Automatic Correction of Idiomatic Usage in English Using Web Search

Ting Qian
Department of Brain and Cognitive Sciences
University of Rochester
Rochester, NY 14627, USA
tqian@mail.rochester.edu

Lin Qiu
Department of Computer Science
State University of New York at Oswego
109 Snygg Hall, Oswego, NY 13126, USA
lqiu@cs.oswego.edu

Abstract

Non-native English speakers often have problems determining the exact form of an idiomatic expression while they have some vague idea about the key words in them. In this paper, we describe a system called Webtionary that allows users to consult idiomatic usage by entering a questionable expression. Webtionary uses web search to find candidate corrections and suggests expressions that are commonly used in writing and semantically-related to the user query. Evaluation results show that Webtionary significantly outperforms direct web search in providing useful suggestions.

1. Introduction

Due to the widespread use of personal webpages, blogs and wikis, the web has become an enormous collection of writings created by people all over the world. Recent studies have shown that the web has the same properties as those of a traditional corpus [4].

In this paper, we describe a system that uses the web as a language source to help non-native English speakers with idiomatic usage. Idiomatic usage involves the use of specific particles (a category of words including "the articles, most prepositions and conjunctions, and some interjections and adverbs" [5]) with nouns, verbs, or adjectives. From our own experiences and the user study described below, non-native English speakers often have problems determining the exact form of an idiomatic expression, while they have some vague ideas about the key words that they want to use. For example, a non-native speaker may remember there is an expression about getting in a car that uses the word "hop," but he or she does not know whether it is "hop on a car," "hop in a car," or "hop a car." As there are no grammatical rules about how to use them in general, it is very difficult for non-native speakers to correctly memorize idiomatic expressions that are perfectly natural to native speakers.

While traditional dictionaries can help users with word definitions and examples, it is impossible for a dictionary to list the idiomatic usage of a word in all possible contexts. If a non-native speaker is unsure of which preposition he or she should use in the above example, and looks up "hop" in Oxford Advanced Learner's Dictionary (one of the most comprehensive and popular dictionaries used by English learners), he or she will find "*hop. 4. [VN] (AmE) to get on a plane, bus, etc*" as the most relevant information. However, this offers little help to the user.

To solve the problem, we developed Webtionary, a tool that provides a natural and flexible interface for users to consult idiomatic usage in English. Our premise is that the correct usage of an expression should appear way more often than the incorrect ones on the Web. Even though the web may contain many incorrect writings, its large amount of data should outweigh its noisiness [4]. In the remaining of this paper, we will briefly overview related work and describe the system in detail.

2. Related Work

A number of applications have been built using the web as a corpus to solve linguistic problems [4]. For example, Cilibrasi and Vitanyi [1] created a method that determines how closely two words are semantically related to each other based on the frequency of their appearance on the same webpage. Shamma et. al. [6] show that a word's web frequency highly correlates with people's familiarity with that word.

However, finding the right usage of an expression on the web requires advanced search skills from forming the right queries to analyzing search results. GoogleFight [2] is a tool that allows users to enter several expressions and rank them based on their web frequencies (i.e. the number returned by a search engine that indicates how many webpages contain the search query). While GoogleFight helps users determine which expression is more popular on the web, it requires the user to enter all

0-7695-3026-5/07 $25.00 © 2007 IEEE
DOI 10.1109/WI.2007.142

possible forms of the expression to find the most popular one. This approach is not only tedious, but also impractical because non-native speakers often miss the correct form (e.g., "hop in a car") in their candidate list. Kiwi [7] allows users to use patterns to perform linguistically useful searches. Users can use wildcards to find popular phrases on the web that match a specific pattern. Such tools can be used to consult idiomatic usage if a wildcard is placed to indicate the questionable part. However, this requires the user to understand how to form patterns correctly. This is very difficult for non-native speakers because they often only know some key words in an expression but not the exact structure.

3. Webtionary

Our system, Webtionary, does not require an enumeration of possible expressions or the use of patterns. For example, users can simply enter "hop car" and get the correct expression "hop in a car." Webtionary provides a web-based interface where the user can enter an expression in question and receive corrections with examples showing how each one is used in sentences.

User query	Correction
help English writing	help with English writing
fed by with the noise	fed up with the noise
look forward to see you	look forward to seeing you

Figure 1. Corrections generated by Webtionary for user queries.

User queries entered into Webtionary are sent to Google. Upon receiving the search results, Webtionary extracts candidate corrections from the search results and use the two metrics to rank the candidates. Figure 1 shows examples of corrections generated by Webtionary for real user queries collected during our evaluation study. In the following, we describe the methods that we use to extract candidate corrections and rank them, an evaluation study, and future research directions.

3.1. Candidate correction generation

When a user query is sent to Google, Google returns a list of snippets. The keywords of the user query are formatted in bold by the "" and "" HTML tags. As a result, we extract candidate corrections that start with the first word of the user query, in bold by the HTML tag, and end with the last word of the user query, which must be in bold as well, as shown in Figure 2.

Because Google automatically removes stop words such as *a, the, as, on* from a search query, the snippets returned from Google will contain expressions that have

Snippet	Candidate
...Deputy Judge **Advocate**for **War**crimes, European command ...	Advocate for War
...office of the Judge **Advocate**General (**War**)	Advocate general war

Figure 2. Candidate corrections extracted from snippets returned from Google.

the "important" words in the user query but with different particles. The query normalization techniques employed by Google help us find exactly what we need: expressions that are similar to the user query but not exactly the same.

The more snippets that we retrieve from Google, the more candidate corrections we can get. However, it will increase the response time as well as the chance of finding the right correction. To balance the speed and the quality of corrections, we experimented with 50 and 80 snippets for a user query, and found that 80 snippets produced better results (see the Evaluation section for further discussion).

3.2. Ranking Algorithm

We rank candidate corrections using the product of two metrics. The semantic similarity metric assesses how semantically close a candidate correction is to the user query. The probability metric assesses the likelihood of a candidate correction being the one that the user wants to write. The combination of these two metrics allows us to find the expressions that are not only popular but also express the meaning as close to the user query as possible.

Semantic similarity metric. We use the Vector Space model and the $tf \cdot idf$ method [3] to compute this metric. We first construct a term index table that includes all the words in the user query and candidate corrections. The table is constructed dynamically every time a user enters a query. Then, we represent the user query q and a candidate correction d_i as vectors:

$$\vec{q} = (w_{1,q}, w_{2,q}, w_{3,q}, \ldots, w_{n,q})$$
$$\vec{d_j} = (w_{1,j}, w_{2,j}, w_{3,j}, \ldots, w_{n,j})$$

In the above vectors, n is the total number of unique terms $(t_1, \ldots t_n)$ in the index table. $w_{i,j}$ is the weight for term t_i in a candidate correction d_j, and $w_{i,q}$ is the weight for the term in the user query q.

Because our user query and candidate corrections have an average length of five words, we use the formula suggested by Salton and Buckley [3] for small texts to

calculate $w_{i,u}$, the weight for a term t_i in an expression u:

$$w_{i,u} = (0.5 + \frac{0.5 \cdot tf_{i,u}}{\max_k tf_{k,u}}) \cdot idf_i \quad (1)$$

In the above formula, $tf_{i,u}$ is the number of times a term t_i occurs in the expression u, called term frequency. $\max_k tf_{k,u}$ is the frequency of the most frequent term k in the expression u. idf_i is the inverse document frequency for a term i. Its value indicates the idea that if a candidate correction and the user query share terms that infrequently appear in the whole expression collection, then it is very likely that the two are semantically-related. idf_i is defined as follows:

$$idf_i = \log \frac{|\{d_1, d_2, \ldots, d_m\}|}{|\{d_j, t_i \in d_j\}|} \quad (2)$$

Once we obtain each $w_{i,u}$ in the user query q and candidate correction d_j, we assess semantic similarity by computing the cosine of the angle between the vectors:

$$sim(d_j, q) = \frac{\sum_{i=1}^{n} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{n} w_{i,q}^2}} \quad (3)$$

The value of (3) indicates how semantically close a candidate correction is to the user query.

Probability metric. The probability metric ranks the candidate corrections by $p(d_j|q)$, the probability of a candidate correction d_j being the one that the user wants to write given the user query q. We compute the probability according to Bayes rule [3]:

$$p(d_j|q) = \frac{p(q|d_j) \cdot p(d_j)}{p(q)} \quad (4)$$

We use a method modeled after the Levenshtein distance [3] to approximate $p(q|d_j)$, the probability of the user wanting to write d_j but writing q instead. Levenshtein distance is a method for calculating the minimum number of editing operations needed to transform one string into another. It assigns a cost of 1 to any insertion or deletion operation, and a cost of 2 to a substitution operation. Our approach to calculate the minimum edit distance between a candidate correction and a miswritten idiomatic expression is similarly based on the weighting scheme shown in Figure 3.

Corrections with smaller distances are more likely to be the right ones. Therefore, we use Equation (5) to link the probability $p(q|d_j)$ with $distance(q, d_j)$:

$$p(q|d_j) = e^{-distance(q,d_j)} \quad (5)$$

We use the web frequency of an expression d_j to approximate $p(d_j)$. We calculate the web frequency $F(d_j)$ by sending dj with double quotes around it as a query

Weight(e)	Editing Operation Type
1	an insertion or a deletion
2	a substitution between two particles, or a substitution between two non-particles
4	a substitution between a particle and a non-particle

Figure 3. The weighting scheme for different editing operations.

to Google and retrieving the number of web pages containing d_j. We normalize it using the maximum web frequency of all candidate corrections. Thus:

$$p(d_j) = \frac{F(d_j)}{\max_k F(d_j)} \quad (6)$$

Given (5), (6), Equation (4) becomes:

$$p(d_j|q) = \frac{e^{-distance(q,d_j)} \cdot \frac{F(d_j)}{\max_k F(d_j)}}{p(q)} \quad (7)$$

In the above formula, $p(q)$ remains the same for a given user query q. Thus, the numerator is the only effective part.

The semantic similarity metric and probability metric each generates a value between 0 and 1. We use the product of these values to rank the candidate corrections in descending order.

4. Evaluation

We conducted a formative evaluation of Webtionary. We invited 20 students who are non-native English speakers to use Webtionary whenever they have questions on idiomatic usage. After a month, students entered a total of 145 queries. We use these queries as our test cases.

During the evaluation, it took 4 seconds for Webtionary to respond to a user query on average. In addition, over 74% of the user queries received one to ten corrections. 8% of the user queries did not receive any suggestion. This is because these queries contain uncommon words that are hard to find in online examples. We believe the percentage of these cases will remain low because the chances of using uncommon words by non-native speakers would also be low.

To evaluate correction generation and ranking algorithm, we compared the quality of the corrections generated by Webtionary using 50 and 80 snippets, with results from sending user queries to Google and using the relevant expressions contained in the snippets as corrections.

We evaluated the probability of having a good correction listed in the top five. Figure 4 shows that when Webtionary processed 80 snippets, it outperformed the other

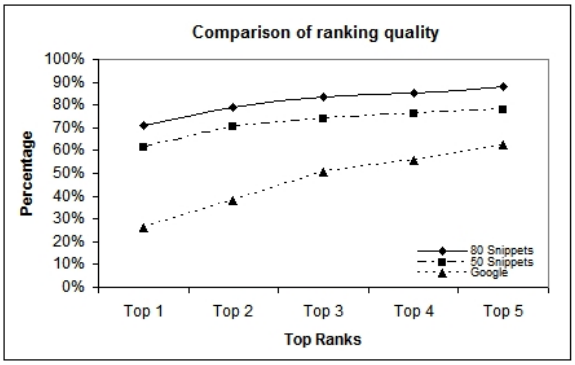

Figure 4. The percentage of test cases that have at least one good suggestion ranked among the top n.

two cases with 71% of the time having the top correction as a good one and 88% of the time having at least one good suggestion in the top 5. This means that for 88% of the time users can scan through the top 5 suggestions and find a usable correction. This is much better than the results obtained directly from Google search where only 26% of the time the top correction is a good correction and 62% of the time there will be at least a good correction in the top five.

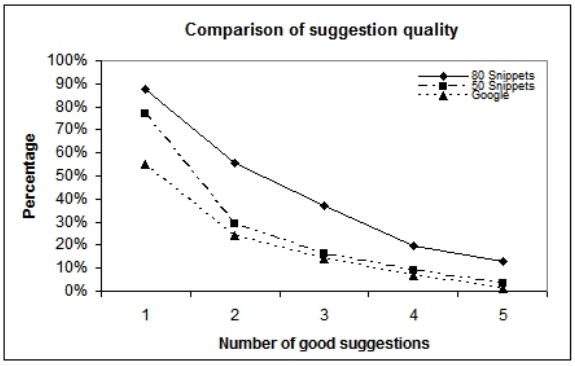

Figure 5. The percentage of test cases that have at least n good suggestions among the top 5.

We also evaluated how many good corrections user could find in the top five. Figure 5 shows that when Webtionary processed 80 snippets, it again outperformed the other two cases with 13% of the time having all five corrections as good suggestions (which is about 13 times better than direct Google search), and 88% of the time having only one good suggestion in the top five (which is about 1.5 times better than direct Google search).

The above results show that Webtionary significantly outperformed direct Google search.

5. Discussion and future work

Webtionary has no control over search options, ranking algorithms, and other technologies used in Google. It would be ideal to build a search engine tailored for linguistic analysis. However, the development of a search engine containing over 10^{10} webpages (the number of webpages currently indexed by Google) requires enormous computational resources and sophisticated expertise that are rarely available. Therefore, building systems over existing search engines remains an endeavor of practical importance.

One direction of future work is to integrate Webtionary into a word processor to provide users with proofread for idiomatic usage. We are currently experimenting with automatic term recognition techniques to extract idiomatic expressions from users writings and verify them using Webtionary.

6. Conclusion

In summary, we have described an approach of using the web to correct idiomatic usage in English. We described Webtionary, a system that provides corrections to questionable idiomatic expressions. While there is still room for improvement, results from the evaluation study show that Webtionary is suitable for practical use. Methods used in Webtionary demonstrate an important step towards using the web to verify English writing without complex linguistic analysis.

References

[1] R. Cilibrasi and P. M. B. Vitanyi. Automatic meaning discovery using google, December 2004.

[2] Googlefight, 2007. http://www.googlefight.com.

[3] D. Jurafsky and J. H. Martin. *Speech And Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, Upper Saddle River, New Jersey, 1 edition, 2000.

[4] A. Kilgarriff and G. Grefenstette. Introduction to the special issue on the web as corpus. *Computational Linguistics*, 29(3):333–347, 2003.

[5] Merriam-webster online dictionary, 2007. http://www.webster.com/dictionary/particle.

[6] D. A. Shamma, S. Owsley, K. J. Hammond, S. Bradshaw, and J. Budzik. Network arts: exposing cultural reality. In *Proceedings of WWW Conference*, pages 41–47. ACM, May 2004.

[7] K. Tanaka-Ishii and H. Nakagawa. A multilingual usage consultation tool based on internet searching: more than a search engine, less than qa. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 363–371, New York, NY, USA, 2005. ACM Press.