A Fast Algorithm for Words Reordering Based on Language Model

Theologos Athanaselis, Stelios Bakamidis, and Ioannis Dologlou

Institute for Language and Speech Processing
Artemidos 6 and Epidavrou, GR-15125,
Maroussi, Greece
Tel.: +302106875300, Fax: +302106854270
{tathana, bakam, ydol}@ilsp.gr
http://www.ilsp.gr

Abstract. What appears to be given in all languages is that words can not be randomly ordered in sentences, but that they must be arranged in certain ways, both globally and locally. The "scrambled" words into a sentence cause a meaningless sentence. Although the use of manually collected grammatical rules can boost the performance of grammar checker in word order diagnosis, the repairing task is still very difficult. This work proposes a method for repairing word order errors in English sentences by reordering words in a sentence and choosing the version that maximizes the number of trigram hits according to a language model. The novelty of this method concerns the use of a permutations' filtering approach in order to reduce the search space among the possible sentences with reordered words. The filtering method is based on bigrams' probabilities. In this work the search space is further reduced using a threshold over bigrams' probabilities. The experimental results show that more than 95% of the test sentences can be repaired using this technique. The comparative advantage of this method is that it is not restricted into a specific set of words, and avoids the laborious and costly process of collecting word order errors for creating error patterns. Unlike most of the approaches, the proposed method is applicable to any language (language models can be simply computed in any language) and does not work only with a specific set of words. The use of parser and/or tagger is not necessary.

1 Introduction

Automatic grammar checking is traditionally done by manually written rules, constructed by computer linguists. Methods for detecting grammatical errors without manually constructed rules have been presented before. Atwell (1987) uses the probabilities in a statistical part-of the speech tagger, detecting errors as low probability part of speech sequences. Golding (1995) showed how methods used for decision lists and Bayesian classifiers could be adapted to detect errors resulting from common spelling confusions among sets such as "there", "their" and "they're". He extracted contexts from correct usage of each confusable word in a training corpus and then identified a new occurrence as an error when it matched the wrong context.

S. Kollias et al. (Eds.): ICANN 2006, Part II, LNCS 4132, pp. 943 – 951, 2006.

Chodorow and Leacock (2000) suggested an unsupervised method for detecting grammatical errors by inferring negative evidence from edited textual corpora. Heift (1998, 2001) released the German Tutor, an intelligent language tutoring system where word order errors are diagnosed by string comparison of base lexical forms. Bigert and Knutsson (2002) presented how a new text is compared to known correct text and deviations from the norm are flagged as suspected errors. Sjobergh (2005) introduced a method of grammar errors recognition by adding errors to a lot of (mostly error free) unannotated text and by using a machine learning algorithm.

Unlike most of the approaches, the proposed method is applicable to any language (language models can be computed in any language) and does not work only with a specific set of words. The use of parser and/or tagger is not necessary. Also, it does not need a manual collection of written rules since they are outlined by the statistical language model.

The paper is organized as follows: the architecture of the entire system and a description of each component follow in section 2. The language model is described in section 3. The 4th section shows how permutations are filtered by the proposed method. The 5th section specifies the method that is used for searching valid trigrams in a sentence. The results of using WSJ experimental scheme are discussed in section 6. Finally, the concluding remarks are made in section 7.

2 System's Architecture

This work presents a new method for detecting and repairing sentences with word order errors that is based on the statistical language model (N-grams). It is straight forward that the best way for reconstructing a sentence with word order errors is to reorder the words. However, the question is how it can be achieved without knowing the attribute of each word. Many techniques have been developed in the past to cope with this problem using a grammar parser and rules. However, the success rates reported in the literature are in fact low. A way for reordering the words is to use all the possible permutations. The crucial drawback of this approach is that given a sentence with length N words the number of all permutations is N!. This number is very large and seems to be restrictive for further processing. The novelty of the proposed method concerns the use of a technique for filtering the initial number of permutations. The process of repairing sentences with word–order errors incorporates the followings tools:

- a simple, and efficient confusion matrix technique
- and language model's trigrams and bigrams.

Consequently, the correctness of each sentence depends on the number of valid trigrams. Therefore, this method evaluates the correctness of each sentence after filtering, and provides as a result, a sentence with the same words but in correct order (Figure 1).

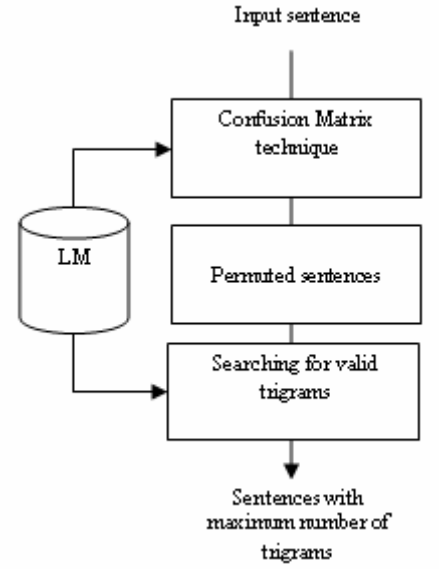

Fig. 1. System's architecture

3 Language Model

The language model (LM) that is used subsequently is the standard statistical N-grams (Young, 1996). The N-grams provide an estimate of $P(W)$, the probability of observed word sequence W. Assuming that the probability of a given word in an utterance depends on the finite number of preceding words, the probability of N-word string can be written as:

$$P(W) = \prod_{i=1}^{N} P(w_i \mid w_{i-1}, w_{i-2}, ..., w_{i-(N-1)}) \quad (1)$$

One major problem with standard N-gram models is that they must be trained from some corpus, and because any particular training corpus is finite, some perfectly acceptable N-grams are bound to be missing from it. That is, the N-gram matrix for any given training corpus is sparse; it is bound to have a very large number of cases of putative "zero probability N-grams" that should have some non zero probability. Some part of this problem is endemic to N-grams; since they can not use long distance context, they always tend to underestimate the probability of strings that happen no tot have occurred nearby in their training corpus. There are some techniques that can be used in order to assign a non zero probability to these zero probability N-grams. In this work, the language model has been trained using BNC and consists of trigrams with Good-Turing discounting (Good, 1953) and Katz back off (Katz, 1987) for smoothing. BNC contains about 6.25M sentences and 100 million words. The figure below depicts the number of bigrams of the LM (Language Model) with respect to their logarithmic probabilities. The 80% of the LM's bigrams are between -5,2 and -1,6.

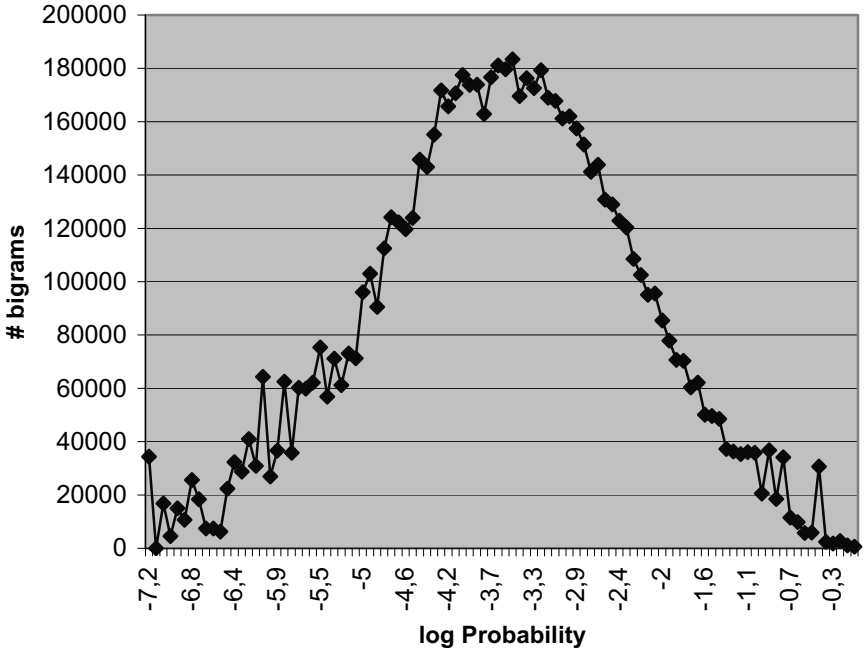

Fig. 2. The bigrams' distribution with regard to their log probabilities

4 Filtering Permutations

Considering that an ungrammatical sentence includes the correct words but in wrong order, it is plausible that generating all the permuted sentences (words reordering) one of them will be the correct sentence (words in correct order). The question here is how feasible is to deal with all the permutations for sentences with large number of words. Therefore, a filtering process of all possible permutations is necessary. The filtering involves the construction of a confusion matrix NxN in order to extract possible permuted sentences.

Given a sentence $a = [w[0], w[1],...w[n-1], w[n]]$ with N words, a confusion matrix $A \in R^{NXN}$ can be constructed.

The size of the matrix depends on the length of the sentence. The objective of this confusion matrix is to extract the valid bigrams according to the language model. The element $P[i, j]$ indicates the validness of each pair of words $(w[i]w[j])$ according to the list of language model's bigrams. If a pair of two words $(w[i]w[j])$ cannot be found in the list of language model bigrams then the corresponding $P[i, j]$ is taken equal to 0 otherwise it is equal to one. Hereafter, the pair of words with $P[i, j]$ equals to 1 is called as valid bigram. Note that, the number of valid bigrams is M lower

Table 1. The construction of a NxN confusion matrix, for the sentence $a = [w[0], w[1], ...w[n-1], w[n]]$

WORD	**w[0]**	**w[1]**	**…….**	**w[n]**
w[0]	P[0,0]	P[1,0]	…….	P[n,0]
w[1]	P[0,1]	P[1,1]	…….	P[n,1]
.	.	.		.
.	.	.		.
.	.	.		.
w[n]	P[0,n]	P[1,n]	…….	P[n,n]

than the size of the confusion matrix which is N^2, since all possible pairs of words are not valid according to the language model. In order to generate permuted sentences using the valid bigrams all the possible words' sequence must be found. This is the search problem and its solution is the domain of this filtering process.

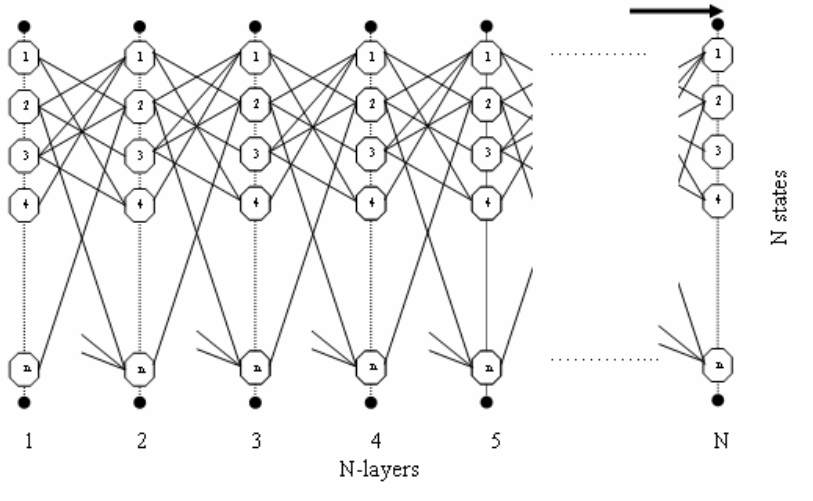

Fig. 3. Illustration of the lattice with N-layers and N states

As with all the search problems there are many approaches. In this paper a left to right approach is used. To understand how it works the permutation filtering process, imagine a network of N layers with N states. The factor N concerns the number of sentence's words. Each layer corresponds to a position in the sentence. Each state is a possible word. All the states on layer 1 are then connected to all possible states on the second layer and so on according to the language model. The connection between two states (i, j) of neighboring layers $(N-1, N)$ exists when the bigram $(w[i]w[j])$

is valid. This network effectively visualizes the algorithm to obtain the permutations. Starting from any state in layer 1 and moving forward through all the available connections to the *N*-th layer of the network, all the possible permutations can be obtained. No state should be "visited" twice in this movement.

5 Searching Valid Trigrams

The prime function of this approach is to decompose any input sentence into a set of trigrams. To do so, a block of words is selected. In order to extract the trigrams of the input sentence, the size of each block is typically set to 3 words, and blocks are normally overlapped by two words. Therefore, an input sentence of length N, includes N-2 trigrams.

The second step of this method involves the search for valid trigrams for each sentence. A probability is assigned to a valid trigram, which is derived by the frequency of its occurrences in the corpus.

In the third step of this method the number of valid trigrams per each permuted sentence is calculated. Considering that the sentence with no word-order errors has the maximum number of valid trigrams, it is expected that any other permuted sentence will have less valid trigrams. Although some of the sentence's trigrams may be typically correct, it is possible not to be included into the list of LM's trigrams. The plethora of LM's trigrams relies on the quality of corpus. The lack of these valid trigrams does not affect the performance of the method since the corresponding trigrams of the permuted sentence will not be included into LM as well. The criterion for ranking all the permuted sentences is the number of valid trigrams. The system provides as an output, a sentence with the maximum number of valid trigrams. In case where two or more sentences have the same number of valid trigrams a new distance metric should be defined. This distance metric is based on the total log probability of the sentence's trigrams. The total log probability is computed by adding the log probability of each valid trigram, whereas the probability of non valid trigrams is assigned to -100000. Therefore the sentence with the maximum total log probability is the system's response.

6 Experimentation

6.1 Experimental Scheme

The experimentation involves a test set of 500 sentences, with 4518 words. Test sentences have been selected randomly from WSJ (Wall Street Journal) corpus. They have variable length with minimum 7 words and maximum 12 words. The 90% of the test words belong to the BNC vocabulary (training data). For experimental purposes our test set consists of sentences with no word order errors and the system's response incorporates 10-best sentences. The goal of this experimentation is to show that the input sentence is included into the 10-best sentences. Note that the test sentences are not included into the training set of the statistical language model that is used as tool for the proposed method.

6.2 Experimental Results

6.2.1 WSJ Test Cases

Figure 4 shows the repairing results using the test sentences. This figure depicts the capability of the system to give as output the correct sentences in the 10-best list. The x-axis corresponds to the place of the correct sentence into this list. The last position (11) indicates that the correct sentence is out of this list.

The findings from the experimentation show that 455 sentences (91% in total) have been repaired using the proposed method (True Corrections). On the other hand, the result for 45 sentences (9% in total) was false (False Corrections). In case of "False Corrections" the system's response does not include the correct sentence into the N-best. The incorrect output of the system can be explained considering that some words are not included into the BNC vocabulary, hence some of the sentences' trigrams are considered as invalid.

It is obvious that the system's performance for detecting and repairing method of ill-formed sentences with word order errors depends mainly on the quality of the corpus. The high success rate of the system is achieved using the grammatically and syntactically correct sentences of BNC.

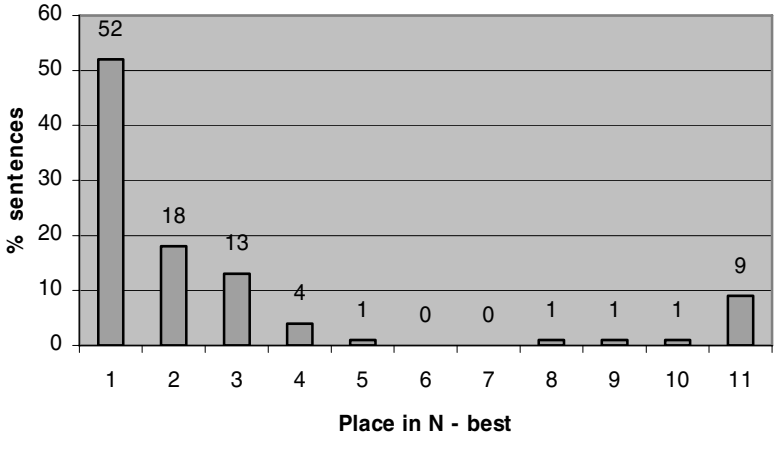

Fig. 4. The percentage of test sentences in different places into the N-best list (N=10)

6.2.2 Reducing Search Space

The figure below depicts the differences in the number of permutations for sentences with length from 7 to 12 words. The point is that the number of permutations that are extracted with the filtering process is significantly lower than the corresponding value without filtering. For sentences with length up to 6 words, the number of permutations is slightly lower when the filtering process is used, while for sentences with length greater than 7 words the filtering process provides a drastical reduction of permutations. It is obvious that the performance of filtering process depends mainly on the

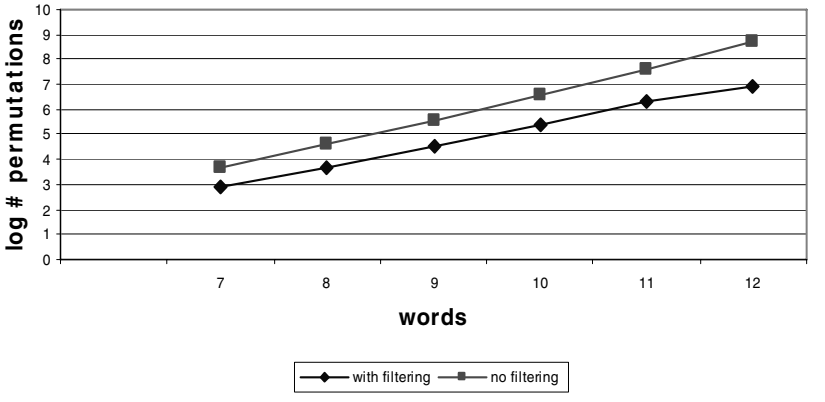

Fig. 5. The logarithmic number of permutations with and without filtering for TOEFL's sentences with 7 up to 12 words

number of valid bigrams. This implies that the language model's reliability affects the outcome of the system and especially of the filtering process.

7 Conclusions

The findings show that most of the sentences can be repaired by this method independently from the sentence's length and the type of word order errors. The major advantage of this technique concerns the application of novel fast algorithm in reducing permutations. The results show that the gain factor for permutations in case of sentences with 12 words is 35dB. With no filtering the number of permutations is 479001600 while with the confusion matrix this quantity decreases drastically to 8790541. The proposed method is effective in repairing erroneous sentences. Therefore the method can be adopted by a grammar checker as a word order repairing tool. The necessity of the grammar checkers in educational purposes and e-learning is more than evident.

By the permutation's filtering process, the system takes advantage of better performance, rapid response and smaller computational space. A comparative advantage of this method is that avoids the laborious and costly process of collecting word order errors for creating error patterns. One of the key questions for further research is whether the use of language model can correct other grammatical errors such as subject- verb disagreement, and if it is possible a further reduction in permutations using probabilities thresholds.

Acknowledgments

The authors would like to thank Mr Kostantinos Mamouras for his programming skills and the insightful comments.

References

1. Atwell, E.S., How to detect grammatical errors in a text without parsing it. In Proceedings of the 3rd EACL, (1987) 38–45
2. Bigert, J., Knutsson. O., Robust error detection: A hybrid approach combining unsupervised error detection and linguistic knowledge. In Proceedings of Robust Methods in Analysis of Natural language Data, (ROMAND 2002), (2002) 10–19
3. Chodorow M., Leacock C. An unsupervised method for detecting grammatical errors. In Proceedings of NAACL'00, (2000) 140–147
4. Feyton, C. M., Teaching ESL/EFL with the internet. Merill Prentice- Hall, (2002)
5. Folse, K.S., Intermediate TOEFL Test Practices (rev. ed.). Ann Arbor, MI: The University of Michigan Press., (1997)
6. Good, I.J., The population frequencies of species and the estimation of population parameters. Biometrika, 40(3 and 4): (1953) 237-264,
7. Golding, A. A., Bayesian hybrid for context-sensitive spelling correction. Proceedings of the 3rd Workshop on Very Large Corpora, (1995) 39-53
8. Hawkins, J. A., A Performance Theory of Order and Constituency. Cambridge, Cambridge University Press, (1994)
9. Heift, T., Intelligent Language Tutoring Systems for Grammar Practice. Zeitschrift für Interkulturellen Fremdsprachenunterricht (Online), 6 (2), (2001) 15 pp
10. Katz S.M., Estimation of probabilities from sparse data for the language model component of a speech recogniser. IEEE Transactions on Acoustics, Speech and Signal Processing, 35(3): (1987) 400-401,
11. Sjöbergh, J., Chunking: an unsupervised method to find errors in text, Proceedings of the 15th Nordic Conference of Computational Linguistics, NODALIDA (2005)
12. Young, S.J., Large Vocabulary Continuous Speech Recognition, IEEE Signal Processing Magazine 13, (5), (1996) 45-57,