A Framework for Collocation Error Correction in Web Pages and Text Documents

Alan Varghese¹, Aparna S. Varde¹, Jing Peng¹ and Eileen Fitzpatrick²
1. Department of Computer Science  
2. Department of Linguistics  
Montclair State University, Montclair, NJ, USA  
(varghesea2 | vardea | pengj | fitzpatricke)@montclair.edu

ABSTRACT

Much of the English in text documents today comes from non-native speakers. Web searches are also conducted very often by non-native speakers. Though highly qualified in their respective fields, these speakers could potentially make errors in collocation, e.g., “dark money” and “stock agora” (instead of the more appropriate English expressions “black money” and “stock market” respectively). These may arise due to literal translation from the respective speaker’s native language or other factors. Such errors could cause problems in contexts such as querying over Web pages, correct understanding of text documents and more. This paper proposes a framework called CollOrder to detect such collocation errors and suggest correctly ordered collocated responses for improving the semantics. This framework integrates machine learning approaches with natural language processing techniques, proposing suitable heuristics to provide responses to collocation errors, ranked in the order of correctness. We discuss the proposed framework with algorithms and experimental evaluation in this paper. We claim that it would be useful in semantically enhancing Web querying e.g., financial news, online shopping etc. It would also help in providing automated error correction in machine translated documents and offering assistance to people using ESL tools.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications - data mining H.3.3 [Information Systems]: Information Search and Retrieval - search process, I.2.7 [Artificial Intelligence]: Natural Language Processing - machine translation, text analysis

General Terms

Algorithms, Documentation, Performance, Experimentation, Human Factors, Languages, Verification

Keywords

Web Queries, Automated Translation, ESL Tutors, News Articles, Online Shopping, Machine Learning, NLP

1. INTRODUCTION

We address the problem of collocation errors. These are errors where users enter expressions with words not typically used together, e.g., “powerful tea” and “quick cars”. These do not represent semantically correct collocations in English even though they are syntactically correct. For example, an expression such as “powerful tea” entered in a search for online shopping is not meaningful in correct native-speaker English. However, it is possible that users who are non-native English speakers, on translating from their native language, use such terms in Web queries or text documents. It is also possible that a machine translation from a foreign language document can have the same mistake. To the best of our knowledge, current systems, including automated machine translators, editors such as MS Word and search engines such as Google, do not automatically correct such errors or provide suggestions to users. For example, if a user enters “powerful tea” in a search engine during online shopping, the results contain “powerful” or “tea” or both. However, the user probably wants to search for the availability of “strong tea”. It would thus be useful to make the Web search more intelligent by providing correctly collocated responses to such errors and accordingly return semantically appropriate results. Note that from the NLP and Web search angles this is an issue of semantics, not syntax. For example, the term “powerful tea” is syntactically correct as it obeys the rules of English grammar, but it is semantically incorrect since it does not convey the intended meaning of the Web query. Likewise such collocation error correction is also directly applicable in the preparation of text documents where an ESL learning tool may be used. Similarly in machine translation such errors can occur and should be corrected to enhance performance.

There has been work on incorporating temporal changes in Web and text data, e.g., [6,7]. There is literature on correlated terms and ontology over the Web, e.g., [11,16]. Soderland et al. [15] deal with discourse analysis and inter-sentential inference generation. All these works, though somewhat relevant to ours, do not address flaws in collocation.

Futagi et al. [5] propose a method for identifying collocation errors using association measures over syntactic patterns. Ramos et al. [14] build an annotation schema with a 3-D topology to classify collocation. Dahlmeier et al. [3] suggests a method of using the native language of the L2 learner to correct collocation errors. Liu et al. [9] propose a probabilistic approach to collocation error correction using British National Corpus (BNC) and Wordnet for language learning tools. These works mainly address collocation for linguistic classification. Some of them require knowledge of the writer’s native language. Also, their focus is not on the semantics of Web queries and text documents, which we deal with in our work.

Park et al. [12] categorizes the different types of collocation errors into insertion, deletion, substitution, and transposition errors. Substitution or alternation errors occur when a non-preferred word is used in place of a more commonly used word. Substitution errors frequently result in collocation errors. In their approach they focus on frequency. While our work can be considered orthogonal to such works, use a variety of similarity measures and combine the measures using machine learning and natural
language processing techniques, in order to provide ranked correct suggestions to users.

The rest of this paper is organized as follows. Section 2 defines in detail the problem of collocation error correction as addressed in this paper. Section 3 describes our proposed framework called CollOrder. Section 4 explains the algorithms we have developed in CollOrder. Section 5 summarizes our evaluation. Section 6 gives the conclusions and future work.

2. PROBLEM DEFINITION

We first introduce the contexts in which collocation errors can occur and then state the precise goals of our work.

2.1 Tools for ESL in Text Documents

Much of the collocation related work that has been done can be applied directly to L2 English learners, when the L2 learner uses an odd collocation. It is useful to highlight such mistakes automatically and provide ranked suggestions. The challenge here is to provide only relevant suggestions. There are two steps to the procedure one is to flag or highlight a mistake and the second to provide relevant ranked suggestions. L1 paraphrasing may be a potential approach in this scenario provided the L1 language of the user of the ESL tool is known. Our goal however, is to develop an approach that can be applied in this situation without the knowledge of the L1 language of the user.

2.2 Automated Machine Translation

There are different papers that address the collocation errors in machine translation. The automated machine translation will most probably create the type of collocation error that a L2 learner makes and even more. In this scenario, L1 paraphrasing can usually be applied since the original language of the document is obviously known. Our goal of proposing a collocation error correction approach can still be applied to this problem without using any knowledge of the original language. It would thus be more generic.

2.3 Querying over Web Pages

Currently, search engines provide alternative suggestions for search expressions in Web queries so that the user can look at them to determine if one of them is better or relevant to their search. This often occurs if the user makes a spelling or grammatical mistake. However, it does not occur for collocation errors as far as we know. We next provide a motivating example to emphasize the importance of collocation error correction in search engines in order to return appropriate Web pages.

Motivating Example: Consider that a user wishes to search for financial news related to black money. If we conduct a Google search using the incorrect collocation “dark money” that could possibly arise due literal translation from the user’s native language, the following Web pages are returned as shown in Figure 1. While the search does provide some information, this is not exactly relevant to the user’s intended query. Now suppose that we provided the user with a suggestion of a correct collocation “black money” and the user searches with this expression thereafter. We clearly see that the resulting Web pages as shown in in Figure 2 are much more applicable to what the user originally intended.

Likewise users can conduct searches on other articles pertaining to financial news and other topics, and it is important to use correctly collocated terms. Another example is “stock market”. Using incorrect collocations such as “stock agora”, “stock mall” would yield undesired Web pages as search results, while using the correct collocation returns more appropriate Web pages including those with financial news on stocks, which would be more helpful. Online shopping is another context in which appropriate search terms are important. We have noticed that using the correct collocation “fast cars” and “strong tea” instead of the collocation errors “quick cars” and “powerful tea” yields better Web pages as responses to user queries, which would help to enhance the online shopping experience.
2.4 Problem Statement
Considering the various contexts provided here such as machine translation and Web search our goals in this work are twofold:

- Detecting collocation errors in L2 written English in the context of Web queries, machine translation and ESL tools for text documents.
- Providing ranked suggestions as responses to the collocation errors in order to assist users in the respective applications.

The first goal is thus to identify the collocation error in a Web search term or in a sentence in a text document. The next goal pertains to listing all possible corrections to the incorrect collocation and eliminating the improbable ones so that the list of suggestions provided to the user is minimal and as far as possible ordered by relevance in descending order.

3. PROPOSED FRAMEWORK
We propose a framework called CollOrder for detecting collocation errors and suggesting correctly ordered responses to them. This framework is illustrated in Figure 3. It consists of an error detection step followed by an error correction step that includes ranking.

![CollOrder Framework Diagram](image)

Figure 3: The CollOrder Framework

3.1 Error Detection
The input to CollOrder is an expression entered by a user. CollOrder has access to huge text corpora of correct English such as the American National Corpus and the British National Corpus, which we refer to as the “corpus databases”. In the error detection step, CollOrder first performs part-of-speech (POS) tagging, i.e., each term in the expression is assigned a part of speech based on its grammatical category: noun, verb etc. This POS tagging is performed on the assumption that parts of speech of the incorrect collocation match those of the correct collocation. After POS tagging, CollOrder searches for matches, comparing the given tagged terms with those in knowledge bases (KBs) of correct English, such as the American National Corpus, ANC (or the British National Corpus, BNC). If a match for the tagged term is found therein with a frequency greater than system-defined thresholds, the assumption is made that the user has entered a correct expression, e.g., “strong tea”. Thus, the approach would not execute further.

3.2 Error Correction
The error correction step executes if tagged terms in the user expression are not found in corpus databases, implying that a collocation error is detected, e.g., if the user enters “powerful tea”.

3.2.1 Search for Potential Collocates
To correct the error, CollOrder conducts a search for frequently used collocates of each tagged term in the expression, again using system-defined frequency thresholds. So, in this example, it would search for frequent collocates of “powerful” and of “tea”. It could discover several frequent collocates of “tea” such as “strong”, “potent”, “good”, “Indian” etc. These could yield potential responses like “strong tea”, “potent tea”. It could also include frequent collocates of “powerful” such as “drink”, “statement”, “person” and so forth. These could yield several potential responses such as “strong tea”, “Indian tea”, “powerful statement”, “powerful drink” as so forth.

Before proceeding further with the responses to be conveyed as suggestions to the user, we first explain the search process in CollOrder that deserves some attention.

3.2.2 Pre-Compute Collocates for Efficient Search
We have found that detailed searching and parsing over huge databases such as the ANC (with complete statements of correct English) is very time-consuming not feasible to execute recurrently each time a user expression is encountered.

Hence, we propose the following **CollOrder Search Heuristic:**
Instead of an exhaustive search over a huge database, a guided search over a smaller indexed knowledge base containing common collocates of selected parts-of-speech and their collocation frequency is equally effective and more efficient.

Thus, we generate smaller KBs (from corpora such as the ANC) by extracting relevant collocates with frequencies above the threshold. These are called the Collocate Frequency Knowledge Bases (CFKBs) and are far more condensed with only the relevant knowledge. For example the ANC is 10GB while its extracted CFKB is only 400MB. Thus, reduction in size is 86%, proportionately reducing search complexity.

We deploy techniques from [10] in our execution to parse sentences. With reference to our heuristic, we argue that after parsing it is useful to retain only those parts of speech relevant for collocation. From the NLP angle, these are abstracted as:

3.2.2.1 nn: noun compound modifier
A noun compound modifier of an NP is any noun that serves to modify the head noun, e.g., “Oil price futures” nn(futures, oil).

3.2.2.2 advmod: adverbal modifier
An adverbal modifier of a word is a (non-clausal) adverb or adverbal phrase (ADV) that serves to modify a word used in an adverbial sense, e.g., “Genetically modified food” advmod(modified, genetically)

3.2.2.3 amod: adjectival modifier
An adjectival modifier of an NP is an adjectival phrase serving to modify the following noun, e.g., “Sam eats red meat” amod(meat, red)
3.3 Ranking the Suggestions

We propose to rank the suggestions using the data mining approach of classification such that it encompasses several measures. The various tasks involved in ranking are as follows.

3.3.1 Pre filtering the suggestions

Prior to performing any calculations for ranking, CollOrder adopts a pre-filtering approach. It filters out collocates that are close to being antonyms of the input collocate. For example, consider the expression “powerful tea”. A collocate such as “light” could also be discovered for “tea”. However, “light tea” conveys almost the opposite meaning as “powerful tea” and hence should be removed from the list of responses to be suggested to the user. This filtration is conducted by using a synonym database and removing the terms that do not occur therein.

3.3.2 Selecting Measures for Ranking

We propose to deploy the following different measures for ranking as they capture various useful aspects of the data. The respective aspects are explained along with each measure.

3.3.2.1 Conditional Probability

In probability theory the conditional probability of A given B is the probability of A occurring if B is known to occur. It is formulated as:

\[ P(A|B) = \frac{P(A \cap B)}{P(B)} \]

With reference to our work, this translates into

\[ \frac{\text{Freq}(A \text{ and } B)}{\text{Freq}(B) \times \frac{N}{\text{Freq}(B) / N}} \]

where N is the total number of words, while A and B are the two terms in the given expression.

Hence, this equates to: \( \frac{\text{Freq}(A \text{ and } B)}{\text{Freq}(B)} \).

For example, consider the suggestion “strong tea” as a response to “powerful tea”. In this context, we calculate the probability of “tea” given “strong”. It is useful to obtain this information because it helps us determine the relative occurrence of the terms. It would be helpful to know how often the term “tea” occurs in conjunction with “strong”, in order to convey it as a suggestion.

3.3.2.2 Jaccard’s Coefficient

We use the Jaccard similarity coefficient as a measure of semantic similarity. It is defined as the size of the intersection divided by the size of the union of the sample sets. The formula is:

\[ J(A, B) = \frac{|A \cap B|}{|A \cup B|} \]

We calculate Jaccard’s coefficient in CollOrder using the following method. The co-occurrence of two terms \( \text{Freq}(A \cap B) \) is calculated by searching through the corpus for the two words with a word window of 32 which we define as co-occurrenceWindow. We refer to the paper Terra et al. [17] to find a suitable window size.

The individual frequencies \( \text{Freq}(A) \) and \( \text{Freq}(B) \) are obtained by searching through the corpus using a search engine.

Thus, Jaccard’s coefficient is calculated using

\[ \frac{\text{Freq}(A \text{ and } B)}{\text{Freq}(A) + \text{Freq}(B)} \]

We consider this measure because it is important to measure this semantic similarity between the term in the user expression and the potential responses to convey suggestions to users. Thus, in using Jaccard’s, we would measure the extent of the semantic similarity between “strong” and “powerful”, “potent and powerful” and so forth, in order to find the appropriateness of

3.3.2.3 Other

The other measures considered include:

- Co-occurrence Window Size
- Part-of-Speech (POS) Tag
- Frequency of Collocates
- Jaccard’s Coefficient
- Conditional Probability

3.2.4 \textit{prt}: phrasal verb particle

The phrasal verb particle relation identifies a phrasal verb, and holds between the verb and its particle. Example: “They shut down the station” \textit{prt}(shut, down).

The knowledge bases hereby generated are several orders of magnitude smaller than the original text corpus databases and contain relevant information for searching of collocates. These can then be used to execute the searches each time a user expression is encountered, which is very efficient.

In order to execute this efficient searching, we implement programs to pre-compute the frequency of collocates and the part-of-speech (POS) tag of collocates found in the original text corpus databases such as ANC. Figure 4 summarizes this approach.

![Figure 4: Creating a CFKB using a corpus DB such as ANC](image)

The first module in Figure 4 deploys the well-known Stanford Parser to parse all the text files in the corpus databases. The American National Corpus is shown here as an example but the same logic applies to other corpora such as the British National Corpus. This module generates a file containing collocates and the POS tag associated with each collocate. The next module in this figure uses the file generated by the first one and creates a relational database containing collocates and the frequency of occurrence along with the POS tag associated with each collocate.

The Collocate Frequency Knowledge base serves as a corpus of collocates in the English Language. Although this is limited to the collocations in the corpus that we use, we argue that ANC and BNC are very comprehensive and the experimental results have proven that it is very relevant. More importantly we would like to highlight one of our contributions to collocation error detection and correction. To the best of our knowledge, this method of pre-computing and materializing collocates in a CFKB accompanied by the Collocate Search Heuristic is unique as an approach for efficient searching of collocates.

Once this efficient searching has been conducted, several potentially correct collocates of terms within the user expression would be discovered. Not all of these would be meaningful in the given context and hence we cannot simply output all of them as suggested responses. It is thus important to conduct filtering and ranking and then convey useful suggestions as the output of CollOrder. We thus proceed to explaining the ranking next. Even though the ranking forms part of the Error Correction step, we are explaining it here as a separate subsection since it encompasses several concepts.
terms such as “strong” and “potent” in being conveyed as the suggested responses “strong tea” and “potent tea” respectively.

### 3.3.2.3 Web Jaccard

Web Jaccard is a semantic similarity measure which is slightly modified form of the Jaccard’s coefficient shown above in that we remove the frequency of intersection of terms from the sum of the individual frequency of occurrences. While searching text documents we propose to count the situations where A and B occur together only once in $\text{Freq}(A \cup B)$ which is the denominator in the Jaccard coefficient.

$$\text{Web Jaccard}(A,B) = \frac{\text{Freq}(A \cup B)}{\text{Freq}(A) + \text{Freq}(B) - \text{Freq}(A \cup B)}$$

We calculate the individual terms in the Web Jaccard similar to the manner in which we calculate the Jaccard coefficient and then we apply the above formula to obtain the Web Jaccard. The rationale behind this measure is therefore almost the same as the rationale behind Jaccard’s coefficient with a minor twist for additional robustness, in order to experiment with more measures.

### 3.3.2.4 Frequency Normalized

In addition to other measures, we also consider the fundamental collocation frequency. However, order to conduct ranking based on various measures, it is helpful to normalize the values for frequency since the other measures are all in the range of 0 to 1.

We perform the normalization as follows. From the list of suggested responses, we find the one with the highest frequency and the lowest frequency and consider these as the upper and lower limits for the original range. We then map all values in that range to values between 0 and 1, thereby normalizing them.

This is formulated as follows. For all the expressions considered as suggested responses:

- $\text{Min} = \text{frequency of the lowest occurring collocate}$
- $\text{Max} = \text{frequency of the highest occurring collocate}$

For any given collocate:

$$\text{Frequency Normalized} = \frac{\text{Frequency of collocate} - \text{Min}}{\text{Max} - \text{Min}}$$

### 3.3.2.5 Frequency Ratio

We hereby propose a measure such that its value is normalized between 0 and 1 and such that it has a higher value if the terms co-occur very frequently and a lower value if the terms co-occur less frequently compared to their individual occurrences. This is a measure purely based on frequency of collocation and does not consider similarity with the original expression. For example, if the original user expression is “powerful tea” and a potential suggestion is “strong tea”, the frequency ratio is calculated between “strong” and “tea”. Note the difference between this one and conditional probability. This measures the co-occurrence of both terms while conditional probability measures the occurrence of one term given another. Our rationale for proposing this one is to experiment with a variety of measures including those that do not incorporate the original user expression. The formula for frequency ratio is given as:

$$\text{Frequency Ratio} = \frac{\text{Freq}(A \cup B)}{\text{Freq}(A) + \text{Freq}(B)}$$

### 3.3.3 Combining the Measures

Empirically, it has been found that different measures yielded good results in different scenarios. We therefore propose to deploy the machine learning paradigm of classification here for combination of measures in order to optimize performance.

We consider each measure as an attribute for the classification task and introduce a target class attribute. We mark all collocations as class n (No) except the ones that are correctly classified a y (Yes). Please refer a sample set given below in Figure 5. This is an example of the training set for the machine learning classifier.

We run a classifier (JRIP) which is an implementation of RIPPER. This classifier implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which was proposed by William W. Cohen as an optimized version of IREP (W.W. Cohen [1]).

<table>
<thead>
<tr>
<th>Input</th>
<th>Suggestion</th>
<th>Conditional prob</th>
<th>freqNorm</th>
<th>Jaccard</th>
<th>freqRatio</th>
<th>Web Jaccard</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>pure</td>
<td>blue</td>
<td>0.02640207</td>
<td>0.836207</td>
<td>0.002956</td>
<td>0.013882</td>
<td>0.002160525</td>
<td>y</td>
</tr>
<tr>
<td>pure</td>
<td>night</td>
<td>0.004351655</td>
<td>0.607759</td>
<td>0.0008523</td>
<td>0.00834</td>
<td>0.0005233</td>
<td>n</td>
</tr>
<tr>
<td>pure</td>
<td>clear</td>
<td>0.00132547</td>
<td>0.094328</td>
<td>0.00177</td>
<td>0.001113</td>
<td>0.00177349</td>
<td>y</td>
</tr>
<tr>
<td>pure</td>
<td>water</td>
<td>0.00139753</td>
<td>0.139483</td>
<td>0.002857</td>
<td>0.002775</td>
<td>0.002663807</td>
<td>n</td>
</tr>
</tbody>
</table>

**Figure 5: Subset of the Training Set for Machine Learning**

We explain the process as follows. We first run the CollOrder program without applying any filtering on a set of incorrect collocates. This yields a comma separated dataset similar to that shown in Figure 5. Each row contains a potential suggestion and also contains all the measures of similarity and frequency and the class is by default n (No). The rows containing correct collocates and the ones that are highly likely to be correct choices are marked as y (Yes). This serves as the training set.

The input to the JRIP algorithm is a comma-separated value file (csv) as shown in Figure 5. The algorithm prunes the rules and produces the following rule which is then used in our program within the ranking section of CollOrder.

$$((\text{Jaccard} >= 0.00338) \text{ AND (frequencyRatio} >= 0.002833))$$

OR

$$((\text{jaccard} >= 0.00177) \text{ AND (frequencyNormalized} >= 0.811321))$$

We have found that this rule has been effective in listing out only the required top-k collocates and thus it automatically determines the value of k, i.e., the number of suggestions to be output for the given user expression (collocation error). So we can assume that Jaccard’s coefficient and Frequency Ratio measures are better than the other measures we considered for this particular situation. Likewise, suitable training sets can be provided for an exhaustive list of common collocates in other situations and the system would learn the rules that are required for ranking the collocates. Upon running this procedure in CollOrder, we have obtained an effective ranking for a variety of examples that yielded good results experimentally. We have considered various user expressions that are incorrect collocations and conducted the ranking of suggested responses using the procedure described herein. This output of CollOrder including the correct responses given to users along with the ranking in various scenarios has been evaluated as effective by detailed user surveys. This is further corroborated in our section on evaluation.

To the best of our knowledge, such an approach of deploying a classifier to combine measures of similarity in the context of collocation error correction has not been used in the literature. Hence we claim this to be a unique contribution on our part.
Note that throughout in the explanation of CollOrder we have explained the logic in our approach using two terms in a collocation. However, the same logic holds good for three terms. For example if the user enters an expression such as “incomplete money help”, instead of the more appropriate “partial financial aid” the terms “incomplete money” and “money help” would both be considered separately and the approach would proceed in the same manner to provide suggestions.

In the rare case that no correct collocation is found within threshold in our CFKB, the system indicates that a suitable response is not obtained from our knowledge bases. This is an issue of sparse data to be addressed as future work.

4. ALGORITHMS
We now proceed with outlining the algorithms that we developed in the CollOrder approach. Figure 6 gives the details of the CollOrder approach based on the explanation given in Section 3.

4.1 Pre-computing Collocates in CollOrder
The algorithm for pre-computing collocates is given below as Algorithm 1. This algorithm determines the “correct collocates” and creates the CFKB. We run the Stanford parser on each of the text documents in the ANC. We find the tagged items present in each of the sentences. If the “Part of Speech Tag” is among those we decided to consider then we store the tagged item as a “collocate” in the CFKB. The function “storeinCFKB” increments the frequency by 1 if it already exists or else it stores it with a frequency of 1.

```java
for all document ∈ ANC do
    sentences ← parse(document)
    for all sentence ∈ sentences do
        for all tagged item ∈ sentence do
            if POSTAG ∈ ('amod', 'advmod', 'n', 'prf') then
                storeinCFKB(taggedItem)
            end if
        end for
    end for
end for
```

Algorithm 1: Pre-computing of Collocates in CollOrder

4.2 CollOrder Main Routine
The main algorithm or routine that invokes the whole CollOrder approach is given next as Algorithm 2. The input to CollOrder is a sentence. First the sentence is parsed to find the tagged items in the input. If the tagged item has a POS tag that we are considering then we check in our CFKB to see if it is present with a frequency greater than threshold (we use 10 as the threshold). If not present then we call our error correction routine. Although we have explained this with a sentence, the same logic applies for a search expression in a Web query or any expression / sentence in machine translation.

```java
function COLLORDER(sentence)
    taggedItems ← parse(sentence)
    for all item ∈ taggedItems do
        if posTag ∈ ('amod', 'advmod', 'n', 'prf') then
            cFreqItem ← storeinCFKB(item)
        end if
    end for
    if cFreqItem = null then
        ERRORCORRECTION(item)
    end if
end function
```

Algorithm 2: CollOrder Main Routine

4.3 Error Correction in CollOrder
The following algorithm, namely Algorithm 3, gives the details of error correction in the CollOrder approach.

In this algorithm, the ERRORCORRECTION procedure:

- Searches though the CFKB for potential suggestions.
- Filters out antonyms
- Mark the suggestions that are synonyms of the input collocate
- Calls the CALCULATEMEASURES routine.
- Calls the FILTERBYJRIPRULES routine to filter out the results by applying the JRIP rules that we extracted to the parameters we calculated in the CALCULATE measures routine.
- Prints out the top-k results in the order of correctness of collocation based on the ranking.

![Diagram](image-url)
Having discussed these algorithms we developed in CollOrder, we now proceed with the details of our experimental evaluation.

Algorithm 3: Error Correction in CollOrder

5. PERFORMANCE EVALUATION

We have developed a software tool based on our CollOrder framework. This tool accepts any expression as an input from the user and if a collocation error is detected it provides a list of suggestions to the user ranked in the order of correctness. To evaluate the effectiveness of this tool and hence our CollOrder framework for detection and correction of collocation errors, we have conducted a performance evaluation using Mechanical Turk as described next.

5.1 Evaluation using Mechanical Turk

The evaluation was performed using Amazon mechanical Turk. 20 screen shots of the CollOrder web application was provided to the evaluators. Each evaluation is called a Human Intelligence Task or HIT and it would be presented like what is shown below.

![Figure 7: Evaluation with Mechanical Turk](image)

Amazon Turk allows us to download the results as a csv which gives us the url of the image that we asked the user to evaluate and the columns with the two entries that the users provided.

Out of the 225 reviews done we find that 92.44% of the time the users agreed that a correct alternative to the collocation errors, i.e., odd sounding collocations, was suggested. This indicates that the error correction step is indeed functioning well.

5.2 Effectiveness of CollOrder in Web Search

Let us assess the usefulness of CollOrder with respect to search results in Web pages. Consider a user evaluation example where a non-native speaker incorrectly enters an expression such as “quick car” in a Web query, when they really meant to search for “fast cars”. This could for example occur in the context of online shopping. The response given by a search engine for this expression is shown in Figure 9. We have 1,270,000 results which do not seem very relevant to the user’s really intended query.

Consider the response given by the CollOrder framework in this situation with the user expression “quick car” which is detected as a collocation error.

![Figure 8: Example Response of CollOrder](image)

We notice that CollOrder outputs only one suggestion here as shown in Figure 8 based on the execution of the search and ranking. Consider that the user conducts the Web search with the suggestion “fast car” given by CollOrder as a correct collocation. We notice the following search results as shown in Figure 10. We now have 10,800,000 results which do not seem very relevant to the user’s really intended query.

The Google suggestions are also more relevant to the intended search.
Likewise, we find that the suggestions given by the CollOrder framework can indeed improve the effectiveness of query results in Web pages. These can be very useful in applications such as online shopping.

While we have evaluated the effectiveness of CollOrder in the context of Web querying and its applications, similar justification can be applied in other contexts such as ESL tools for text documents and automated machine translation. We briefly explain these below.

### 5.3 Potential Use in Machine Translation

A full consideration of the use CollOrder in machine translation is beyond the scope of our current paper. However, we present the following as potential effectiveness of the framework.

Machine translation software can convert a sentence from a foreign language to English. This sentence can then be sent to CollOrder. The CollOrder framework can determine whether the...
collocations in the sentence are correct. If they are incorrect, it can provide ranked suggestions as correct collocations. The machine translation software can then decide whether the sentence needs to be changed. If so, it can either change the collocation to the suggested collocations, or else flag it for manual review with a list of ranked suggestions.

5.4 Potential Use in ESL Writing Aids

The CollOrder framework can also assist in evaluating text written by non-native speakers such that it can be useful in ESL writing aids. Again, this has not been implemented but is considered as potential effectiveness of CollOrder. The process of text evaluation in the writing aid could be as shown in Figure 11.

The software can read each sentence and invoke CollOrder for each sentence. If CollOrder marks the collocations as odd then the suggestions can be used to correct the text or provide a suggestion list for manual review.

![Figure 11: Text evaluation flowchart](image)

5.5 Discussion on Further Challenges

Based on the performance evaluation of CollOrder, it is clear that effective responses to collocation errors are provided. However, there is scope for more improvement. In this context, we outline further challenges as discussed below.

**Domain Knowledge:** A significant challenge includes capturing knowledge on domain-specific aspects for several text corpora, e.g., those in scientific fields. Such knowledge may not be present in standard databases such as the American National Corpus, as a result of which we may face the problem of not finding any correct response to a given expression. In order to address this we propose to include several other domain-specific databases in our search besides standard corpus databases of correct English sentences, which in turn brings us to another challenge.

**Literary Allusion:** It is challenging to define a precise notion of correctness to clearly distinguish between acceptable and unacceptable terms in collocation, considering issues such as literary allusion, e.g., “inanimate human”. If a writer has used such an expression, there seems to be no appropriate response here. If this term appears in the database of correct English it could possibly be a poetic term, although with respect to common parlance it should be detected as an error. Thus the notion of correctness needs to take into account such anomalies. If we have domain-specific databases on various literary subjects then this challenge is even more prominent.

**Sparse Data:** The issue of correct but sparse data poses an issue, which could also lead to the problem described earlier of not finding any correct responses within a given threshold. Some terms may not show up in text corpora even though they are correct expressions. It is therefore important to address this issue and propose appropriate solutions.

These and other such challenges provide the potential for further research in the area of collocation error correction.

6. CONCLUSIONS

In this work, we have proposed a framework called CollOrder to detect and correct collocation errors in written English. This is useful in the context of searches in Web pages, ESL help in text documents and also in automated machine translation. The main contributions of this research include:

- Proposing the overall CollOrder framework based on an integrated approach incorporating natural language processing and machine learning
- Proposing efficient searching in CollOrder through search heuristics and pre-computation of collocates with materialization
- Proposing a method to rank collocates in CollOrder based on classification and similarity measures
- Implementing the CollOrder framework using real data and developing a GUI for user interaction
- Conducting detailed user surveys and objectively evaluating the effectiveness of CollOrder
- Providing inputs as correct collocations in various contexts such as Web queries, ESL tools and machine translation to assist L2 users

This work would be useful to developers of intelligent tutoring systems, search engines, text editors and translation tools. It would be of interest to the database, data mining and AI communities due to emphasis on aspects such as search and ranking, natural language processing, machine learning and Web data. It would also appeal to users from various domains such as finance, marketing and linguistics.

Future work includes addressing challenges such as sparse but correct data, literary allusion and domain knowledge. There is potential for further enhancing the performance of the CollOrder framework by using powerful computers and an in-network synonym and antonym dictionary. We could also host the search engine for the ANC /BNC in-house on a powerful server. This would make the framework more efficient thus providing even better user satisfaction.

7. REFERENCES


