A CORPUS BASED TECHNIQUE FOR REPAIRING ILL-FORMED SENTENCES WITH WORD ORDER ERRORS USING CO-OCCURRENCES OF N-GRAMS

Theologos Athanaselis, Konstantinos Mamouras, Stelios Bakamidis, Ioannis Dologlou

Institute for Language and Speech Processing (ILSP) / R.C “ATHENA”
Artemidos 6 & Epidavrou, MAROUSSI, ATHENS, GR-151 25, GREECE

Corresponding Author

Theologos Athanaselis
Department of Voice and Sound Technology
Institute for Language and Speech Processing (ILSP) / R.C “ATHENA”
Artemidos 6 & Epidavrou
GR – 151 25 MAROUSSI
Athens
Greece
Tel:+30(210)6875416
Fax: +30(210)6854270
E-mail: tathana@ilsp.gr
A CORPUS BASED TECHNIQUE FOR REPAIRING ILL-FORMED
SENTENCES WITH WORD ORDER ERRORS USING CO-
OCCURENCES OF N-GRAMS

Abstract

There are several reasons to expect that recognising word order errors in a text will be a difficult
problem, and recognition rates reported in the literature are in fact low. Although grammatical rules
constructed by computational linguists improve the performance of a grammar checker in word order
diagnosis, the repairing task is still very difficult. This paper describes a method to repair any sentence
with wrong word order using a statistical language model (LM). A good indicator of whether a person
really knows a language is the ability to use the appropriate words in a sentence in correct word order.
The “scrambled” words in a sentence produce a meaningless sentence. Most languages have a fairly
fixed word order. This paper introduces a method, which is language independent, for repairing word
order errors in sentences using the probabilities of most typical trigrams and bigrams extracted from a
large text corpus such as the British National Corpus (BNC).

Keywords

Word order errors; statistical language model; permutations filtering; British National Corpus.
1 Introduction

Syntax is the term used to describe relationships of constituents to one another in sentences of a language. Some languages have relatively restrictive word orders, often relying on the order of constituents to convey important grammatical information. Others, often those that convey grammatical information through inflection, allow more flexibility which can be used to encode pragmatic information such as topicalisation or focus. Most languages however have some preferred word order which is used most frequently. For example, in English the normal way of ordering elements is subject, verb, object (e.g. Boy meets girl) (Hawkins, 1994). Subjects and objects are composed of noun phrases, and within each noun phrase there are elements such as articles, adjectives, and relative clauses associated with the nouns that head the phrase (e.g. the tall woman who is wearing a hat). Native speakers of a language seem to have a sense about the order of constituents of a phrase, and such knowledge appears to be outside of what one learns in school (Schneider and McCoy, 1998).

Errors in grammar, spelling and punctuation obscure the meaning of what is being said; they decrease clarity and interfere with communication. When you write something you want it to be understood (unless your intention is really obfuscation), and anything which makes it harder for your reader to understand what you are saying works against your purpose in writing. For non-native speakers and writers, word order errors are more frequent in English as a Second Language (ESL) writing tasks (Faigley, 2003). These errors come from the student if he is translating (thinking in his/her native language and trying to translate it into English).

On the other hand, there are languages that appear to have word order freedom like Modern Greek. It is a highly flexible language where word order is concerned. The functions of the nouns are very clear due to the morphological forms. In English, the position of each noun tells the listener what role the noun plays. The strict rule of SVO (subject-verb-object) does not apply to Greek. Depending on the word order, an emphasis may arise for an idea in the sentence, and of course, the positions of adverbs can change the meaning of the sentence. In other words, the word order could affect the writing style which in turn depends on the writer’s intentions.

Writers can create sentence structures that convey their intended effect, and the choice of sentence types has a great deal of influence on the reader's perceptions; placing key words and phrases in positions of emphasis in a sentence can highlight some parts of a sentence and de-emphasize others. Examples: "Although Neal Weasel was often moody and brusque in my classes, he is surely a genius"; "Although Neal Weasel is surely a genius, he was often moody and brusque in my classes."
Research on detecting erroneous sentences can be mainly classified into three categories. The first category makes use of hand-crafted rules. Some of these methods are based on template rules (Heidorn, 2000) and others on mal-rules in context-free grammars (Michaud et al., 2000; Bender et al., 2004; Naber 2003). These methods have been shown to be effective in detecting certain kinds of grammatical errors, but it is expensive to write non-conflicting rules in order to cover the wide range of grammatical errors.

The second category focuses on parsing ill-formed sentences. To cope with grammatical errors, new mechanisms (Fouvry, 2003; Vogel and Cooper, 1995; Lee and Seneff, 2006) have been incorporated into parsers which, otherwise, are intended for analyzing well-formed sentences. Atwell (1987b) uses the probabilities in a statistical part-of-speech tagger, detecting errors as low probability part of speech sequences. Another example for error-production rules in context-free-grammars is ICICLE, a broad coverage system designed for teaching English to American Sign Language signers. In ARBORETUM, such rules are used in conjunction with an aligned generation strategy, so that “the corrected form should match the input in all ways except those affected by the correction”. A disadvantage of the above parsing based approaches is that as more and more types of errors need to be handled, the grammars become increasingly complicated, and the number of ambiguous parses grows exponentially.

The third category uses statistical techniques to detect erroneous sentences. Instead of asking experts to write hand-crafted rules, statistical approaches (Chodrow and Leacock, 2000; Izumi et al., 2003; Brockett et al., 2006; Costa-Jussa and Fonollosa, 2009; Sun et al., 2007) build statistical models to indentify sentences containing errors. Chodorow and Leacock (2000) suggested an unsupervised method for detecting grammatical errors by inferring negative evidence from edited textual corpora. Izumi (2003) aims to detect omission-type and replacement-type errors to learn rules to detect errors for speech recognition outputs. The phrasal Statistical Machine Translation (SMT) technique is employed to indentify and correct writing errors (Brockett et al., 2006). This method must collect a large number of parallel corpora (pairs of erroneous sentences and their corrections). Costa-Jussa and Fonollosa, introduced an N-gram-based reordering approach with training data, word-classes instead of words themselves. Sun et al., (2007) proposed a method to detect erroneous sentences by integrating pattern discovery with supervised learning models. However existing statistical approaches focus on predefined errors that usually require that errors to be specified and tagged in the training sentences. This is time consuming and labor intensive.

There are also other studies on detecting grammar errors at sentence level. More (2006) introduced an English grammar checker for non-native English speakers. The main characteristic of this checker is the use of an Internet search engine. As the number of web pages written in English is immense, the system hypothesises that a piece of text not found on the Web is probably badly written. The checker warns the user about the ill formed text segment and an Internet engine searches for contexts that can
help the user decide whether to correct the segment or not. 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.

Automatic grammar checking is traditionally done using manually written rules constructed by computational linguists. Methods for detecting grammatical errors without manually constructed rules have been presented before. Golding (1995) showed how methods used for decision lists and Bayesian classifiers could be adapted to detect spelling errors (i.e. misspelling “there” and “their”). 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. Bigert and Knutsson (2002) showed how a new text can be compared to known correct text and deviations from the norm flagged as suspected errors. Sjöbergh and Knutsson (2005) introduced a method of grammar error recognition by adding errors to several (mostly error free) unannotated texts and applying a machine learning algorithm.

In the framework of grammar error detection and repairing, our method is based on the statistical language model. Such a model describes probabilistically the constraints on word order found in language: typical word sequences are assigned high probabilities, while atypical ones are assigned low probabilities. For this reason the statistical language model can be used in reordering words and creating meaningful sentences. The use of statistical language models in natural language generation is very significant for many applications e.g. man-machine interface, automatic translation, text generation etc. In the corpus-based approach (Church and Mercer, 1993), a language model is provided to measure the possible candidates. Langkilde and Knight (1998) introduced Nitrogen, a system that uses bigram and unigram statistics as a way of choosing among different lexical and syntactic options. Bangalore and Knight (2000) presented a model of syntactic choice which, like Langkilde and Knight (1998) relies on a linear language model, but unlike their approach, also used a tree based representation of syntactic structure, a tree model, and an independently hand-crafted grammar.

In contrast to existing statistical methods, the proposed method is applicable to any language (language models can be computed in any language) and it is not restricted to a specific set of words. For that reason use of parser and/or tagger is not necessary. Also, it does not require manual collection of written rules since they are outlined by the statistical language model. A comparative advantage of this method is that it avoids the laborious and costly process of collecting word order errors for creating error patterns. Finally, the performance of the method does not depend on the word order patterns which vary from language to language, and for that reason it can be applied to any other language with less fixed word order.
The remainder of this paper is organized as follows: section 2 describes the method for finding the correct word order in a sentence. Section 2 includes the architecture of the entire system and a description of the language model. Section 3 shows how permutations are filtered by the proposed method and describes the method that is used for searching for valid trigrams in a sentence. An experimental setup using TOEFL, WSJ, and non native English corpus test data is given in section 5. The results of using different experimental schemes, the concluding remarks and future plans are discussed in section 6.

2 Finding the correct word order in a sentence

This paper presents a new method for repairing sentences with candidate word order errors that is based on the conjunction of a new association technique with a statistical language model. The best way to reconstruct a sentence with word order errors is to reorder the words’ sequence. However, the question is how it can be achieved without knowing the Part of Speech (POS) tag of each word. Many techniques have been developed in the past to cope with this problem using a grammar parser and rules. This paper deals with a new technique for handling word order errors using all the possible words permutations of the sentence. The process of repairing sentences with word order errors incorporates the followings tools:

- a fast algorithm for filtering the sentence’s permutations
- and a statistical Language Model (LM) based on N-grams.

The concept is based on the following two axioms: the first axiom concerns the assumption that taking into account all the permutations of a sentence with word order errors, it is absolutely certain that the correct sentence will be included in the set of the permutations. The second axiom relies on the assumption that the number of valid trigrams (sentence’s trigrams that are included in the language model) increases as the number of word order errors declines. Therefore, the system provides as output a list of N-best sentences according to the number of valid trigrams (see Figure 1). The following steps summarise the main algorithmic steps of the proposed method for repairing sentences’ word order errors.

1. The basic units for the reconstruction method are the sentence’s words. 
   \[ W_1W_2...W_{n-2}W_{n-1}W_n \] (where \( W_i \) is the i-th word).

2. Construction of an association matrix for extracting valid bigrams
3. Construction of a network with valid bigrams in order to form possible permuted sentences of length N
4. Decomposition of each permuted sentence into a set of trigrams
5. Evaluation of the sentences according to the number of valid trigrams
6. If more than one sentence has the same number of valid trigrams the sum of trigrams’ log probability is taken into account.

Figure 1 The architecture of the proposed system
2.1 Language model generation

The language model (LM) that is used subsequently is the standard statistical N-grams model (Jelinek, 1976). The N-grams model provides an estimate of $P(W)$, the probability of observed word sequence $W$. Assuming that the probability of a given word-token in an utterance depends on a finite number of preceding words-tokens, the probability of an N-word string can be written as:

$$P(W) = \prod_{i=1}^{n} P(w_i \mid w_{i-1}, w_{i-2}, \ldots, w_{i-(n-1)})$$  \hspace{1cm} (1)

N-grams simultaneously encode syntax, semantics and pragmatics and they concentrate on local dependencies. This makes them very effective for languages where word order is important and the strongest contextual effects tend to come from near neighbours. A statistical language model probabilistically describes the constraints on word order found in language: typical word sequences are assigned high probabilities, while atypical ones are assigned low probabilities. N-grams have also been chosen, because the N-gram probability distributions can be computed directly from text data requiring no explicit linguistic rules (e.g. formal grammars). Text resources are available in any language using electronic databases. In cases where no resources are available on the Internet, someone can download html pages and extract raw texts. N-gram computation depends on the amount and the quality of the training data. The greater N, the better language modeling. Previous work shows that we can extract bigrams with reliable probabilities, but for better language modeling, we need greater N-grams. For that reason we used trigrams with less reliable probabilities.

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 large number of putative “zero probability N-grams” that should have some non zero probability. Part of this problem is inherent to N-grams; since they can not use long distance context, they always tend to underestimate the probability of strings that are not adjacent in their training corpus. Although there is a non-trivial limitation for the use of N-gram models, techniques like back-off smoothing are used. These techniques are used to assign a non zero probability to these zero probability N-grams. In this work, the language model has been trained using the British National Corpus (BNC) with Good-Turing discounting (Good, 1953) and Katz’s back off technique for smoothing (Katz, 1987). BNC contains about 6.25M sentences and 100 million words. The language model database contains bigrams with log probabilities from -7.26475 up to -0.00261 and trigrams with upper bound -5.844125 and lower bound -0.000154. These log probabilities were obtained after getting rid of the large number of zero probability N-grams using the backoff implemented techniques.
Table 1 shows the number of unique unigrams, bigrams, and trigrams of language model for the English language. As an example, a bigram consists of two words-tokens i.e. “her baby” and a trigram consists of three words-tokens i.e. “the major impact”.

<table>
<thead>
<tr>
<th>LM elements</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigrams</td>
<td>126062</td>
</tr>
<tr>
<td>Bigrams</td>
<td>816674</td>
</tr>
<tr>
<td>Trigrams</td>
<td>803315</td>
</tr>
</tbody>
</table>

Table 1 The number of different elements of LM

Figures 2 and 3 depict the number of bigrams and trigrams of the language model with respect to their logarithmic probabilities.

**Figure 2** The distribution of bigrams with respect to the log probabilities. The A symbol corresponds to 75302 single bigrams with log probability -5.48.
3 A more efficient solution

Considering that an ungrammatical sentence includes the correct words but in the wrong order, it is plausible that all the permutations of the sentence words are generated, one of them will be the correct sentence. The question is how feasible it is to deal with all the permutations for sentences with a large number of words. The difficulty with this approach is to handle all these permutations. Note that given a sentence with length $N$ words, the number of permutations is $N!$. Therefore, a filtering process of all possible permutations is necessary.

3.1 Permutation filtering

The filtering involves the construction of an $N \times N$ association matrix to extract possible reordered sentences.
Flowchart of the permutations filtering technique for an input sentence with $N$ words. An association matrix using the words of the input sentence is constructed to detect valid bigrams using the language model. Then, the permuted words are formed using a lattice with $N$-layers and $N$ states.

Given a sentence $a = [w[1], w[2], ... , w[N-1], w[N]]$ with $N$ words, an association matrix $A \in \mathbb{R}^{N \times N}$ can be constructed,

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>w[1]</td>
<td>■</td>
<td>P[1,2]</td>
<td>P[1,3]</td>
<td></td>
<td>P[1,N-1]</td>
<td>P[1,N]</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w[N-1]</td>
<td>P[N-1,1]</td>
<td>P[N-1,2]</td>
<td>P[N-1,3]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Construction of the $N \times N$ association matrix, for a given sentence $a = [w[1], w[2], ... , w[N-1], w[N]]$

The size of the matrix depends on the length of the sentence. The objective of this association 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 words $(w[i]w[j])$ cannot be found in the list of language model bigrams then the corresponding $P[i, j]$ is equal to zero otherwise it is equal to one. Although some valid bigrams do not exist in the training corpus, using techniques like Katz’s backing off or Good-Turing discounting, they are assigned non zero probabilities. Hereafter, a pair of words with probability $P[i, j]$ greater than zero is called a valid bigram. Note, that the number of valid bigrams $M$ is smaller than the size of the association matrix which is $N^2$ since all possible pairs of words are not valid according to the language model.

To generate permuted sentences using the valid bigrams, all the possible word sequences must be found. This is a search problem and its solution is the domain of this filtering process.
As with all search problems, there are many approaches. In this work, a forward approach is used. To understand how the permutation filtering process works, imagine a network of $N$ layers with $N$ states. The factor $N$ is the number of words in the sentence. 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. The network below visualizes the algorithm used 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.

To reduce the number of permutations further, it is necessary to use two different constraints. The first one considers the fact that several words cannot be in the first and the last position of the BNC sentence. For this reason, it is important to estimate the frequency of bigrams such as $P(<s>,W_i)$ and $P(W_n,<s>)$. Where $<s>$ and $</s>$ are tokens corresponding to the start and the end of each BNC sentence respectively. Additionally, a reordering constraint ($\lambda$) can be applied for avoiding long distance reordering in a sentence.

The advantage of the filtering technique is that it only relies on the use of valid bigrams and not on the use of every bigram. The number of word permutations in a sentence depends on the number of valid bigrams. Considering that for a sentence with 7 words the number of all possible bigrams is 49, it is found that using the language model, the number of valid bigrams is significantly lower. Combining
only the valid bigrams, the number of permutations for a sentence with $N$ words declines drastically. The next section describes the use of valid trigrams in ranking permuted sentences.

### 3.2 Searching for valid trigrams
The prime function of this approach is to decompose an 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.

![Diagram](image)

**Figure 6** The architecture of the subsystem for repairing sentences with word-order errors. It is based on the algorithm for searching valid trigrams according to the Language Model (LM)

![Diagram](image)

**Figure 7** Decomposition method of an input sentence using a sliding window
The second step involves searching for valid trigrams for each sentence. A probability is assigned to a valid trigram, which is derived from the frequency of its occurrences in the corpus (Figure 8).

**Figure 8** The two phase algorithm (PHASE 1, PHASE 2) for searching a set of trigrams in LM. In phase 1, the input sentence is decomposed into a set of trigrams. In phase 2 the set of trigrams are searched in the list of LM’s trigrams. The result of this process is a list of valid trigrams $W_{N-2}W_{N-1}W_N$ and their probabilities $p_i$.

In the third step of this method, the number of valid trigrams per each permuted sentence is calculated. Although some of the sentence’s trigrams may be typically correct, it is possible they are not included into the list of LM’s trigrams due to the data sparsity problem. For this reason, discounting and smoothing techniques are used.

The criterion for ranking all the permuted sentences is the number of valid trigrams. The system provides an output of the top ten sentences with the maximum number of valid trigrams. In cases 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 logarithmic probability of the trigrams. The total log probability is computed by adding the log probability of each trigram, whereas the log probability of non-valid trigrams is assigned a negative number.

The aim of next example is to show that the sentence, “I have also campaigned for the government to give AIDS greater recognition” (with no word order errors), has the best score compared with the rest of the reordered sentences (result from filtering process). The sentences are top-ranked according to the number of valid trigrams.
Table 3 depicts the valid bigrams that have been extracted by the construction of the association matrix for the sentence “I have also campaigned for the government to give AIDS greater recognition”. These bigrams will be used for the generation of the sentence’s possible permutations. Considering that the number of words is 12, the number of all possible bigrams is 132. In this example, the number of valid bigrams is reduced to 61 by employing the filtering process.

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>have</th>
<th>also</th>
<th>campaigned</th>
<th>for</th>
<th>the</th>
<th>government</th>
<th>to</th>
<th>give</th>
<th>AIDS</th>
<th>greater</th>
<th>recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>have</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
<td>■</td>
<td>■</td>
<td></td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
<td></td>
</tr>
<tr>
<td>also</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td>■</td>
<td></td>
<td>■</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
</tr>
<tr>
<td>campaigned</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■</td>
<td>■</td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>for</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
<td>■</td>
<td>■</td>
<td></td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
<td></td>
</tr>
<tr>
<td>the</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>government</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■</td>
<td>■</td>
<td></td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
<td>■</td>
<td>■</td>
<td></td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
<td></td>
</tr>
<tr>
<td>give</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
</tr>
<tr>
<td>AIDS</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>greater</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
<td>■</td>
<td>■</td>
<td></td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recognition</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
<td></td>
<td>■</td>
<td></td>
<td>■</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3** The association matrix for the input sentence “I have also campaigned for the government to give AIDS greater recognition”. The symbol (■) refers to valid pair of words (valid bigrams). According to the language model there are 61 valid bigrams.

The number of reordered sentences as a result of by the permutations filtering technique is 245,519. With no filtering, the corresponding value is 479,001,600. The reduction of the number of permutations makes the selection of the sentence with the maximum number of trigrams feasible. It can be seen in Table 4 that the input sentence has a greater number of trigrams compared with the reordered sentences.
I have also campaigned for the government to give AIDS greater recognition

<table>
<thead>
<tr>
<th>Sentence</th>
<th># of trigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  I have also campaigned for the government to give AIDS greater recognition</td>
<td>10</td>
</tr>
<tr>
<td>2  I also have campaigned for the government to give AIDS greater recognition</td>
<td>9</td>
</tr>
<tr>
<td>3  I have also campaigned to give AIDS greater recognition for the government</td>
<td>7</td>
</tr>
<tr>
<td>4  I have also campaigned for the government to give greater recognition AIDS</td>
<td>8</td>
</tr>
<tr>
<td>5  for the government to give AIDS greater recognition I have also campaigned</td>
<td>8</td>
</tr>
<tr>
<td>6  for the government I have also campaigned to give AIDS greater recognition</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4 The top ranked sentences in descending order according to the total number of valid trigrams. The highlighted sentence is the input sentence and the other sentences have been extracted from the permutations’ filtering.

For sentences with a length of up to 8 words, the number of permutations is slightly lower when the filtering process is used. While for sentences with a length greater than 8 words, the filtering process reduces the number of permutations. Note that by increasing the size of the training corpus, better trigram statistics are obtained which improves the ranking of sentences. At the same time, the increase in the number of valid bigrams is considerably lower than that of valid trigrams, which guarantees that no further computational complexity is added.

Figure 9 The number of permutations with and without filtering in logarithmic scale for sentences with length from 5 to 12. The symbol (*) denotes the log number of the sentence’s permutations without filtering while the symbol (o) presents the log number of the permutations extracted from the filtering method.
4 Evaluation

This section describes the results from 3 different experimental tasks using the reordering constraint ($\lambda$). This constraint forces the words’ position within certain limits. In the case of the TOEFL test data, the input sentence’s words cannot be reordered in distance ($\lambda$) greater than 3 words. In the case of the WSJ test data, different values of $\lambda$ are used. In the case of the non-native English speakers corpus, no such constraint is applied.

4.1 The Test of English as a Foreign Language (TOEFL) experiment

The experiment involved a test set of 310 sentences with length between 4 and 12 words with a total of 2347 words. These sentences have been selected randomly from the section “Structure” of online TOEFL past exams (Folse, 1997; Feyton, 2002). The TOEFL test refers to the Test of English as a Foreign Language. The TOEFL program is designed to measure the ability of non-native speakers to read, write and understand English as used at colleges and universities in North America. The Structure section focuses on recognizing vocabulary, grammar and proper usage of standard written English. There are two types of questions in the Structure section of the TOEFL test. One question type presents candidates with a sentence containing a blank line. Test-takers must choose a word or phrase that appropriately fills in the blank. This question type is a part of Word order practice. The other question type consists of complete sentences with four separate underlined words. Candidates must choose which of the four underlined answer choices contains an error in grammar or usage. For experimental purposes our test set consists of sentences from TOEFL’s word order practice. These sentences were selected from the list of the answer choices but are not the correct ones. The goal of the experimental scheme is to confirm that the outcome of the method (sentence with best score) is the TOEFL’s correct answer. For experimental purposes, the reordering constraint is set to $\lambda=3$. The reason for selecting $\lambda=3$ is that most of the TOEFL answers consist of 4 words. This constraint restricts the position of each input word to distance less than 3. Therefore the permutations that do not comply with this restriction are omitted. The next figure shows an example of TOEFL grammar test.
4.1.1 Errors profile

A report on gathered data in this study is presented in this section. We discuss a categorization of sentences found in the test set according to the length and the type of each sentence; and we also describe the distribution of errors in the entire test data and in different types of sentences (Park et al., 1997). Figure 11 shows the percentage of different types of sentences in the test set. The test set contains 180 positive sentences, 96 questions, 31 negative sentences and 3 imperative sentences.

Figure 10 A simple TOEFL sentence practice

<table>
<thead>
<tr>
<th>EXAMPLE 1</th>
</tr>
</thead>
</table>

------- explores the nature of guilt and responsibility and builds to a remarkable conclusion.

A. The written beautifully novel
B. The beautifully written novel
C. The novel beautifully written
D. The written novel beautifully

---

Figure 11 The percentage of different type sentences in the corpus

The histogram in Figure 12 shows the frequency of the corpus’ sentences as a function of their length. Most of the corpus contains sentences with lengths between 4 and 12 words.

In the TOEFL test set of 2347 words, 315 instances of word order errors were found. The test sentences display 5 different word order errors. The word order errors concern the transposition of verbs, nouns, adjectives, adverbs, and pronouns, thus violating the sentences’ word order constraints (Izumi et al., 2003). The most common errors were verb transpositions with 35.0% and adverb transpositions with 30.5%. The errors with adjective transpositions were lower (19.9%). Noun transpositions were less frequent at 11.4%, and errors with pronouns were least frequent at 3.4%.

Figure 13 shows the distribution of word order errors for each type of sentence. As shown in Figure 13, the most frequent errors in the whole test set were verb transpositions with 35.0%; this holds for all different types of sentences except for questions where the most frequent word order errors were the adverb transpositions. Regarding imperative sentences, it can be observed that there were no pronoun, noun and adverb transpositions.

### 4.1.2 Results using TOEFL

The evaluation of this method was conducted by comparing the output of the system with the correct answer choice that is indicated by TOEFL. The findings from the experimentation
(Figure 15) show that 296 sentences (95% in total) were repaired using the proposed method (True Corrections). On the other hand, the result for 5% of the input sentences was false (False Corrections). In the case of “False Corrections” the system’s response is different from the correct sentence. In cases where $\lambda$ equals N-1, 84% of the test sentences were detected and fixed using the proposed method, while for the rest of the sentences (16%), our system did not manage to correct the user’s word order errors.

4.2  The Wall Street Journal Experimental scheme

The experiment involved a test set of 1,189,085 sentences (9,576,805 test words). Test sentences were randomly selected from Wall Street Journal (WSJ) corpus. They have variable length with a minimum of 7 words and a maximum of 12 words, and they are free from word order errors (Table 5). It can’t be answered whether 12 words are enough or not, but in the case of real life texts (test data) the idea is to find words like comma, and, etc to split long sentences into smaller parts. The test words belong to the BNC vocabulary (training data). For experimental purposes the system’s response incorporated 10-best sentences. The goal of this experiment was to find the ranking of the input sentences in the 10-best list. Ideally, it is expected that the correct sentence is in the first position.

<table>
<thead>
<tr>
<th>words per sentence</th>
<th># test set sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>269323</td>
</tr>
<tr>
<td>8</td>
<td>234717</td>
</tr>
<tr>
<td>9</td>
<td>205216</td>
</tr>
<tr>
<td>10</td>
<td>180356</td>
</tr>
<tr>
<td>11</td>
<td>158693</td>
</tr>
<tr>
<td>12</td>
<td>140780</td>
</tr>
</tbody>
</table>

Table 5  Distribution of test data with respect to the sentences’ length

4.2.1  Experimental results

4.2.1.1 WSJ test cases

To evaluate our method according to WSJ test data, three experiments were undertaken with three different reordering constraints ($\lambda=3$, 4, N-2) and one with no reordering constraint. For $\lambda= N-1$
(no reordering constraint) the system can detect and fix 73% of the input sentences. Meanwhile, for 27% of the input sentences a false alarm was returned. The system’s performance depends on the $\lambda$ values. As $\lambda$ increases, the percentage of test sentences in the first position drops (Figure 14). The question is how many permutations are omitted using such a reordering constraint. The percentage of permutations that were ranked higher than the input sentence and omitted using the factor $\lambda = 3$ was 0.01% of the overall permutations for $\lambda = N-1$.

![Figure 14](image)

**Figure 14** The ranking of test sentences in the N-best list (N=10) for different window’s size

Figure 15 shows the results using different test sentences without reordering constraint. This figure depicts the capability of the system to rank the correct sentence in the 10-best list, with or without the use of the $<s>$ and $</s>$ tokens. The x-axis corresponds to the position of the correct sentence in this list. The last position (11) indicates that the correct sentence is out of the list.
4.2.1.2 Permutations filtering results

Table 6 presents the mean value of permutations for sentences with lengths from 7 to 12 words, when no reordering constraint is used. The point is that the number of permutations that were extracted with the filtering process was significantly lower than without filtering. For sentences with lengths up to 6 words, the number of permutations is slightly lower when the filtering process is used, while for sentences with lengths greater than 7 words, the filtering process provided a drastic reduction in the number of permutations. It is obvious that the performance of the filtering process depends mainly on the number of valid bigrams. This implies that the language model’s reliability affects the outcome of the system and especially of the filtering process. Figure 16 depicts the mean value of permutations for test sentences in log scale.

Figure 15 The percentage of test sentences in different positions in the N-best list (N=10) with or without the use of <s> and </s> tokens
Table 6 The mean value of permutations for test sentences

<table>
<thead>
<tr>
<th>Words</th>
<th>with filtering</th>
<th>no filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1252</td>
<td>5040</td>
</tr>
<tr>
<td>8</td>
<td>10560</td>
<td>40320</td>
</tr>
<tr>
<td>9</td>
<td>82012</td>
<td>362880</td>
</tr>
<tr>
<td>10</td>
<td>652602</td>
<td>3628800</td>
</tr>
<tr>
<td>11</td>
<td>5963401</td>
<td>39916800</td>
</tr>
<tr>
<td>12</td>
<td>11378400</td>
<td>479001600</td>
</tr>
</tbody>
</table>

Figure 16 The mean value of permutations in logarithmic scale

Both Figures 17 and 18 give the distribution of test sentences according to the number of valid bigrams in the case of 9 word sentences. Figure 17 depicts the number of test sentences with respect to the valid-bigrams. Experimentally, it was shown that the distribution of valid bigrams per test sentence is similar for different sentence lengths. As we can see, the peak of the distribution is at 56 which implies that most of the sentences can be treated efficiently with our fast algorithm. Note that for a 9 word sentence, the association matrix contains 72 valid-bigrams when all word pairs belong to the language model database.
Another important issue is determining the dependency of permutations and valid bigrams. Figure 18 shows the mean value of permutations as a function of valid bigrams. The mean value of permutations corresponds to the permutations’ average for (9 words)-test sentences with a specific number of valid bigrams. Note that for sentences with 72 valid bigrams, the number of permutations maximizes. In this case the fast algorithm does not work, and all the possible permutations are considered.

**Figure 17** Number of test sentences with respect to the valid-bigrams, 9 word sentence

**Figure 18** The mean value of permutations for different number of valid bigrams
4.3 Non native English corpus

The experiment involves a test set of 97 sentences translated from Greek to English. The sentences were randomly selected from the Web. They have variable length from 10 to 12 words. A total of 63 persons (54% male and 46% female) participated in the experiment. The ages of the participants ranged from 24 to 56 years old. 90.19% of the participants held a university degree. Each participant was asked to translate at least 20 test sentences to English. All the participants had Greek as their mother tongue and English as a second language. The purpose of this experiment was to determine how the system performs on real data. For that reason, a native English-speaking person was used as a rater. The aim was to check whether our system could fix real life word order errors. The rater selected only the sentences that include word order errors. Each one of these sentences was fed to our system to investigate whether it could be fixed or not. No reordering constraint was applied to each input sentence. The rater compared the users’ translations against the output of the system to check their correctness.

A total of 522 answers were gathered. This means that every sentence was answered on average 522 / 97 = 5.4 times. From 522 answers, more than half of them included word order errors (274). These sentences were free from other grammatical errors since they have been corrected by the English rater.

4.3.1 Experimental results

The findings from the experimentation (Figure 19) show that, according to the rater, 70% of the users’ answers have been repaired using our system or remained unchanged. For the rest of the sentences (30%), our system did not manage to correct the users’ word order errors. To explain the different results for our system using WSJ corpus (73% correct answers) and non native English corpus (70% correct answers), consider that non native speakers are prone to make more word order errors.
Summary and Conclusions

The findings from the experiments show that in the case of the TOEFL data set, 95% of the test sentences were detected and repaired using the proposed method. Moreover, 73% of the WSJ’s test sentences were suggested by our system as top ranked sentences, and for non-native English speakers’ corpus, 70% of the users’ answers were repaired using our system. This implies that the input sentence provides more valid trigrams compared with the rest of the permuted sentences and can be fixed using the proposed system. False alarms detecting ill-formed sentences is similar to repairing, ranging from 10 to 30%, because both use the same mechanism. The fast algorithm for filtering permutations works efficiently since with no filtering, the number of permutations is 479,001,600 while with filtering, it decreases to 11,378,400. If one word (i.e proper name) is missing, the proposed fast scheme will take into account all the possible permutations and evaluate them by means of trigrams. To compare the proposed system with the WORD 2000 grammar checker, a set of 100 sentences from the WSJ corpus was selected. For each sentence the position of a single word was changed randomly. For example, the following sentences were used:

A) “I had previously spent about two years asking local social services”.

B) “You should stress that your concern for human rights is not in any way politically partisan”.

The results showed that only 10% of the WSJ’s test sentences can be detected as ill formed by WORD 2000. On the other hand our system detected and repaired 73% of the test sentences as illformed.
The novelty of the proposed method is the use of a technique for filtering the initial number of permutations and the elimination of grammatical rules to repair sentences with word-order errors. Another aspect of the method is the ability to use it to distinguish different writing styles. The findings show that most of the sentences can be repaired by this method independently from the type of word order errors. Further consideration involves an evaluation of the proposed method using real data with misspellings. Fixing word order errors combined with misspellings certainly invites research. Also, another important issue is to clarify whether the incorporation of grammar information can improve the system’s performance. Five simple improvements over basic language models such as variable length N-grams, caching, skipping, clustering and sentence mixture models will be tested and compared.

References
Atwell, E.S., 1987a, “How to detect grammatical errors in a text without parsing it”. In Proceedings of the 3rd EACL, 38–45.


BNC http://www.natcorp.ox.ac.uk/corpus/index.xml


Mayfield Tomokiyo L., and Jones, R., 2001, You're Not From 'Round Here, Are You? Naive Bayes Detection of Non-native Utterance Text. Proceedings of NAACL, Pittsburgh,

More, J., 2006, “A grammar Checker based on Web searching” Digithum [online article]. Iss. 8. UOC.


TOEFL Test of English as a Foreign Language, Educational Testing Service http://toeflpractice.ets.org/