Collocation Error Correction in Web Queries and Text Documents

A THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Science in the Graduate Program of Montclair State University

By

ALAN T VARGHESE

Montclair State University

May 2013

Collocation Error Correction in Web Queries and Text Documents

Abstract

Collocations are words in English that occur together frequently. Non-native speakers of English tend to confuse certain terms with other similar terms. This causes one of the terms to be substituted with a term that is not commonly used with the other term. “Powerful tea” is an example of such an odd collocation. In this scenario the more commonly used term is “strong tea”.

This paper proposes an approach called CollOrder to detect such odd collocations in written English. CollOrder also provides suggestions to correct the odd collocate. These suggestions are filtered and ranked as top-k suggestions.

We make use of large text corpora such as the American National Corpus (ANC) to identify the common collocations and we use search heuristics to speed the search of collocations and preparing a list of suggestions. We have created a Collocation Frequency Knowledge Base (CFKB). We combine various measures of similarity and frequency of usage using a machine learning classifier to arrive at a formula that can be used to filter and rank the top-k suggestions.

We have implemented a web based solution to evaluate the approach and have considered factors such as caching of intermediate results to enable it to be used in real time.

We claim that our approach would be useful in semantically enhancing Web information retrieval, providing automated error correction in machine translated documents and offering assistance to students using ESL tools.

MONTCLAIR STATE UNIVERSITY

Collocation Error Correction in Web Queries and Text Documents

by
Alan T Varghese

A Master's Thesis Submitted to the Faculty of
Montclair State University
In Partial Fulfillment of the Requirements
For the Degree of
Master of Science
May, 2013

School College of Science and Mathematics

Department Computer Science

Certified by:

_____
Dr. Robert Prezant
(Dean)

_____
Date

Thesis Committee:

_____
Dr. Aparna Varde
Thesis Sponsor

_____
Dr. Eileen M Fitzpatrick
Committee Member

_____
Dr. Anna Feldman
Committee Member

_____
Dr. James Benham
Department Chair

Author Note

This thesis would not have been possible without the guidance of my advisor Dr. Aparna S. Varde. She has gone above and beyond her responsibilities to help and guide during this long process for patiently listening to me and reviewing my work even over weekends and for guiding me through the research process, for imparting so much valuable knowledge and for all her encouragement and words of kindness.

Dr. Eileen's expertise in this field has been a major driving force behind this thesis. I cannot express enough gratitude for her taking time during her sabbatical to help and guide me in this process. I thank her for all the help she has provided.

Dr. Jing's expertise in machine learning has helped me implement a solution that produces acceptable results. The training he provided with tools like WEKA, and the entire machine learning course was the basis of the entire implementation. I thank him for all the guidance he has provided.

I also want to thank my family who has been very supportive. I also thank the people who provided feedback on the implementation.

Contents

Table of Figures

Table of Algorithms

1. INTRODUCTION

We address the problem of collocation errors. The term collocation (Firth, 1957) describes a sequence of words that is conventionally used together in a particular way by native speakers and appears more often together than one would expect by chance. The correct use of collocations is a major difficulty for ESL students (Farghal and Obiedat, 1995)[10]. These are errors where users enter expressions with words not typically used together, e.g., “powerful tea”, “high women” and “crimson tape”. These do not represent semantically correct collocations in English even though they are syntactically correct. An expression such as “powerful tea” 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. 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. For example, if a user enters “powerful tea” in a search engine, the results contain “powerful” or “tea” or both. However, the user probably means to search for sources of information on related terms such as “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., [14,16] Hou, Kaluarachi. There is literature on correlated terms and ontologies over the Web, e.g., [23,31] Miller, Suchanek. Soderland et al. [30] deal with discourse analysis and intersentential inference generation. All these works, though somewhat relevant to ours, do not address flaws in collocation.

Futagi et al. [12] propose a method for identifying collocation errors using association measures over syntactic patterns. Ramos et al. [26] build an annotation schema with a 3-D topology to classify collocation. Dahlmeier et al.[8] suggests a method of using the native language of the L2 learner to correct collocation errors. Liu et al. [21] 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. Their focus is not on the semantics of Web queries and text documents, which we deal with in our work.

Park et al. [24] 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. For example, “make homework” should be “do homework,” and “clean sky” should be “clear sky.” This is the type of error that we are focusing on in our work. In their approach they use edit distances and frequency for error detection and correction. Weeds, J (2005) [35] define the co-occurrence retrieval using precision and recall measures and they calculate similarity using additive and difference-weighted models. We, in addition to frequency, use a variety of similarity measures and combine the measures using data mining techniques, in order to provide ranked correct suggestions to users.

The rest of this thesis is organized as follows. In section 2 we present a literature survey and section 3 defines in detail the problem of collocation error correction as addressed in this paper. Section 4 describes our proposed solution approach called CollOrder. Section 5 explains the algorithms we have developed in CollOrder. Section 7 summarizes our evaluation and Section 8 describes some potential

applications of CollOrder Section 9 gives the conclusions and future work. The references are listed at the bottom.

2. Literature Survey

In this section we present a brief survey of related work in this area

2.1 Correcting Semantic Collocation Errors with L1-induced Paraphrases (Dahlmeier et al [8])

They present a novel approach for automatic collocation error correction in learner English which is based on paraphrases extracted from parallel corpora. Their key assumption is that collocation errors are often caused by semantic similarity in the first language (L1-language) of the writer. They show that L1-induced paraphrases outperform traditional approaches based on edit distance, homophones, and WordNet synonyms.

Their key observation is that words are potentially confusable for an EFL student if they have similar translations in the writer's first language (L1-language), or in other words if they have the same semantics in the L1-language of the writer.

While these types of L1-transfer errors have been known in the EFL teaching literature (Swan and Smith, 2001[32]), research in Grammatical Error Correction has mostly ignored this fact. The key component in their approach generates L1-induced paraphrases which we automatically extract from an L1-English parallel corpus.

They are primarily interested in finding candidates which are not substitutable in their English context but appear to be substitutable in the L1-language of the writer, i.e., one forms a grammatical English sentence but the other does not.

L1-transfer: They suspect that an error is caused by L1-transfer if the erroneous phrase and its correction share a common translation in a Chinese-English phrase table. Their method is applicable to any language pair where parallel corpora are available.

They use the popular technique of paraphrasing with parallel corpora (Bannard and Callison-Burch, 2005[1]) to automatically find collocation candidates from a sentence-aligned L1-English parallel corpus.

2.2 Automatic Identification of Non-compositional Phrases[19]

In machine translation, word-for-word translation of non-compositional expressions can result in very misleading (sometimes laughable) translations. In information retrieval, expansion of words in a non-compositional expression can lead to dramatic decrease in precision without any gain in recall. Less obviously, non-compositional expressions need to be treated differently than other phrases in many statistical or corpus-based NLP methods.

One possible way to separate compositional phrases and non-compositional ones is to check the existence and mutual information values of phrases obtained by substituting one of the words with a similar word. A phrase is probably non-compositional if such substitutions are not found in the collocation database or their mutual information values are significantly different from that of the phrase.

A method to identify non-compositional phrases.

The method is based on the assumption that non-compositional phrases have a significantly different mutual information value than the phrases that are similar to their literal meanings. Their experiment shows that this hypothesis is generally true. However, many collocations resulted from systematic parser errors also tend to posses this property.

There have been numerous previous research on extracting collocations from corpus, e.g., (Choueka, 1988) and (Smadja, 1993) [3,29]. They do not, however, make a distinction between compositional and non-compositional collocations. Mutual information has often been used to separate systematic

associations from accidental ones. It was also used to compute the distributional similarity between words (Hindle, 1990; Lin, 1998)[13,20].

2.3 AwkChecker: An Assistive Tool for Detecting and Correcting Collocation Errors [24]

Detecting and correcting collocation preferences shares some similarities with spell checking. In spell checkers, words are compared to a dictionary. If the word is found, it passes and remains unflagged. However, if the word is not present in the dictionary, spell checkers suggest a list of alternatives, typically by measuring the edit distance between the typed word and dictionary words using a function called the Levenshtein distance. While they make use of a similar approach in their L2 error detector, unlike spell checking, no dictionary exists for collocation preferences. Thus, a dictionary must first be constructed before we can detect or correct awkwardness. AwkChecker builds a dictionary of n-grams (sequences of words) from a given corpus and records frequencies of each sequence within the underlying corpus. At present, AwkChecker builds a dictionary of 2-5 word phrases. All 2-5 word phrases contained in the corpus have associated frequencies in the dictionary. While n-grams are not a novel technique, L2 language error detection and correction typically use more complex linguistic models such as decision trees, statistical machine translators, and others.

They define a function that measures the acceptability of a phrase. Given a phrase e, the acceptability of e heavily depends on the actual usage frequencies. However, if there are better alternatives, the likelihood of e being awkward increases.

The function A(e) captures these factors by comparing the frequency of phrase e, g(e), to the product of the cost of transforming e into c, f(e,c) and the frequency of c, g(c), for the best alternative phrase in the corpus. If A(e) is less than a user-customizable threshold, the phrase e is flagged as a collocation error. To efficiently compute A(e), they employ a search engine (inverted index), the Wumpus Information Retrieval System (Buttcher, S., and Clarke, C)[2].

Correction of an awkward phrase requires a candidate list of alternative phrases to be created. They use the Levenshtein distance metric to generate a list of candidates.

2.4 Our Approach

In the following sections we will see that our approach is orthogonal to many of the literature and has specific advantages over all of the above mentioned approaches. Some of them don't provide the filtering and ranking that we provide and some of them need the existence of parallel corpora. Our approach is a more generalized one based on machine learning and provides filtering and ranking which makes it much more practically useful.

3. PROBLEM DEFINITION

In order to proceed with defining the problem more specifically, we first introduce the contexts in which collocation errors can occur. Note that L1 learners of English are native speakers of English. L2 English learners are people who are non-native speakers of English and their native language is considered as *their* L1 language. Refer to the work of Cook et. al [5] for details about L1 and L2 languages.

3.1 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. Refer to Park et, al [24].

3.2 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.

3.3 Semantic Suggestions in Search Engines

Currently search engines provide alternative suggestions for search terms so that the user can look at that to determine if one of them is better or relevant to the users' search. The user input in a search engine is usually a search term or a short sentence. It has been noticed that if there are collocation errors in this search the user will not get the expected results. Our approach is aimed at enabling search engines to provide ranked suggestions to the user. To the best of our knowledge, an approach for collocation error correction in search engines has not yet been proposed. We next provide a motivating example to emphasize the importance of collocation error correction in search engines.

Motivating Example: If we conduct a Google search for "powerful tea" the following results are returned as shown in Figure 1. Assuming that the L2 learner made a mistake in the collocation and really intended to search for "strong tea" we see that the results are not relevant.

Now let us assume that we provided the user with a suggestion of "strong tea" and the user searches for "strong tea" thereafter.

We clearly see that the results in Figure 2 are much more applicable to what the user originally intended. We even have images of strong tea shown in the search result. Also notice on the bottom the user now gets suggestions based on his search, which was absent in the original search.

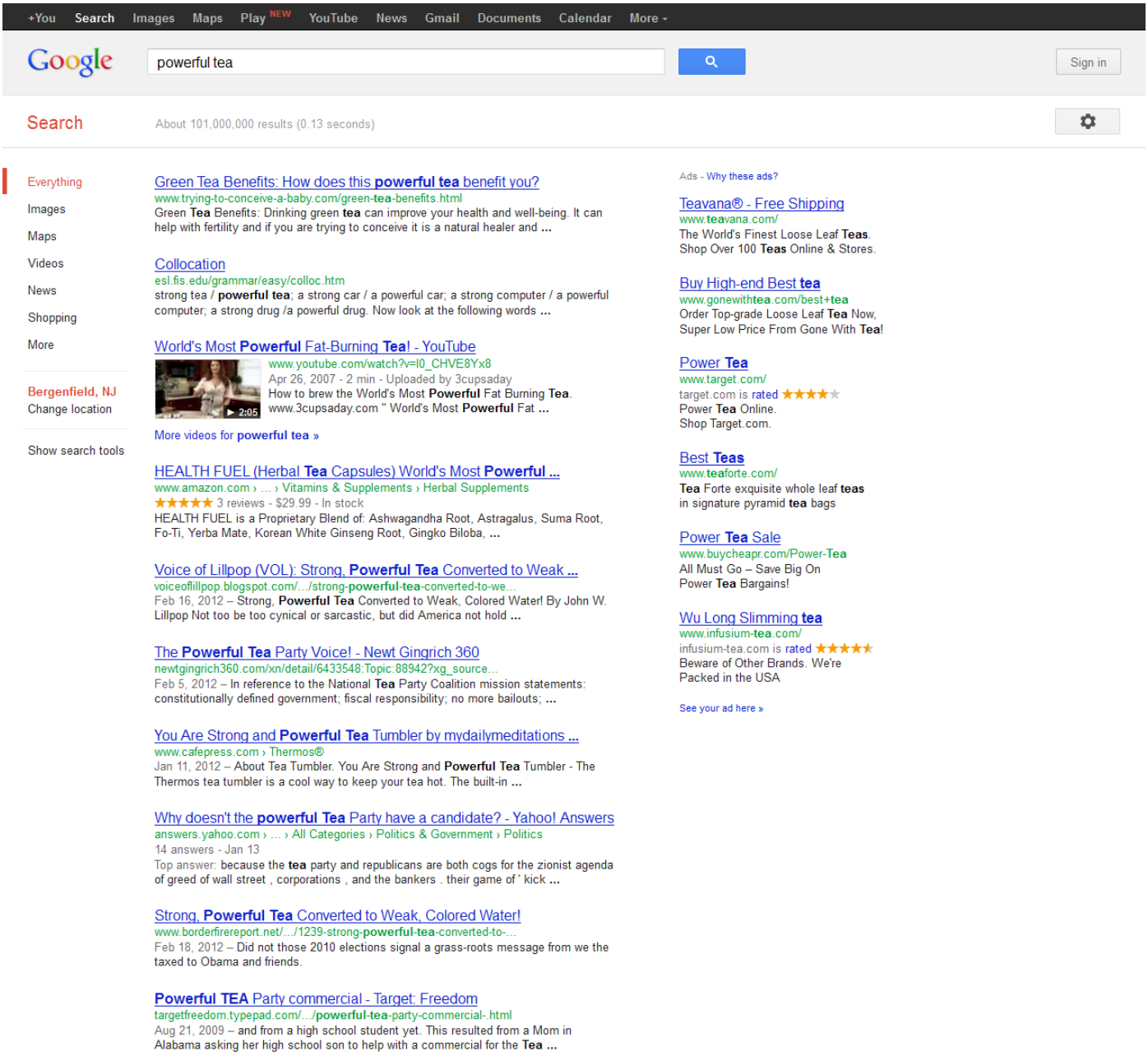


Figure 1: Example of a collocation error "powerful tea" in a Web search

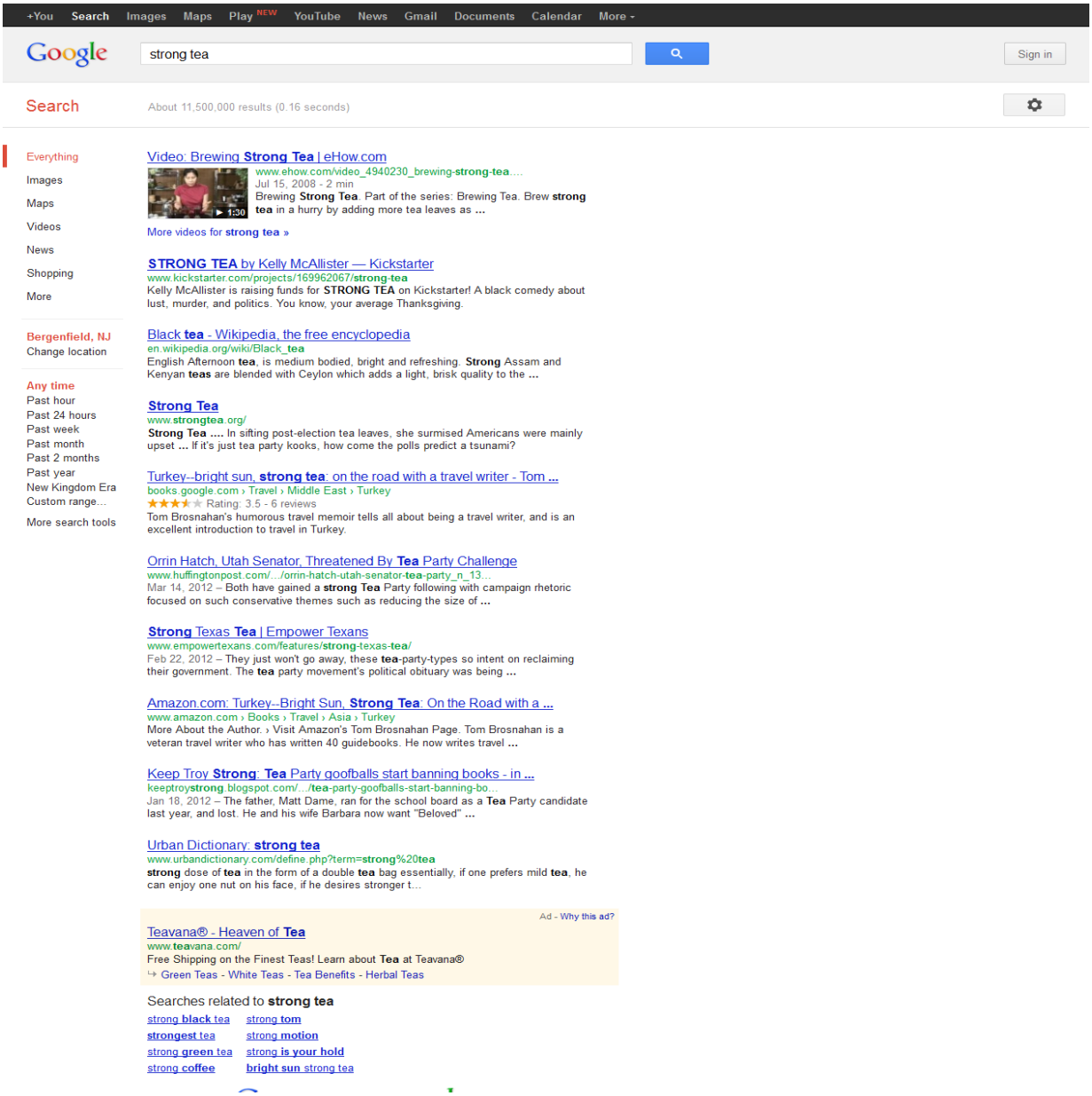


Figure 2: Searching using a correct collocation "strong tea"

3.4 Problem Statement

Considering the various contexts provided here such as machine translation, ESL learning aids 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.

4. PROPOSED APPROACH: CollOrder

We propose the CollOrder approach for detecting collocation errors and suggesting correctly ordered responses to them. This approach is outlined in Figure 3. It consists of an error detection step followed by an error correction step that includes ranking.

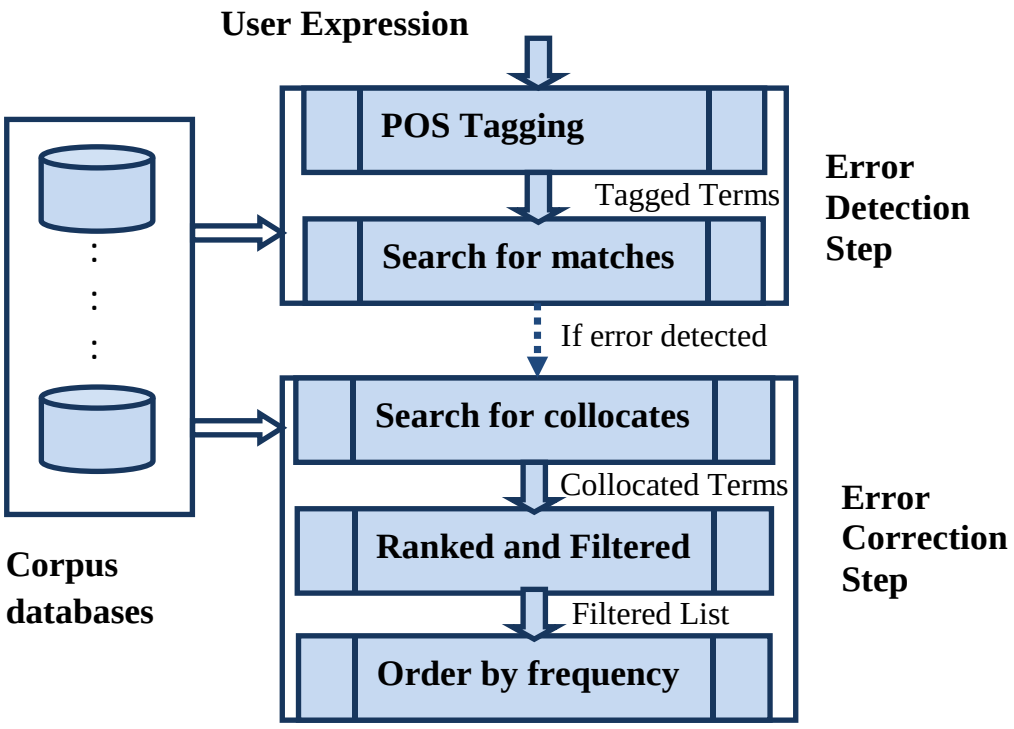


Figure 3: The CollOrder approach

4.1 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.

4.2 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".

4.2.1 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 potentially correct 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.

4.2.2 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 / BNC) 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 Marnaffe [22] 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:

4.2.2.1 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).

4.2.2.2 advmod: adverbial modifier

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

4.2.2.3 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)

4.2.2.4 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" prt(shut, down).

We only consider these four types as collocates for our evaluation. We used a list of correct collocations from Distributional Semantics and Compositionality (DISCO 2011) and found that these covered 80%.

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.

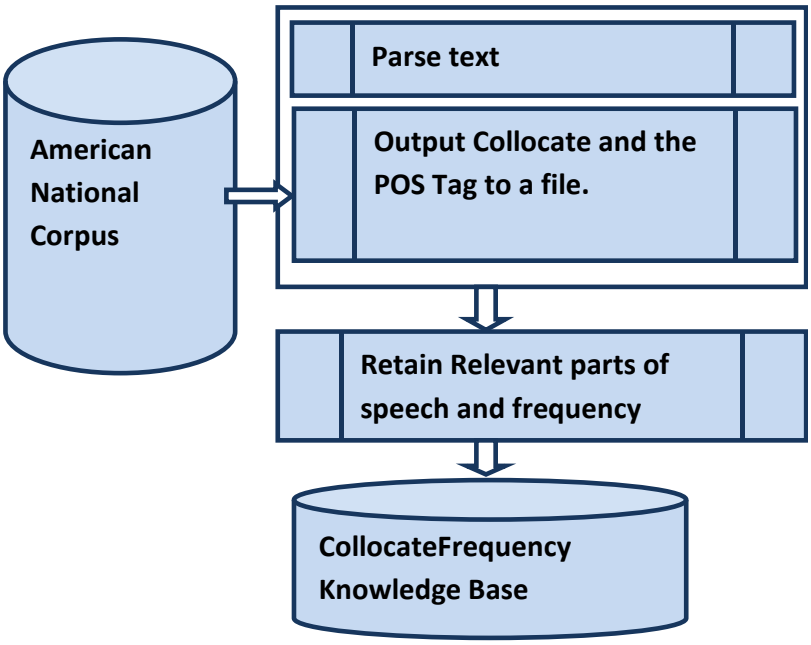


Figure 4: Creating a CFKB using a corpus DB such as ANC

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 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.

4.3 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.

4.3.1 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.

4.3.2 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. Lee, L.[18] has empirically evaluated a number of distributional similarity measures, including the skew divergence, and analyzed their information sources.

4.3.2.1 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 $\frac{Freq(A\ and\ B)/N}{Freq(B)/N}$ where N is the total number of words, while A and B are the two terms in the given expression.

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

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.

4.3.2.2 Jaccard's Coefficient

We use the Jaccard similarity coefficient as a measure of semantic similarity. Jaccard's Coefficient (Salton and McGill 1983)[28], also known as the Tanimoto Coefficient (Resnik 1995)[27], is a popular combinatorial similarity measure. 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 $Freq(A \cap B)$ is calculated by searching through the corpus for the two words with a word window of 32 which we define as *co-occurrence Window*. We refer to the paper Terra et al. [33] to find a suitable window size.

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

Thus, Jaccard's coefficient is calculated using $\frac{Freq(A \cap B)}{Freq(A)+Freq(B)}$

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 terms such as "strong" and "potent" in being conveyed as the suggested responses "strong tea" and "potent tea" respectively.

4.3.2.3 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 $Freq(A \cup B)$ which is the denominator in the Jaccard coefficient.

$$WebJaccard(A,B) = \frac{Freq(A\ and\ B)}{Freq(A) + Freq(B) - Freq(A\ and\ B)}$$

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.

According to Bollegala et. al. [6] this measure is used because due to the scale and noise in Web data, it is possible that two words may appear on some pages purely accidentally. In order to reduce the adverse effects attributable to random co-occurrence the co-occurrence is reduced from the total occurrence.

They also recommend that below a certain threshold the value should be zero.

4.3.2.4 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:

Min = frequency of the lowest occurring collocate

Max = frequency of the highest occurring collocate

For any given collocate:

Frequency Normalized = (Frequency of collocate – Min)/(Max – Min)

This is a very common approach in machine learning technique where we can make use of a normalized value instead of the actual value so that comparisons can be performed and better learning models can be formulated. The method that we apply is called *rescaling* which is a type of *feature scaling*. Refer [15] Juszczak, P., to see how feature scaling is used in data mining.

4.3.2.5 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:

Frequency Ratio = Freq(A and B)/(Freq(A)+Freq(B))

4.3.3 Combining the Measures

Empirically, it has been found that different measures yielded good results in different scenarios. We therefore propose to deploy the data mining 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 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 [4])

Input	Suggestion	Conditional prob	freqNorm	jaccard	freqRatio	Web Jaccard	Class
pure sky	blue sky	0.02040207	0.836207	0.002096	0.013882	0.002100525	y
pure sky	night sky	0.004351685	0.607759	0.000523	0.003834	0.0005233	n
pure sky	clear sky	0.001323547	0.094828	0.00177	0.001113	0.00177349	y
pure sky	pure water	0.001397421	0.159483	0.002457	0.001275	0.002463307	n

Figure 5: Subset of the Training Set for 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 the correct collocate and the ones that are highly likely to be correct choices are marked as y (Yes). This serves as the training set.

Then we use the well-known data mining tool WEKA which allows us to generate the rules that we can use in our program for ranking. The input to the JRIP algorithm in WEKA 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.

((Jaccard >= 0.00338) AND (frequencyRatio >= 0.002833))

OR

((jaccard >= 0.00177) 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 “thoroughly beautiful man”, the terms “thoroughly beautiful” and beautiful man” would both be detected as erroneous 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 ongoing work.

5. 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 earlier.

5.1 Pre-computing Collocates in CollOrder

The algorithm for pre-computing collocates is given below as Algorithm1. 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.

```
for all document ∈ ANC do                          ▷ ANC contains text files
    sentences ⇐ parse(document)
    for all sentence ∈ sentences do
        for all taggedItem ∈ sentence do
            if POSTAG ∈ ('amod','advmod','nn','prt')) then
                storeinCFKB(taggedItem)
            end if
        end for
    end for
end for
```

Algorithm 1: Pre-computing of Collocates in CollOrder

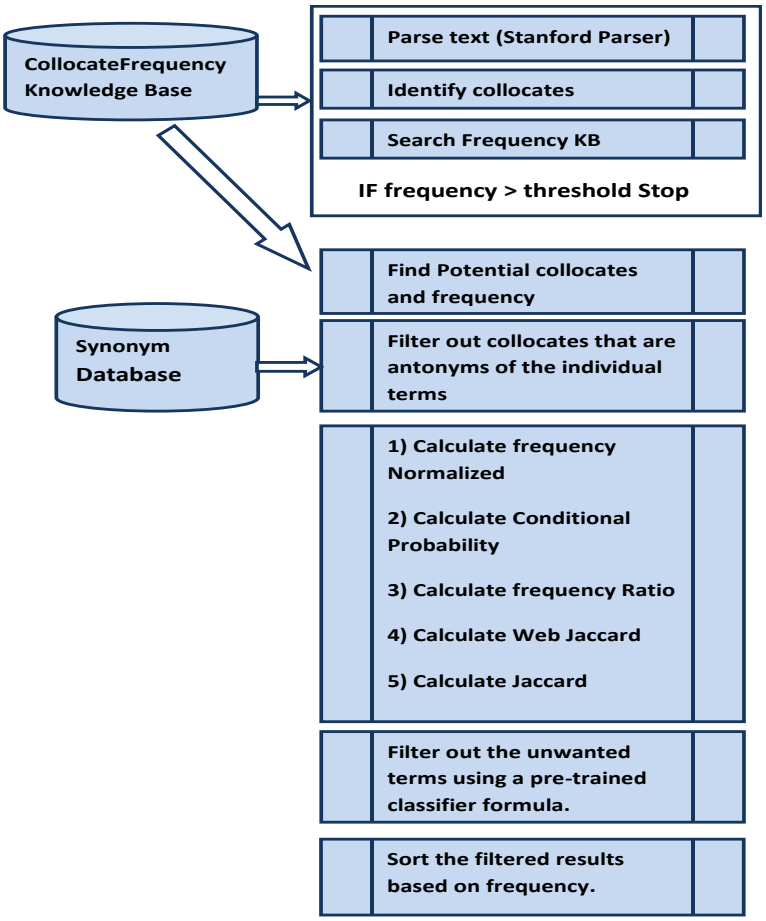


Figure 6: Details of the CollOrder Approach

5.2 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.

```
function COLLORDER(sentence)
    taggedItems ⇐ parse(sentence)
    for all item ∈ taggedItems do
        if posTag ∈ ('amod','advmod','nn','prt') then
            cfkbItem ⇐ findInCFKB(item)                    ▷ Error Detection
                                        ▷ Item should exist and freq > threshold
            if cfkbItem = null then
                ERRORCORRECTION(item)
            end if
        end if
    end for
end function
```

Algorithm 2: CollOrder Main Routine

5.3 Error Correction in CollOrder

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

```
function ERRORCORRECTION(inputCollocate)
    suggestions ⇐ getCollocatesFromCFKB(inputCollocate)
    filterAntonyms(inputCollocate, suggestions)
    markSynonyms(inputCollocate, suggestions)
    CALCULATEMEASURES(inputCollocate, suggestions)
    FILTERBYJRIPRULES(suggestions)
                            ▷ Suggestions that do not match are filtered out
    PRINTALLsuggestions
            ▷ The filtered suggestions are already sorted based on frequency
end function
function CALCULATEMEASURES(inputCollocate,suggestions)
    sortByFrequency(suggestions)                              ▷ Descending
    minFreq ⇐ suggestions[n].frequency                  ▷ n is size of list
    minFreq ⇐ suggestions[0].frequency
    for all collocation ∈ suggestions do
        if inputCollocate.term2 = collocation.term2 then
                                        ▷ input:pure sky, collocation: clear sky
            freq1 ⇐ SEARCHCOOCCUR(inputCollocate.term1, collocation.term1)
            freqA ⇐ SEARCHANC(inputCollocate.term1)
            freqB ⇐ SEARCHANC(collocation.term1)
            freqPart1 ⇐ FreqB
            freqPart2 ⇐ SEARCHANC(collocation.term2)
        else
                                        ▷ input:pure sky, collocation: pure white
            freq1 ⇐ SEARCHCOOCCUR(inputCollocate.term2, collocation.term2)
            freqA ⇐ SEARCHANC(inputCollocate.term2)
            freqB ⇐ SEARCHANC(collocation.term2)
            freqPart1 ⇐ SEARCHANC(collocation.term1)
            freqPart2 ⇐ FreqB
        end if
        collocation.jaccard ⇐ freq1/(freqA + freqB)
        collocation.frequencyRatio ⇐ collocation.frequency/(freqPart1 +
freqPart2)
        collocation.frequencyNorm ⇐ (collocation.frequency−minFreq)/(maxFreq−
minFreq)
        collocation.webjaccard ⇐ freq1/(freqA + freqB − freq1)
        if collocation.webjaccard < 0 then collocation.webjaccard ⇐ 0
        end if
    end for
end function
function FILTERBYJRIPRULES(suggestions)
                            ▷ This function uses the rules created by JRIP
                    ▷ This provides a hueristic based on the combination
                ▷ or Ensemble of the individual measures we calculated
    for all collocation ∈ suggestions do
        if ensembleFunction(collocation) == false then
            suggestions.remove(collocation)
        end if
    end for
end function
```

Algorithm 3: Error Correction in CollOrder

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
- Finally it prints out the top-k results in the order of correctness of collocation based on the ranking

Having discussed these algorithms we developed in CollOrder, we now proceed with the details of our experimental evaluation.

6. System Demonstration

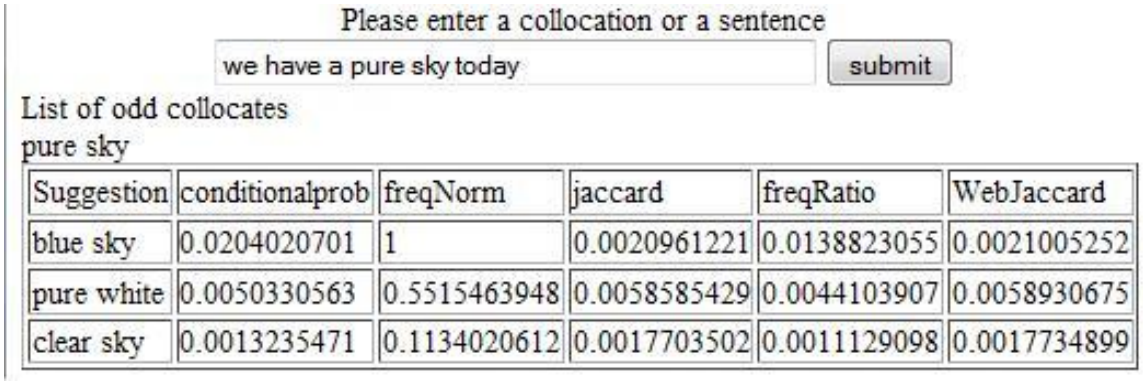
We now present a few screenshots and examples along with explanation to demonstrate the working of the CollOrder system. Please note that only a few examples have been shown here, while many more will be depicted in a live demo.

- **Input:** The CollOrder system has been developed in Java and the core system accepts a string as input. The input can be a sentence or a part of a sentence.

  Examples:

  "We have a pure sky today" or simply "pure sky"; "I need powerful coffee" or just "powerful coffee"; "I want to buy a quick car" or "quick car"

  For evaluating the system we have developed a Web interface to the system and the interface is shown in Figure 7.



Suggestion	conditionalprob	freqNorm	jaccard	freqRatio	WebJaccard
blue sky	0.0204020701	1	0.0020961221	0.0138823055	0.0021005252
pure white	0.0050330563	0.5515463948	0.0058585429	0.0044103907	0.0058930675
clear sky	0.0013235471	0.1134020612	0.0017703502	0.0011129098	0.0017734899

Figure 7: CollOrder Web Interface Screenshot

- **Parsing:** The input is parsed by a PCFG (Probabilistic Context Free Grammar) parser as in [17] to identify the typed dependencies in the input. We have found that the parser works well with inputs that are full sentences and also with just a part of a sentence. We show the processing that occurs in CollOrder pertaining to different parses below. We have considered a few examples of collocations here. If these were sent to CollOrder, the parser would yield the following in a system demonstration:

  Input: I need powerful coffee

  - *Parsing:* [nsubj(need-2, I-1), root(ROOT-0, need-2), det(coffee-5, a-3), amod(coffee-5, powerful-4), dobj(need-2, coffee-5)]

  Input: shut down

  - *Parsing:* [root(ROOT-0, shut-1), prt(shut-1, down-2)]

  Input: We have a clear sky today

  - *Parsing:* [nsubj(have-2, we-1), root(ROOT-0, have-2), det(sky-5, a-3), amod(sky-5, clear-4), dobj(have-2, sky-5), tmod(have-2, today-6)]

  Input: We have a pure sky today

  - *Parsing:* [nsubj(have-2, we-1), root(ROOT-0, have-2), det(sky-5, a-3), amod(sky-5, pure-4), dobj(have-2, sky-5), tmod(have-2, today-6)]

  Input: computer products

  - *Parsing:*[nn(products-2, computer-1), root(ROOT-0, products-2)]

Input: genetically modified

- *Parsing:*[advmod(modified-2, genetically-1), root(ROOT-0, modified-2)]

- **Identify Collocates:** We use the input from the parses to obtain the typed dependencies that we consider as collocates.

  Thus, collocates as identified by the system demo are:

  amod(coffee-5, powerful-4)

  prt(shut-1, down-2)

  amod(sky-5, clear-4)

  amod(sky-5, pure-4)

  nn(products-2, computer-1)

  advmod(modified-2, genetically-1)

- **Check if Collocates are Odd:** For each collocate identified in the input, the CollOrder system finds the frequency of occurrence in the CFKB (Collocate Frequency Knowledge Base) which consists of correct collocations. This is achieved by a SQL query to the MYSQL database that we created to store this CFKB. In the system demo, the frequencies of the "collocates" in the current example would be as shown in Figure 8.

Collocate	**Frequency**
amod(coffee-5, powerful-4)	0
prt(shut-1, down-2)	379
amod(sky-5, clear-4)	33
amod(sky-5, pure-4)	0
nn(products-2, computer-1)	33
advmod(modified-2, genetically-1)	20

Figure 8: Collocates and their Frequency

If the frequency is less than threshold then, the term is flagged as odd and the remaining collocates are flagged as correct collocations. (Please note that the threshold is an experimental parameter and in our experiments we have established the threshold as 10 based on discussion with domain experts). Accordingly, in the given examples, amod(coffee-5, powerful-4) and amod(sky-5, pure-4) are detected as incorrect collocates in the demo.

- **Find Suggestions:** CollOrder substitutes the odd collocation with a correct one. For this, it first finds all the collocations in the CFKB with terms that begin with the first term and those that end with the second term. It deploys SQL queries to fetch this information from a pre-computed knowledge base, the CFKB, for efficiency. In Figure 9, we show only one example on "pure sky" for brevity as would appear in the system demo. More can be shown in a live demo.

blue sky	night sky	clear sky	pure water	gray sky
cloudless sky	pure form	pure gold	dark sky	pure white
pure chance	pure speculation	evening sky	pure pleasure	pure mathematics
whole sky	pure science	pure joy	pure research	black sky
pure delight	eastern sky	pale sky	darkening sky	azure sky
pure wool	summer sky	pure light	northern sky	red sky
starry sky	pure silk	pure alcohol	winter sky	pure cotton
pure fiction	pure genius	pure coincidence	brilliant sky	pure solvent
pure oxygen	pure spirit	pure group	pure fantasy	bright sky
pure system	pure reason	deep sky	pure judgement	pure accid7ent
morning sky	pure entertainment			

Figure 9: List of potential suggestions for Odd Collocates

Further processing in CollOrder for finding suggestions occurs in the following steps:

- *Pre-filter*

  ─ Dependency Type (POS Tag)

  We eliminate collocates that are not types that match our input collocate.

  ─ Antonyms

  We also eliminate words that are antonyms of their counter parts.

  e.g.: if the input is “big range”, then “small range” would be eliminated from the list of suggestions.

- *Sort by frequency*

  We order the list from the most frequent to the least frequent. In the example shown here, the list of potential suggestions is already sorted by frequency.

- *Calculate measures*

  We calculate values of the similarity measures (discussed in the previous section) pertaining to the given example.

  Sample calculations are shown below as would appear in the demo screenshots.

  For the suggestion “blue sky”:

  Jaccard’s Coefficient = 28/(10048+ 3310)

  Frequency Ratio = 205/(10048 + 4719)

  Frequency Normalized = (205-11)/(205-11)

  For the suggestion “night sky”:

  Jaccard’s Coefficient = 20/(34929+ 3310)

  Frequency Ratio = 152/(34929 + 4719)

  Frequency Normalized = (152-11)/(205-11)

Please note that to arrive at the values, it is mandatory to perform a search on the corpus and that slows the system down considerably, so it can take a few minutes to return the results. We overcome this issue by caching the results in our MySQL database after we encounter it for the first time. Thus, the expensive search does not have to be performed the next time a user inputs the same or similar terms. For example, the co-occurrence of "pure" and "blue" = 28 with a co-occurrence window of 30 is used in the calculation of Jaccard's Coefficient for blue sky. This value is cached in the database against the terms "pure" and "blue" so that any future comparisons that require this co-occurrence won't need a full text search.

- **Filter Results:** In this step, the system uses the formula that was derived upon using a classifier to filter out the results. For the given example, this yields:

  ((Jaccard >= 0.00338) AND (frequencyRatio >= 0.002833))
  OR
  ((jaccard >= 0.00177) AND (frequencyNormalized >= 0.811321))

  Accordingly, the demo screenshots of the filtered results are shown in Figure 10.

Suggestion	freqNorm	Jaccard	freqRatio
blue sky	1	0.0020961221	0.0138823055
pure white	0.5515463948	0.0058585429	0.0044103907
clear sky	0.1134020612	0.0017703502	0.0011129098

Figure 10: Filtered Results for Suggestions

- **Output:** Thus, the final output that would be displayed to the user through the user interface in the "pure sky" example would be as shown in Figure 11.

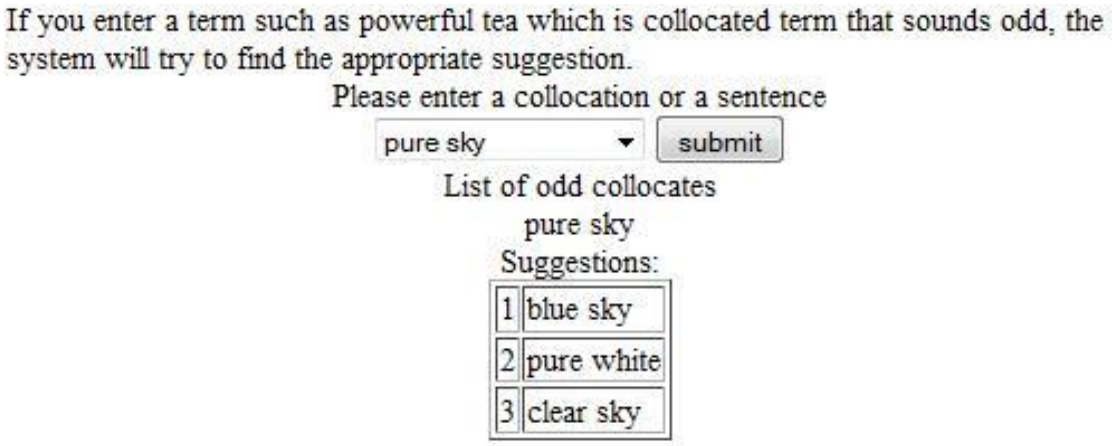


Figure 11: Web Interface Output Suggestions for an Odd Collocation Example

Likewise we have demonstrated the functioning of the CollOrder system along with its user interaction.

7. EVALUATION

We have developed a software tool based on our CollOrder approach. 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 approach for detection and correction of collocation errors, we have conducted user surveys as described next.

7.1 Evaluation using Amazon Mechanical Turk

7.1.1 Mechanical Turk

The Amazon Mechanical Turk (MTurk) is a crowd sourcing Internet marketplace that enables computer programmers (known as Requesters) to co-ordinate the use of human intelligence to perform tasks that computers are currently unable to do. It is one of the suites of Amazon Web Services. The Requesters are able to post tasks known as HITs (Human Intelligence Tasks), such as choosing the best among several photographs of a store-front, writing product descriptions, or identifying performers on music CDs. Workers (called Providers in Mechanical Turk's Terms of Service, or, more colloquially, Turkers) can then browse among existing tasks and complete them for a monetary payment set by the Requester.

To place HITs, the requesting programs use an open Application Programming Interface, or the more limited MTurk Requester site.

MTurk Requester site

The MTurk requester site allows the creation of templates for HIT that will be presented to the evaluators.

Publish HITs to the marketplace

You can load millions of HITs into the marketplace. Each HIT can have multiple assignments so that different Workers can provide answers to the same set of questions and you can compare the results to form an agreed-upon answer.

Workers submit assignments for review

When a Worker completes your HIT, he or she submits an assignment for you to review.

Approve or reject assignments

When your work items have been completed, you can review the results and approve or reject them.

Our Evaluation

We describe how the evaluation was performed in this paragraph.

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 in Figure 12. This Figure shows what a *mechanical turk* worker would see when he/she attempts to work on an HIT.

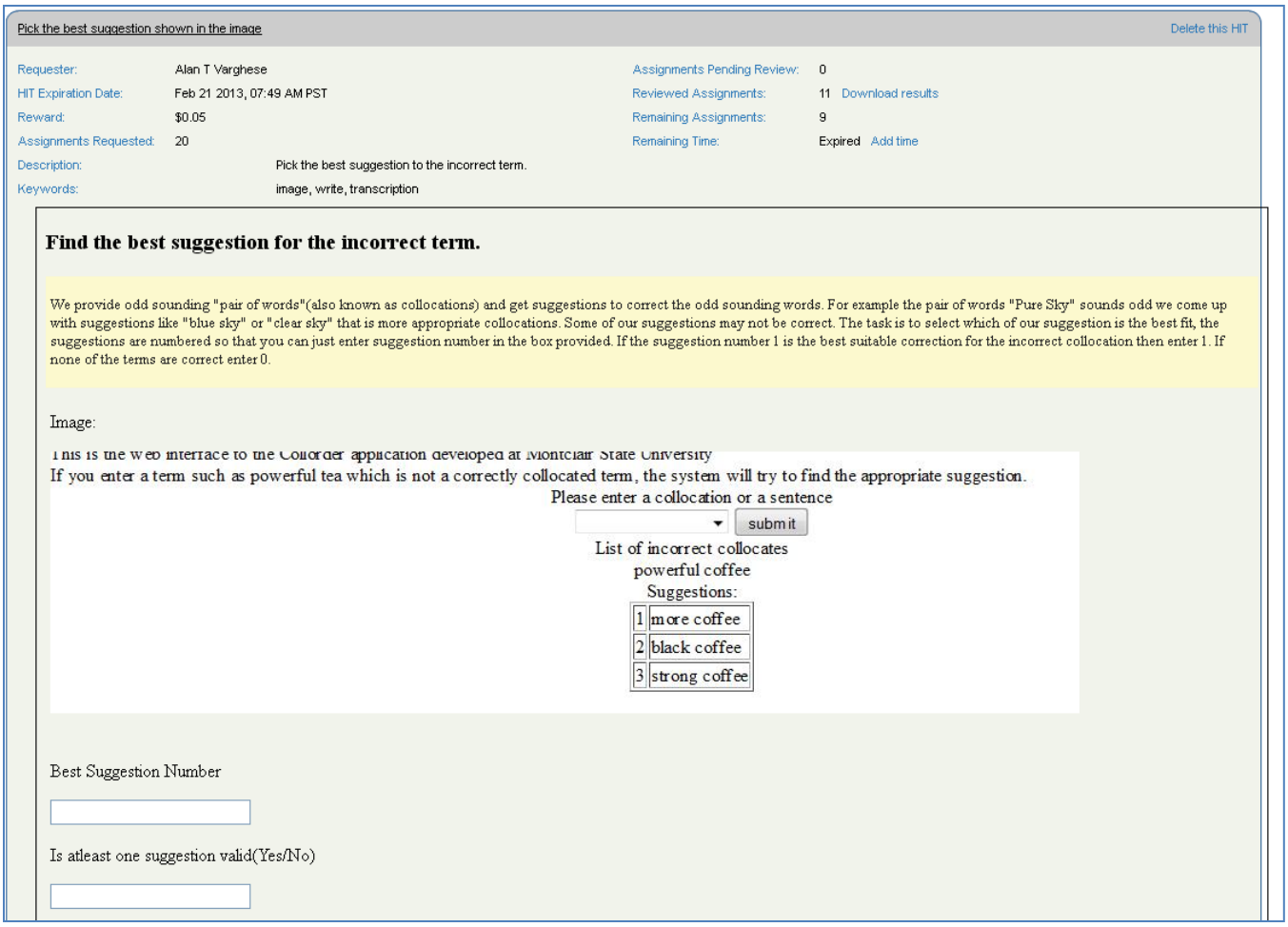


Figure 12: Screenshot of an HIT

Each evaluator was asked to look at the image and answer two questions:

1) The rank of the best suggestion:

   (Based on the image above the answers can be 1, 2 or 3 or 0)0 would mean a “No” to the next answer

2) Was a good suggestion present in the list:

   Yes/No

Amazon Turk allows us to download the results as a csv(comma separated value) file 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.

Contents of the CSV:

HITId	HITTypeId	Title	Description	Keywords	Reward	CreationTime
2WW4EGDHDM253WYZEON6QEZ6OMOCMH	23BKMN9U8P9YLW9MNM931UYIM00DU2	Pick the best suggestion shown in the image	Pick the best suggestion to the incorrect term.	image, write, transcription	$0.05	Thu Feb 14 15:49:04 GMT 2013
2WW4EGDHDM253WYZEON6QEZ6OMOCMH	23BKMN9U8P9YLW9MNM931UYIM00DU2	Pick the best suggestion shown in the image	Pick the best suggestion to the incorrect term.	image, write, transcription	$0.05	Thu Feb 14 15:49:04 GMT 2013
2WW4EGDHDM253WYZEON6QEZ6OMOCMH	23BKMN9U8P9YLW9MNM931UYIM00DU2	Pick the best suggestion shown in the image	Pick the best suggestion to the incorrect term.	image, write, transcription	$0.05	Thu Feb 14 15:49:04 GMT 2013
2WW4EGDHDM253WYZEON6QEZ6OMOCMH	23BKMN9U8P9YLW9MNM931UYIM00DU2	Pick the best suggestion shown in the image	Pick the best suggestion to the incorrect term.	image, write, transcription	$0.05	Thu Feb 14 15:49:04 GMT 2013
2WW4EGDHDM253WYZEON6QEZ6OMOCMH	23BKMN9U8P9YLW9MNM931UYIM00DU2	Pick the best suggestion shown in the image	Pick the best suggestion to the incorrect term.	image, write, transcription	$0.05	Thu Feb 14 15:49:04 GMT 2013
2WW4EGDHDM253WYZEON6QEZ6OMOCMH	23BKMN9U8P9YLW9MNM931UYIM00DU2	Pick the best suggestion shown in the image	Pick the best suggestion to the incorrect term.	image, write, transcription	$0.05	Thu Feb 14 15:49:04 GMT 2013
2WW4EGDHDM253WYZEON6QEZ6OMOCMH	23BKMN9U8P9YLW9MNM931UYIM00DU2	Pick the best suggestion shown in the image	Pick the best suggestion to the incorrect term.	image, write, transcription	$0.05	Thu Feb 14 15:49:04 GMT 2013
2WW4EGDHDM253WYZEON6QEZ6OMOCMH	23BKMN9U8P9YLW9MNM931UYIM00DU2	Pick the best suggestion shown in the image	Pick the best suggestion to the incorrect term.	image, write, transcription	$0.05	Thu Feb 14 15:49:04 GMT 2013
2WW4EGDHDM253WYZEON6QEZ6OMOCMH	23BKMN9U8P9YLW9MNM931UYIM00DU2	Pick the best suggestion shown in the image	Pick the best suggestion to the incorrect term.	image, write, transcription	$0.05	Thu Feb 14 15:49:04 GMT 2013
2WW4EGDHDM253WYZEON6QEZ6OMOCMH	23BKMN9U8P9YLW9MNM931UYIM00DU2	Pick the best suggestion shown in the image	Pick the best suggestion to the incorrect term.	image, write, transcription	$0.05	Thu Feb 14 15:49:04 GMT 2013

MaxAssignments	RequesterAnnotation	AssignmentDurationInSeconds	AutoApprovalDelayInSeconds	Expiration	NumberOfSimilarHITs	LifetimeInSeconds	AssignmentId	WorkerId
20	BatchId:1037651;	1800	259200	Thu Feb 21 15:49:04 GMT 2013			21B85OZIZEHMSY5CFDH5NWPXLFPZBD	A1VKHE5L6QVFUV
20	BatchId:1037651;	1800	259200	Thu Feb 21 15:49:04 GMT 2013			27EB3Q39Z0B6CH4WKT5V1TVRK1PKVL	A3QIEF1HBF69Y4
20	BatchId:1037651;	1800	259200	Thu Feb 21 15:49:04 GMT 2013			27UQ45UUKQOYYBYMSCJ8XG010SRDV0	A2WNW8A4MOR7T7
20	BatchId:1037651;	1800	259200	Thu Feb 21 15:49:04 GMT 2013			28U4E6AUBXHWA5N5CY0VKZWDR30X1V	A1G05O3HM7DNVZ
20	BatchId:1037651;	1800	259200	Thu Feb 21 15:49:04 GMT 2013			2CBE1M7PKK9L9KMEQZ30DLZXP6NZSW	A1E6RS45GUAFC3
20	BatchId:1037651;	1800	259200	Thu Feb 21 15:49:04 GMT 2013			2CBENLVWJ5OPTOEL7OZYVWB7E7E1TI	A21SF3IKIZB0VN
20	BatchId:1037651;	1800	259200	Thu Feb 21 15:49:04 GMT 2013			2H5ZKPFE4EGDZ12SWAZIINZW6ZI7HA	A2LCFORIW0NF1S
20	BatchId:1037651;	1800	259200	Thu Feb 21 15:49:04 GMT 2013			2TVNAB6BFMFF6ME8KQKJODGESU6VIL	ARX0S1CIDJLOX
20	BatchId:1037651;	1800	259200	Thu Feb 21 15:49:04 GMT 2013			2U4CVLL3CA3345QFHT39WQU7XI99Z6	A3R467YDQ1C8VG
20	BatchId:1037651;	1800	259200	Thu Feb 21 15:49:04 GMT 2013			2UKSZVXX2Q45CI0GFN7NRW1NSQS7PF	A2LO0Y6U9D4PRV

AssignmentStatus	AcceptTime	SubmitTime	AutoApprovalTime	ApprovalTime	RejectionTime	RequesterFeedback	WorkTimeInSeconds	LifetimeApprovalRate
Approved	Thu Feb 14 15:57:12 GMT 2013	Thu Feb 14 15:57:21 GMT 2013	Sun Feb 17 07:57:21 PST 2013	Thu Feb 14 14:17:17 PST 2013			9	100% (16/16)
Approved	Sat Feb 16 22:03:28 GMT 2013	Sat Feb 16 22:04:54 GMT 2013	Tue Feb 19 14:04:54 PST 2013	Sat Feb 16 17:43:29 PST 2013			86	100% (20/20)
Approved	Tue Feb 19 11:03:09 GMT 2013	Tue Feb 19 11:03:24 GMT 2013	Fri Feb 22 03:03:24 PST 2013	Tue Feb 19 13:06:20 PST 2013			15	100% (20/20)
Approved	Thu Feb 14 15:54:31 GMT 2013	Thu Feb 14 15:54:48 GMT 2013	Sun Feb 17 07:54:48 PST 2013	Thu Feb 14 14:17:14 PST 2013			17	100% (20/20)
Approved	Mon Feb 18 14:01:22 GMT 2013	Mon Feb 18 14:01:34 GMT 2013	Thu Feb 21 06:01:34 PST 2013	Mon Feb 18 06:23:33 PST 2013			12	100% (20/20)
Approved	Thu Feb 14 17:02:36 GMT 2013	Thu Feb 14 17:02:56 GMT 2013	Sun Feb 17 09:02:56 PST 2013	Thu Feb 14 14:17:15 PST 2013			20	100% (20/20)
Approved	Thu Feb 14 15:55:16 GMT 2013	Thu Feb 14 15:55:43 GMT 2013	Sun Feb 17 07:55:43 PST 2013	Thu Feb 14 14:17:17 PST 2013			27	100% (12/12)
Approved	Thu Feb 14 19:26:25 GMT 2013	Thu Feb 14 19:26:38 GMT 2013	Sun Feb 17 11:26:38 PST 2013	Thu Feb 14 14:17:11 PST 2013			13	100% (20/20)
Approved	Mon Feb 18 22:15:21 GMT 2013	Mon Feb 18 22:15:36 GMT 2013	Thu Feb 21 14:15:36 PST 2013	Tue Feb 19 13:06:17 PST 2013			15	100% (19/19)
Approved	Thu Feb 14 16:10:24 GMT 2013	Thu Feb 14 16:10:29 GMT 2013	Sun Feb 17 08:10:29 PST 2013	Thu Feb 14 14:17:13 PST 2013			5	100% (15/15)

Last30DaysApprovalRate	Last7DaysApprovalRate	Input.image_url	Answer.Tag1	Answer.Tag2	Approve	Reject
100% (16/16)	0% (0/0)	http://ec2-184-72-209-105.compute-1.amazonaws.com:8080/CollorderWeb/01puresky.JPG	3	Yes		
100% (20/20)	0% (0/0)	http://ec2-184-72-209-105.compute-1.amazonaws.com:8080/CollorderWeb/01puresky.JPG	3	Yes		
100% (20/20)	0% (0/0)	http://ec2-184-72-209-105.compute-1.amazonaws.com:8080/CollorderWeb/01puresky.JPG	3	Yes		
100% (20/20)	0% (0/0)	http://ec2-184-72-209-105.compute-1.amazonaws.com:8080/CollorderWeb/01puresky.JPG	3	Yes		
100% (20/20)	0% (0/0)	http://ec2-184-72-209-105.compute-1.amazonaws.com:8080/CollorderWeb/01puresky.JPG	3	Yes		
100% (20/20)	0% (0/0)	http://ec2-184-72-209-105.compute-1.amazonaws.com:8080/CollorderWeb/01puresky.JPG	1	yes		
100% (12/12)	0% (0/0)	http://ec2-184-72-209-105.compute-1.amazonaws.com:8080/CollorderWeb/01puresky.JPG	3	Yes		
100% (20/20)	0% (0/0)	http://ec2-184-72-209-105.compute-1.amazonaws.com:8080/CollorderWeb/01puresky.JPG	3	yes		
100% (19/19)	0% (0/0)	http://ec2-184-72-209-105.compute-1.amazonaws.com:8080/CollorderWeb/01puresky.JPG	3	yes		
100% (15/15)	0% (0/0)	http://ec2-184-72-209-105.compute-1.amazonaws.com:8080/CollorderWeb/01puresky.JPG	3	Yes		

Figure 13: Screen shot of the csv returned by mechanical turk

Note* The contents have been broken down into pieces because the entire width does not fit on the page:

We have explained each of the columns in the Table below:

Table 1: Explanation of Terms in the CSV file used in evaluation

HITID	This is the ID of the HIT
HITTypeID	This is an ID that is used by *mechanical turk* to identify the type of this HIT.
Title	This is the title that we give to our HIT when we create the HIT.
Description	This is a brief description of our HIT which we have to provide at the time of creation.
Keywords	These are the keywords we provide so that people can find our HIT.
Reward	This is the reward for completing the HIT.
Creation Time	The date and time when the HIT was created.
Max Assignments	This is the total number of assignments. We had 20 HITs.
Assignment Duration in Secs	This is the time allotted for completion of the HIT.
Autoapproval Delay in Secs	This is the delay after which the HIT will be automatically approved. Even if the requester does not approve it.
Expiration	The date and time at which the HIT expire and no longer be available to work on.
Number of similar HITS	This is the number of HITs that are similar.
Lifetimeinsecs	The number of seconds that the HIT was available.
AssignmentID	This is a unique ID for the specific assignment of an HIT.
WorkerID	This is the ID of the worker that completed the HIT.
AssignmentStatus	This is the status of the assignment,

	Completed/Approved/Rejected.
AcceptTime	This is the date and time at which the assignment was accepted.
SubmitTime	This is the time at which the assignment was submitted
AutoApprovalTime	This is the date and time at which the submission will be automatically approved
ApprovalTime	This is the date and time at which the task was approved.
Rejectiontime	This is the date and time at which the task was rejected.
RequesterFeedback	The feedback we provide to the worker will be shown here.
Worktimeinseconds	The time the worker spend on the HIT.
Lifetimeapprovalrate	The approval rate of the HIT.
Lst30DaysApprovalRate	The Approval rate of the HIT over a period of 30 days.
Input.image_url	URL of the image that we provided: The image appears as shown in Figure 10
Answer.Tag1	Contains the number of the suggestion that is the best choice.
Answer.Tag2	Indicates if the results shown have at least one suggestion that is valid.

The result CSV file contains a lot of accounting details which are not relevant to the evaluation. The ones that we need are:

Input.image_url: The link of the image that we provided: The image appears as shown in Figure 10

Answer.Tag1: Contains the number of the suggestion that is the best choice.

Answer.Tag2: Indicates if the results shown have at least one suggestion that is valid.

We allowed 1 week for the evaluation and we obtained 225 valid responses in total, so around 10 people evaluated each image.

7.2 User Surveys

We performed a survey on 20 items. We got a total of 225 valid responses. Here is the summary of the evaluation.

Out of the 225 reviews done 208 reviews indicate that the at least one of the suggestions provided was correct. Converting this to a percentage we find that 92.44% of the time the users agreed that a correct alternative to the odd sounding collocate was suggested.

This clearly indicates that the Oddness correction step is very effective.

Out of 225 reviews 95 reviews indicated that the 1st suggestion the correct one, 43 indicated that the 2nd suggestion was the best one 36 reviews indicated that the 3rd one was the correct one.

We find that 77.33% of the time the correct suggestion is in the top 3 suggestions provided by the ranking mechanism.

This clearly indicates that our ranking is also effective.

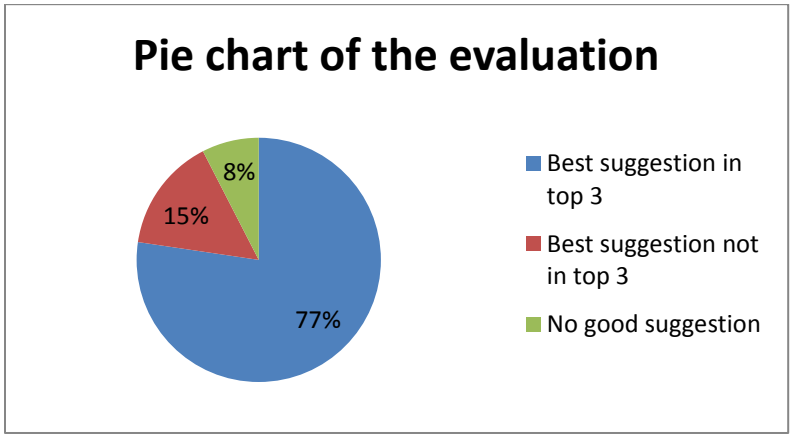


Figure 14: Pie chart of the evaluation

7.3 Detecting Correct Collocates

One of the other measures of the application is to be able to detect the correct collocates. We have run an evaluation on the DISCO 2011(Distributional Semantics and Compositionality) word list. The results are tabulated below in Table 2. The "Error list" column shows the ones that our application incorrectly identified as odd collocates. The "Correct List" column shows the ones that our application correctly identified as collocates.

We identified 81 correctly we flagged 20 as errors.

Table 2: Selected words from DISCO 2011

Error List:	Correct List:
increase understanding	social capital
story start	smart card
future lie	family move
digital radio	central heating
address issue	indigenous people
evidence show	new market
business need	little girl
industry need	poor man
government need	open fire
company need	religious belief
world need	small island
support people	full training
increase knowledge	hard copy
broken link	black hole
economist call	small print
online casino	new window

country need	common sense
spread word	homeless people
foot bill	broad range
word mean	red tape
	double number
	high mountain
	big issue
	holistic approach
	heavy rain
	international airport
	central bank
	wide range
	left hand
	single point
	wet weather
	offer support
	red wine
	big picture
	third party
	early stage
	second hand
	flat tyre
	system work
	coastal path
	unlikely event

	large file
	key player
	high quality
	high street
	high dose
	middle age
	serial port
	dark side
	civil war
	first class
	old testament
	mobile phone
	hot topic
	focus attention
	heavy metal
	system use
	equal opportunity
	statistical analysis
	open access
	social security
	early version
	sunny day
	community need
	leading edge
	offer advice

	secondary care
	unfair dismissal
	open day
	written permission
	panoramic view
	hot spot
	short film
	inner city
	positive response
	fine art
	cheap price
	big fish
	high speed
	rechargeable battery
	popular culture

7.4 Evaluating the response time of the implementation

We evaluated the performance of our implementation and we have come up with an approach to make the system useful in real time systems that require quick responses.

When we search for collocates we internally make HTTP calls to a corpus search engine and each call can take up to 2 seconds. So depending on the number of potential suggestions we find in our CFKB we have to make multiple calls to the website. Hence the wait time for response can be up to a few minutes.

We have implemented a cache table in our MySQL database to cache intermediate results so that we can save corpus search and calculation. So we look in the cache first before we go out to the website so if the user searched for a collocate and it takes about 3-5 minutes to return the value, then next time someone

searches for that term it will take only milliseconds to respond back. We feel that this would be acceptable in search engines. This is somewhat similar to the memorization techniques used in computer science.

7.5 Discussion on Further Challenges

Domain Knowledge: A significant challenge includes capturing domain-specific knowledge with respect to several text corpora, e.g., 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 define a notion of correctness that incorporates aspects other than frequency.

More than two words: The current *implementation* only addresses two word collocations. The approach that we have provided can be extended to handle incorrect collocations with more than two words like thoroughly beautiful man.

We propose to deal with these and other challenges as future work which would further enhance the performance of CollOrder.

8. Applications of CollOrder

We consider some of the applications of CollOrder.

- **Web Search**: A lot of non-native speakers of English search for information using search engines like Google, Yahoo etc. We show the effectiveness of CollOrder in web search in section 7.1.
- **Machine Translation**: This is an area in which literal translations of words can lead to laughable phrases (according to Lin, D [19]). There is work done by Brants, T. et al. [7] in the use of language models in machine translation. We have not evaluated the effectiveness but have shown how we can apply CollOrder in the context of machine translation.
- **Potential use in ESL Writing Aids**: Since this is used by non-native speakers of English there is a potential for collocation errors.

In the sections below we have provided a description of how we can apply CollOrder in each of the above areas.

8.1 Effectiveness of CollOrder in Web Search

Let us assess the usefulness of CollOrder with respect to Web search. Consider a user evaluation example where a non-native speaker *wants to search for "fast cars"* but enters an expression such as "quick car" on a search engine. The response given by Google for this expression is shown in Figure 15. We have 1,270,000 results which do not seem very relevant to the user's really intended query.

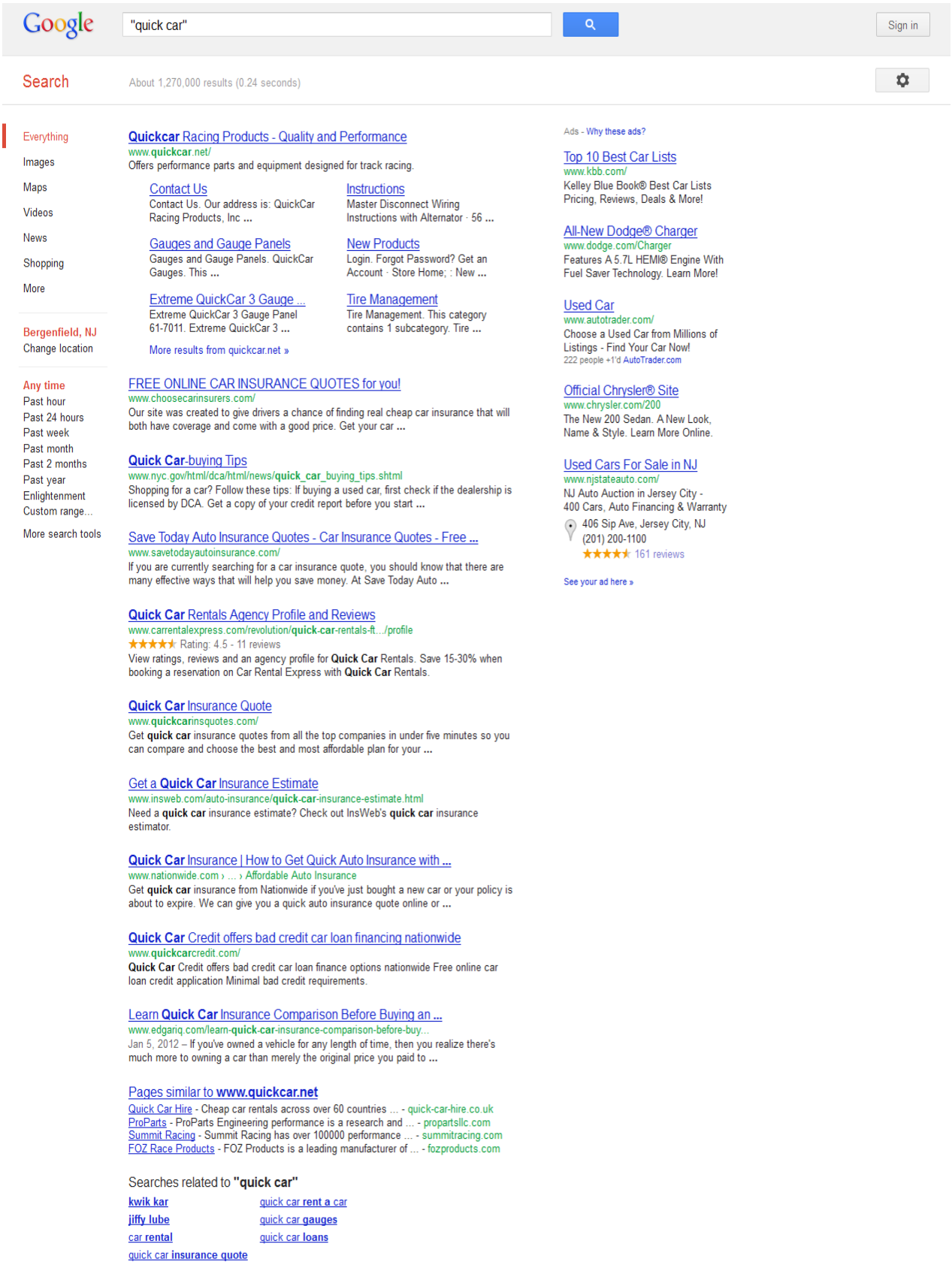


Figure 15: Web Search with user expression “quick car”

Consider the response given by CollOrder in this situation with the user expression “quick car”

Suggestions:

1	Fast Car

We notice that CollOrder outputs only one suggestion here 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. We notice the following search results as shown in Figure 16. We now have 10,800,000 results and the results are much more relevant. We even get images for fast cars. The Google suggestions are also more relevant to the intended search.

Likewise, we find that the suggestions given by CollOrder can indeed improve the effectiveness of Web query results. Similar arguments can be applied in other contexts such as ESL tools for text documents and automated machine translation.

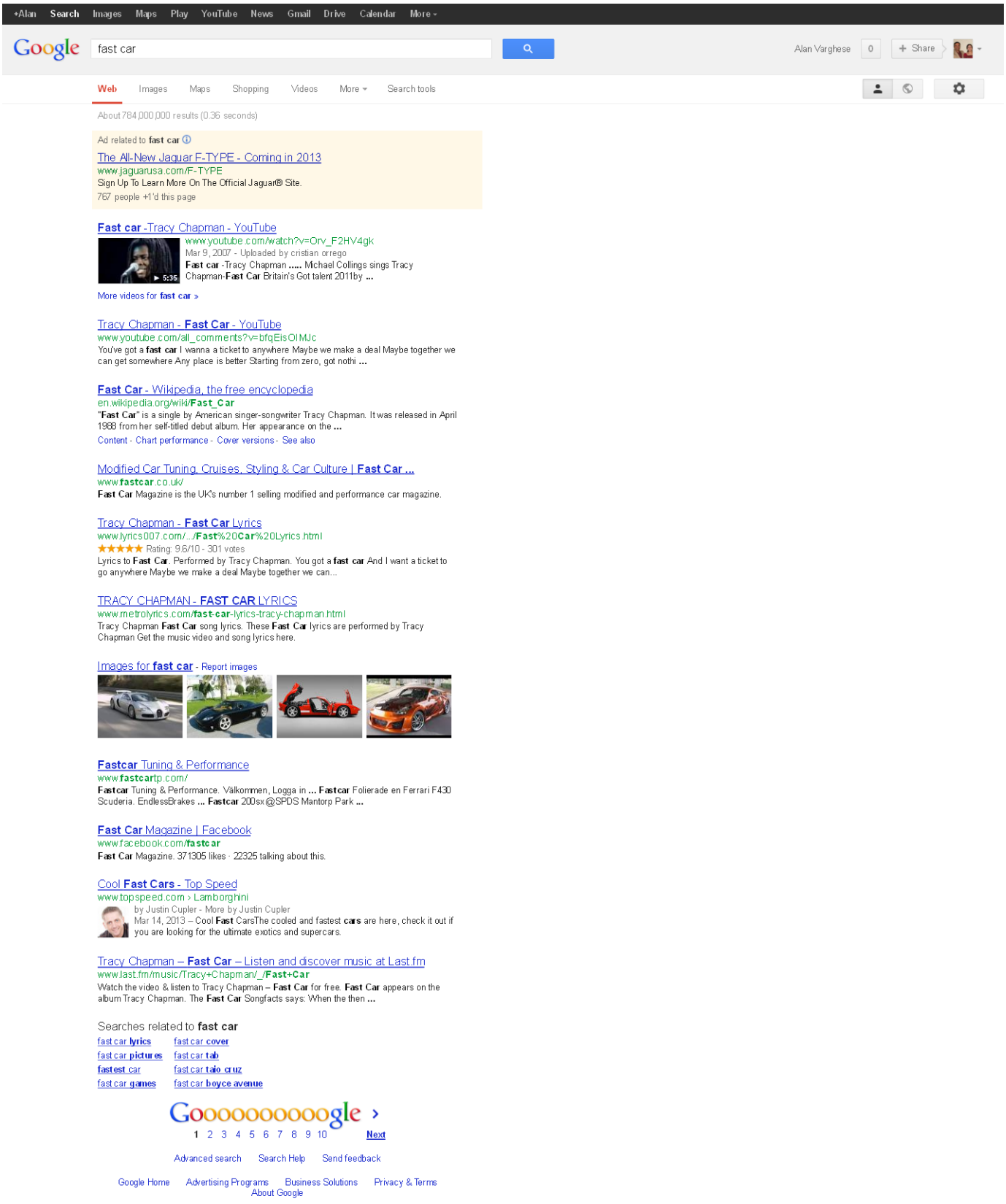


Figure 16: Web Search with CollOrder suggestion "fast car"

8.2 Potential use in Machine Translation

We have presented a description of how CollOrder can assist in machine translation. This is just one possible scenario and we have not done a formal implementation or evaluation of this.

- Machine Translation Software converts a sentence from foreign language to English.
- Send the sentence into CollOrder.
    - CollOrder returns if the collocations in the sentence are correct or if incorrect provides suggestions.
- Machine translation software decides if the sentence needs to be changed
    - Changes the collocations to suggested collocation.
    - Flags it for manual review
    - No change required for correct collocate

8.3 Potential use in ESL Writing Aids

We have presented a description of how CollOrder can assist in evaluating text written by nonnative speakers. This is just one possible scenario and we have not done a formal implementation or evaluation of this.

- A program <evaluator> reads the text and breaks it into sentences.
- Send the sentence into CollOrder.
    - CollOrder returns if the collocations in the sentence are correct or if incorrect provides suggestions.
- The program<evaluator> decides if the sentence needs to be changed
    - Changes the collocations to suggested collocation.
    - Flags it for manual review with potential suggestions

9. CONCLUSIONS AND FUTURE WORK

In this work, we have proposed the CollOrder approach to detect and correct collocation errors in L2 written English. This is useful in the context of Web queries, text documents with ESL help and automated machine translation. The main contributions of this research include:

- Proposing the overall CollOrder approach based on an integrated framework of natural language processing, searching and ranking for detecting and correcting collocation errors in L2 written English
- 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 approach 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, information retrieval and data mining professionals due to emphasis on aspects such as search and ranking, natural language processing, Web and text data, similarity measures and classification. It would also appeal to the computational linguistics community.

Future work includes addressing the issue of sparse but correct data, improving the performance of the system by using powerful computers and in-network synonym and antonym dictionary. We could also host the search engine for the ANC /BNC in-house on a powerful server.

10. REFERENCES

[1] Bannard, C and Callison-Burch, C. 2005. Paraphrasing with bilingual parallel corpora. In Proceedings of ACL(2005): 597-604.

[2] Buttcher, S., and Clarke, C. Indexing time vs. query time: trade-offs in dynamic information retrieval systems. Proceedings of the 14th ACM international conference on Information and knowledge management (2005):317–318.

[3] Choueka, Y. 1988. Looking for needles in a haystack or locating interesting collocational expressions in large textual databases. In Proceedings of the RIA O Conference on User-Oriented Content-Based Text and Image Handling, Cambridge, MA, (1988): 21-24.

[4] Cohen,W.W. 1995. Fast effective rule induction. In *Proc. of the 12th Intl. Conf. on Machine Learning*, 115–123.

[5] Cook, V. J., J. Long, and McDonough, S. 1979. "First and second language learning." The Mother Tongue and Other Languages in Education, CILTR (1979): 7-22.

[6] Bollegala, D., Matsuo,Y., and Ishizuka,M . 2009. Measuring the similarity between implicit semantic relations using web search engines. *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, Ricardo Baeza-Yates, Paolo Boldi, Berthier Ribeiro-Neto, and B. Barla Cambazoglu (Eds.), ACM. 104-113.

[7] Brants, T, et al. 2007. Large language models in machine translation. *In EMNLP* (2007) 858-867

[8] Dahlmeier, D., and Tou, H. N. 2011. *Correcting semantic collocation errors with L1-induced paraphrases*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing,107–117.

[9] Deane, P., and Higgins, D. 2007. Using Singular-Value Decomposition on local word contexts to derive a measure of constructional similarity, in *Fitzpatrick, E. (editor). Corpus Linguistics Beyond the Word: Corpus Research from Phrase to Discourse*. Rodopi, 43-58.

[10] Farghal, M, and Obiedat, H. 1995. Collocations: A neglected variable in EFL. International Review of Applied Linguistics, 33.

[11] Firth, J. R. 1957. *Papers in Linguistics 1934-1951*. Oxford University Press, London.

[12] Futagi, Y., Deane, P., Chodorow, M., and Tetreault, J. 2008. A computational approach to detecting collocation errors in the writing of non-native speakers of English. *Computer Assisted Language Learning*, 21(4): 353-367.

[13] Hindle, D. 1990. Noun classification from predicate argument structures. In Proceedings of ACL-90(1990): 268-275.

[14] Hou, U. L, Mamoulis, N., Berberich, K., and Bedathur, S. 2010. Durable top-k search in document archives. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (SIGMOD), ACM* 555-566.

[15] Juszczak, P., Tax, D. M. J., and Dui, R. P. W. 2002. Feature scaling in support vector data descriptions. *Proc. 8th Annu. Conf. Adv. School Comput. Imaging*: 95–10.

[16] Kalurachchi, A. C, Roychoudhury, S., Varde, A. S., and Weikum, G. 2011. SITAC: discovering semantically identical temporally altering concepts in text archives. *EDBT*, 566-569.

[17] Klein, D., and Manning, C. D. 2003. Accurate Unlexicalized Parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 423-430.

[18] Lee, L. 1999. Measures of Distributional Similarity. *Proceedings of ACL-99*: 25-32

[19] Lin, D. 1999. Automatic Identification of Non-compositional Phrases, *Proceedings of ACL-99:*317-324.

[20] Lin,D. 1998. Automatic retrieval and clustering of similar words. In Proceedings of COLING/ACL-98(1998):768-774.

[21] Liu, L. A., Wible, D., Tsao, N-L., 2009. Automated suggestions for miscollocations, *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, 47-50.

[22] Marneffe, M., MacCartney, B., Manning, C. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. *LREC*, (6):449-454.

[23] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D. and Miller, K. 1990. WordNet: An online lexical database: *International Journal of Lexicography*, 3(4): 235-244.

[24] Park, T., Lank, E., Poupart, P., and Terry, M.. 2008. Is the sky pure today? AwkChecker: an assistive tool for detecting and correcting collocation errors. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. ACM, 121-130.

[25] Pradhan, A. M., Varde, A. S., Peng, J. and Fitzpatrick, E. M. 2010. Automatic Classification of Article Errors in L2 Written English. *AAAI's FLAIRS Conference,* 259-264.

[26] Ramos, M. A., et al. 2010. Towards a Motivated Annotation Schema of Collocation Errors in Learner Corpora, *LREC*, 3209-3214.

[27] Resnik P. 1995. Using information content to evaluate semantic similarity in a taxonomy. In Proceedings of IJCAI-95, pages 448-453.

[28] Salton, G and McGill, M. J. 1983. Introduction to Modern Information Retrieval. McGraw-Hill.

[29] Smadja, F. 1993. Retrieving collocations from text: Xtract. Computational Linguistics, 19(1)(1993):143-178.

[30] Soderland, S. and Lehnert, W. 1994. Corpus-Driven Knowledge Acquisition for Discourse Analysis, *AAAI*, 827-832.

[31] Suchanek, F. M., Kasneci, G., and Weikum, G.. 2008. YAGO: A Large Ontology from Wikipedia and WordNet. *Web Semantics*, 6(3): 203-217.

[32] Swan, M and Smith, B. 2001. Learner English: A Teacher's Guide to Interference and Other Problems. Cambridge University Press, Cambridge, UK(2001).

[33] Terra, E. and Clarke, C. L. A. 2003. Frequency estimates for statistical word similarity measures. *HLT/NAACL*, 244–251.

[34] Turney, P,D., and Pantel, P. "From frequency to meaning: Vector space models of semantics." *Journal of artificial intelligence research 37.1* (2010): 141-188.

[35] Weeds, J, and Weir, D. 2005. Co-occurrence Retrieval: a General Framework for Lexical Distributional Similarity. *Computational Linguistics 31(4)*: 439-476.