences. In addition we show for each of the three metrics how the results change if short sentences (shorter than 7 tokens) are excluded from the evaluation.

7. Conclusion and future work

We have shown that a discriminative high order sequence model can be used to detect errors in English learner writing. This enables a general approach to error detection, at the cost of requiring annotated data. High-order models outperform lower order models significantly for this problem.

It is obvious that there are several avenues to pursue in order to improve upon these initial results. Two possibilities that we would like to highlight are the model structure and the feature set. As mentioned in Section 3, instead of using a separate POS tagger we could follow McCallum et al. (2003) and design a model that jointly predicts two sequences: POS tags and error tags. As for feature sets, we conducted some preliminary additional experiments where we added a second set of language model features, based on a different language model, namely the Microsoft web n-gram model (Wang et al. 2010). The addition of these features raised both precision and recall.

Finally, an error detection system is only of practical use if it is combined with a component that suggests possible corrections. For future work, we envision a combination of generic error detection with a corpus-based lookup system that finds alternative strings that have been observed in similar contexts. All these alternatives can then be scored by a language model in the original context of the user input, allowing only those suggestions to be shown to the user that achieve a better language model score than the original input. This combination of error detection and error correction has the advantage that the error detection component can be used to provide recall, i.e. it can be allowed to operate at a lower precision level. The error correction component, on the other hand, then reduces the number of false flags by vetting potential corrections by language model scores.

Acknowledgments

We would like to thank Claudia Leacock for the manual error analysis, Michel Galley for detailed comments on an earlier draft and Chris Quirk for discussions and help around the MEMM model implementation. The idea of the *ratio to random* language model features is Yizheng Cai’s. We also greatly benefited from the comments of the anonymous reviewers.

References

Eric Steven Atwell. 1986. How to detect grammatical errors in a text without parsing it. In *Proceedings of EACL*, pp. 38-45.

Léon Bottou. 1991. Une approche théorique de l’apprentissage connexionniste: Applications à la reconnaissance de la parole. Doctoral dissertation, Université de Paris XI.

Johnny Bigert and Ola Knutsson. 2002. Robust error detection: a hybrid approach combining unsupervised error detection and linguistic knowledge. In *Proceedings of the Second Workshop on Robust Methods in Analysis of Natural Language Data*, pp. 10-19.

Martin Chodorow and Claudia Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proceedings of NAACL*, pp. 140-147.

Martin Chodorow, Joel Tetreault and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions*, pp. 25-30.

Rachele De Felice and Stephen G. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of COLING*, pp. 169-176.

Michael Gamon, Jianfeng Gao, Chris Brockett, Alexandre Klementiev, William Dolan, Dmitriy Belenko and Lucy Vanderwende. 2008. Using Contextual Speller Techniques and Language Modeling for ESL Error Correction. In *Proceedings of IJCNLP*.

Michael Gamon. 2010. Using mostly native data to correct errors in learners' writing. In *Proceedings of NAACL*.

Jianfeng Gao, Joshua Goodman, and Jiangbo Miao. 2001. The use of clustering techniques for language modeling--Application to Asian languages. Computational Linguistics and Chinese Language Processing, 6(1), 27-60.

Na-Rae Han, Joel Tetreault, Soo-Hwa Lee and Jin-Young Ha. 2010. Using error-annotated ESL data to develop an ESL error correction system. In *Proceedings of LREC*.

Emi Izumi, Kiyotaka Uchimoto and Hitoshi Isahara. 2004. SST speech corpus of Japanese learners' English and automatic detection of learners' errors. *International Computer Archive of Modern English Journal*, 28:31-48.

John Lafferty, Andrew McCallum and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICWSM*, pp. 282-289.

Claudia Leacock, Martin Chodorow, Michael Gamon and Joel Tetreault. 2010. *Automated Grammatical Error Detection for Language Learners*. Morgan and Claypool.

Mitchell P. Marcus, Beatrice Santorini and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19:313-330.

Andrew McCallum, Dayne Freitag and Fernando Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of ICML*, pp. 591-598.

Andrew McCallum, Khashayar Rohanimanesh and Charles Sutton. 2003. Dynamic Conditional Random Fields for jointly labeling multiple sequences. In *Proceedings of NIPS Workshop on Syntax, Semantics and Statistics*.

Einat Minkov, Richard C. Wang, Anthony Tomsaic and William C. Cohen. 2010. NER systems that suit user's preferences: Adjusting the Recall-Precision trade-off for entity extraction. In *Proceedings of NAACL*, pp. 93-96.

Patrick Nguyen, Jianfeng Gao, and Milind Mahajan. 2007. MSRLM: A scalable language modeling toolkit (MSR-TR-2007-144). Redmond, WA: Microsoft.

Daisuke Okanohara and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *Proceedings of ACL*, pp. 73-80.

Y. Albert Park and Roger Levy. 2011. Automated whole sentence grammar correction using a Noisy Channel Model. In *Proceedings of ACL 2011*.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING/ACL*, pp. 443-440.

Alla Rozovskaya and Dan Roth. 2010a. Training Paradigms for correcting errors in grammar and usage. In *Proceedings of NAACL-HLT*.

Alla Rozovskaya and Dan Roth. 2010b. Generating confusion sets for context-sensitive error correction. In *Proceedings of EMNLP*.

Jonas Sjöbergh. 2005. Chunking: An unsupervised method to find errors in text. In *Proceedings of the 15th NODALIDA conference*.

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 ACL*, pp. 81-88.

Joel Tetreault and Martin Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of COLING*, pp. 865-872.

Kristina Toutanova, Dan Klein, Chris Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL*, pp. 252-259.

Joachim Wagner, Jennifer Foster, and Josef van Genabith. 2007. Judging grammaticality: Experiments in sentence classification. *In Proceedings of EMNLP & CONLL*, pp 112-121.

Kuansan Wang, Christopher Thrasher, Evelyne Viegas, Xialong Li, and Paul Hsu. 2010. An Overview of Microsoft web n-gram corpus and applications. In: *Proceedings of NAACL 2010*.