

CS276A Final Project Write-up: A Web-based Writing Tool

Joris Morbée (jmorbee@stanford.edu)

Amit Seker (aseker00@stanford.edu)

submitted 12/3/2002

1 Introduction

When one is writing an English text – especially as a non-native speaker – one is often confronted with the problem of how to write a sentence in a way that is correct and sounds ‘natural’. For example: should you write ‘interested in pursuing’ or ‘interested to pursue’? Native speakers may use their linguistic feeling, and sometimes dictionaries provide the answer, but often these two options do not provide a clear solution. As a result, one of the authors of this project has already tried a different strategy, making use of the Internet. When submitting a query like ‘‘interested in pursuing’’ ‘‘interested to pursue’’ in Altavista, one gets frequency information like ‘interested to pursue 350 – interested in pursuing 118035’, which gives a clear indication of the answer. In this project we would like to make use of this observation to build a web-based writing tool.

Our first goal is to create a writing tool that implements the principle to provide the user with frequency information about different linguistic alternatives, including the possibility to restrict the web search to a limited domain (section 2). In section 3 we will explain how we evaluate our system, and we will use our evaluation method to examine the influence of the size and the style of the corpus. To extend the scope of our project beyond this basic functionality, section 4 explores three other applications of our concept. This includes style comparison and making the system more proactive. In section 5 we give an overview of some related work, and section 6 summarizes our conclusions.

2 Implementation

The implementaton of the first goal of our project – build a tool that provides the user with frequency information for different linguistic alternatives – is relatively easy. To estimate the frequency of a linguistic structure, we submit it to the Google search engine, using the Google API [1]. As part of the result, Google returns the number of relevant documents, i.e. the number of documents that contain the linguistic structure. We use this as a measure for the frequency of the word in the Google collection, even when Google indicates that the number of documents is only an estimate.

In order to make this program more accessible, we decided to create a web interface using our CGI account: our “Phrase Comparison Search Engine” is available on the web and can be found at:

[http://cgi-courses.stanford.edu/
~aseker00/cgi-bin/comparephrases.](http://cgi-courses.stanford.edu/~aseker00/cgi-bin/comparephrases)

Given two phrases the engine will inform the user of the frequencies of both phrases, so that the user gets an indication of which structure is probably the best to use.

In addition, the web interface offers the option of getting frequency information from a certain style of websites only. This implies a restriction of the Google search to a certain website or type of website. This could help a user who is aiming at a specific target audience by indicating the “appropriateness” of the phrase in that specific domain, rather than in general. As an example, we let the user choose between the following domains: *General* - for searching in the

entire web, *Academic* - for sites containing “academic style” papers (such as `citeseer.com`), *News* - for “news style” articles (`cnn.com`, `nytimes.com` etc.), *Sports* - for “sports-writing style” articles (`espn.com`, `sportsillustrated.com`, etc.), *Popular* - for casual/unofficial writing style (this could be any popular chat room, mailing lists, or Britney Spears’s portal web site), *USA* – for American English (searching all-American domains like `.gov`), *AU* – the `au` domain, and *UK* – the `uk` domain.

We call this system the *comparephrases* system.

3 Evaluation

3.1 Experimental setup

The question arises how we can evaluate the performance of our system. The need for an evaluation was stressed in the practicum sessions and in the feedback that we received on our proposal. We see two major options to provide a meaningful evaluation:

1. From the perspective of the user, a meaningful way of evaluating a system like ours, is to gather a test group of people who are willing to try to use the system in their writing. These people can then assign scores to the system. This might give an impression of the usefulness and correctness of our idea.
2. For the purpose of evaluation, one could see the system as a “classifier”: when one submits a pair of two phrases (one of which is better or more commonly used than the other), our program “classifies” this pair into one of two categories, by indicating which one of the two phrases is more common. Given this “classifier” behavior, one could try to evaluate the system in the same way that most classification systems are evaluated: by running the system on a large set of examples, and counting the number of times when the classification is right or wrong respectively. This gives a measure of the correctness (and hence usefulness) of the program.

The disadvantages of the former approach are that it is rather subjective, and that one needs to find enough volunteers for the results to be significant. The latter approach yields an objective measure, but requires a test database of relevant pairs of phrases. We decided to pursue the latter method of evaluation, because of the objectiveness of the results.

How can we obtain such a set of test pairs? There is no standard test set for this task. In e.g. the *GramTime News* [2] a number of grammatical questions are answered from a corpus-based perspective, but these questions are not in a suitable format for our system. So we decided to create our own test set: a test set of pairs of phrases, where each pair contains one phrase with a typical use of the language, and one phrase with a slightly different and less common expression. Different options exist to create this database:

- One could try to produce such a database by hand (e.g. based on [2]), but this would require a huge effort and it does not seem to us that this is very interesting from the perspective of this course.
- Another option is to try to generate test queries automatically. While this will almost certainly decrease the quality and relevance of the database, it will probably provide us with a larger database in a smaller amount of time.

For the reasons mentioned, we chose to generate our test database automatically.

How can we do this? As stated above, we need to produce a pair of phrases, one of which represents a typical expression, and one of which represents a slightly different and less common or even incorrect way of expressing the same content. We use the following approach to obtain such pairs:

- We assume that the phrase with the “typical” use of the language can be extracted from the example sentences in WordNet [3]. By feeding WordNet with random words, one can obtain a large number of such typical sentences or expressions.

- After obtaining a typical phrase, we create a corresponding uncommon or even incorrect phrase by transforming the original phrase in one of the two following ways:

1. Every example sentence contains the word that was given as an input to WordNet, or one of its synonyms. We scan the sentence to find that word – or the synonym – and then replace this word by a randomly chosen other synonym. We assume that the WordNet examples use the best suitable synonym in the given context, and that by replacing this word, we create a sentence that still has the same meaning, but is less common or natural.
2. We see if a common preposition occurs with the given word, and then randomly replace the preposition by another common preposition. One problem with this approach is that it is difficult to be sure whether a specific preposition in the example sentence is really attached to the main word. To solve this problem, we only consider prepositions that occur directly after the focus word. In a more advanced version, one could actually try to parse the sentence in order to select a preposition. Other potential disadvantages of our approach include the fact that the randomly chosen preposition is not necessarily semantically related to the original one, so that even a non-native speaker would know that it is not appropriate in the given context. One last problem is the fact that in order to give the new preposition a fair chance in the phrase comparison, sometimes there should be other changes to the sentence: e.g the example “interested in pursuing” versus “interested to pursue”. Solving all these problems would require a more in-depth analysis of the sentence, which we thought was beyond the scope of this project.

Because of the disadvantages of the second ap-

proach, we generated around 95% of our test pairs using the first approach. Some examples of the first approach are: “frequently uttered sentiments” versus “frequently verbalised sentiments” and “experience vertigo” versus “undergo vertigo”. A native speaker confirmed that the first sentence was the better one in each of these cases. An example of the second approach is “careful with money” versus “careful at money”.

The pairs created by this method are often too long, so that they are not suited for our system. One could try to select the most relevant part of each sentence, but this would be very hard to do accurately without a deeper linguistic analysis. Therefore, we just select the WordNet examples that are short enough (three words or less). We implemented our ideas using AWK and generated a few thousands of these test pairs, and after removing doubles, we randomly chose 1000 pairs (2000 sentences) as our test set. Theoretically, we could easily produce a bigger database, but in practice the size should be in the order of magnitude of 1000, because of the restrictions that the Google API puts on the number of queries that can be issued per day.

3.2 Results

With this test set, we can obtain an objective score of the phrase comparison system, by submitting each of the pairs, and counting the number of correct answers (when the system selects the original sentence as the more common one), wrong answers (when the system selects the transformed sentence as the most common one) and “undecided” answers (when there is no difference between both frequencies, in restricted web searches often because both frequencies are zero). We ran these tests with different style restrictions: *General* (all of the web), *UK* (uk domain), *AU* (au domain) and *E-mail* (mailing lists¹). We chose these domains, because – unlike the other domains – they can be expressed using one

¹The e-mail archive restriction is implemented by including `+To +From +Subject +Date +‘Reply-to’` in the search. This spans about two million documents, almost all of which seem to be e-mails (we did a manual inspection).

single query, which is important given the restriction on the number of queries, imposed by Google.

When the *comparephrases* system is run on the generated test set – for each of the web search restrictions mentioned above – we obtain the results in table 1. We notice that, as the corpus becomes smaller, the percentage of correct answers decreases, while the percentage of “undecided” response increases with about the same amount. This can be explained from the perspective of statistics of small numbers: with fewer documents in the collections there are fewer possible frequencies (because they are always whole numbers), so the chance that the two frequencies are the same (“undecided”) becomes bigger. This would suggest that the percentage of wrong answers would also decrease. However, there is also an opposite tendency: in a very crude approximation, one could compare the *comparephrases* experiment to an experiment in which one tries to detect which side of a coin is “weighted” by throwing it a number of times and counting which side of the coin appears more often. The fewer times one throws the coin, the bigger the chance of answering the question wrong (and the bigger the chance of being undecided, at least with an even number of throws). A detailed statistical analysis is of course beyond the scope of this project.

But it is not only the size of the corpus that influences the results. One could also expect that the different styles of different web restrictions would have an impact on the results: e.g. the differences between Citeseer and Geocities, or the differences between UK and AU. To investigate all this, we focus on the comparison between two domains: the General domain on one hand, and the UK domain on the other hand. First of all, these two domains have clearly different sizes. Secondly, we can expect that they have different styles: we can expect that the UK domain will contain mostly British English, while the General domain has a large fraction of American English. The influence of the size of the corpus is clear from table 2: when the corpus size decreases, the biggest stream is from *correct* to *undecided*, while *wrong* gains slightly more from

		UK		
		%	C	W
General	C	67.8	3.8	12.3
	W	2.0	13.0	.9
	U	.0	.2	.0

Table 2: Joint results obtained with *General* and *UK* restrictions (C = correct, W = wrong, U = undecided).

correct than it loses to *undecided* and *correct*. This is consistent with the explanation given above. The exchange of answers between *correct* and *wrong* requires a more detailed study. This exchange means that a number of test set pairs have a different answer depending on the corpus. A number of these cases can be attributed to randomness, but a number of other cases reflect the differences in style between the two search restrictions (i.e. American versus British English). A few examples from our automatically generated test set are given in table 3. A number of these examples are clear illustrations

Test set pair	Frequency	
	General	UK
attendance is mandatory	34900	195
attendance is compulsory	5730	1140
a gravelly voice	1610	44
a raspy voice	4380	41
grade tests	13160	59
mark tests	1260	151
fully grown	53900	3070
full grown	98600	1910
a large city	73200	2380
a big city	118000	2220
African nations	92800	10400
African states	93700	4290
intentional damage	5600	238
deliberate damage	3600	980
spacious skies	5580	33
wide skies	671	102

Table 3: Examples of differences between *General* and *UK* search restrictions.

Search restriction	# documents	% correct	% wrong	% undecided
General	3 billion	83.9	16.9	.2
UK	8 million	69.8	17.0	13.2
AU	4 million	66.8	15.7	17.5
E-mail	2 million	64.6	16.5	18.9

Table 1: Results of the *comparephrases* system on the generated test set, for different search restrictions.

of the difference between American and British English: *attendance is mandatory* versus *attendance is compulsory*, or *grade tests* versus *mark tests*. Other differences are rather illustrations of cultural differences, e.g. the use of *African states* versus *African nations*. And the Web also introduces some artifacts: *spacious skies* is not necessarily more common than *wide skies* in everyday American speech, it just happens to appear in the first line of the popular song “America the Beautiful”, hence its relatively high frequency on the web.

In general our evaluation shows us that the systems works fairly accurately, and is capable of using different styles.

4 Extensions

After building and evaluating our basic functionality, we conducted some experiments to extend the scope of this project, and explore other applications of the same ideas. The three extensions are *phrasecheck*, *comparestyles*, and *phrasefreq*. They are accessible on the web, at the URL:

`http://cgi-courses.stanford.edu/
~aseker00/cgi-bin/*.`

where `*` should be replaced by *phrasecheck*, *comparestyles* or *phrasefreq*.

4.1 Checking phrases: *phrasecheck*

The first extension of our basic idea and is demonstrated in the *phrasecheck* interface. The idea is to make the system more proactive, so that, given a full text, it detects which structures are not very common or unnatural. One could expect that a

thorough solution of this problem would require deep language understanding, but we were interested in exploring to what extent raw frequency information could be helpful.

Given a piece of text, our system will flag words and groups of words that seem incorrect or unnatural, either because of a spelling error (if the word is not in the dictionary and has a small frequency on the Web) or because of a structure error (if the combination of words around it has a low “structure score”). So if a user makes a spelling error that the spelling checker does not catch (such as typing “want top go” instead of “want to go”), the structure checker will flag it. The system is based on the same idea as *comparephrases* - a user can enter a text and choose a domain. The system will then process the text as follows: it will start from the first word and slide a window of length three words over the text, and submit a query to check the frequency of the phrase in the specified domain. At the same time the system checks each word for spelling errors by looking up if the word is in the WordNet dictionary, or if it is widespread enough over the web. The extra web check is useful, because the WordNet dictionary does not include certain words, like stop words (such as “the”, “and”, “from”, “for” etc) or names and new trendy words. The window is reset for every ‘.’, ‘”’, ‘(’ or ‘)’, ‘!’ or ‘?’, and it ignores ‘,’. So for example, for the text: *hi, my name is “john doe”. and I am (just checking) to see ...* the window will look like: “hi”, “hi my”, “hi my name”, “my name is”, “john”, “john doe”, “and,” “and I”, “and I am”, “just”, “just checking”, “to”, “to see”, ... One-word-structures do not get structure-processed, only spell checked. After we have the window’s frequency we need to

check if that frequency score is low enough to consider it as a “structure error”. To do that we have to consider the relative frequencies of the individual words in the phrase so that a window that contains uncommon words will not get flagged just because the words in it have a low frequency in the domain. There are different ways to normalize the frequency of the window. The most obvious solution is to divide by the frequencies of the individual words (especially since we have this information available from the previous step), but the problem is that this creates very low scores for structures that contain very common words. An elegant and for the purpose of this project adequate normalization is to divide by the frequency of a query made up from the words in the window but without the quotes. The advantage here is that Google automatically excludes very common words like “the”. Once the score is obtained, one needs to decide which structures get flagged. The cut-off value for the *General* domain is 0.0002, and for the rest of the domains it is 0.2, corresponding to a factor 1000 in corpus size. We got this value by trial and error on various phrases. This value promises that spell errors will most likely be flagged (unless somehow those errors still make “structural sense”) and that most of the possibly structurally correct structures will not get flagged. Of course there is still a lot of room for optimization, but our main objective was to explore if our idea for making a more proactive system based on simple Web statistics was a reasonable one. ²

4.2 Comparing styles: *comparestyles*

Our second extension is inspired by the work in section 3: instead of comparing two phrases in one domain, one could compare how one phrase scores in different domains. This is demonstrated in the *comparestyles* interface. A user can submit a phrase and the domains of interest and then the system replies with the frequency of the phrase in each specified domain. Because the different domains have dif-

²In that perspective, this system (like *phrasefreq*) is designed to illustrate our concept, and it is not yet optimized for efficiency (one possible way to do this is e.g. to parallelize the processing of different sentences).

ferent sizes, the output is normalized to take into account the size of the domain. To achieve that we figure out in advance the size of the domain by obtaining the frequency of a common phrase in the domain. We chose the phrase “a” as the size indicating query since “the” and “and” result in a GoogleSearch exception. This method of normalizing the frequencies can probably be optimized, because a few experiments show that it does not do a good job when the differences in size between the corpora are substantial (e.g. *UK* versus *General*).

4.3 Frequency of phrases in different styles: *phrasefreq*

We can extend the style comparison to longer pieces of text. This idea is demonstrated in the *phrasefreq* interface. A user can submit a text and choose the domains of interest and the system will calculate partial scores of the text for each domain, and provide the score of each domain relative to the others. The score processing is done as follows: as before we take a 3 word sliding window over the text using the same procedure as in *phrasecheck*, only this time we do submit one word phrases as well. The frequency of the window is then normalized as in section 4.2 (with the same disadvantages) to take into account the size of the domain. So now as the window slides over the text we build a scoring matrix with an element for each domain (style) and each windows. Each entry in the matrix corresponds to the frequency of the window-phrase in the domain divided by the size of the domain. Once the text processing is done we compute a score matrix in which each domain (style) is compared to each of the others. We do that by dividing every window score of each style by the corresponding window score of the other style. So if a certain window will have a high score in one style relative to a second style the score will be high. By adding up these numbers, we can see how a certain style scores in comparison to another style. The system outputs both score matrices.

5 Related work

The idea of using frequencies from the Web as a source of linguistic information about common usage of the language does not seem to be very widespread. Vaufreydayz et al. [6, 7] and Géry et al. [4] collect documents from the Web to build large corpora. In [6, 7] the aim is to build spoken language models for French spoken language speech recognition (for which written texts are less suited than for e.g. dictation). In [4] the corpora are used for noun phrase extraction for information retrieval. The advantages of an internet corpus is that it can be very large and diversified and that its dynamic aspect makes it possible to follow vocabulary evolution.

These authors, however, do not make direct use of Web search engines to collect the information. Grefenstette and Nioche [5] on the other hand make direct use of Altavista queries on a limited number of simple words to estimate the size and the evolution of the presence of a given language on the internet.

6 Conclusion

In this project we have shown that frequency information from the Web (as obtained with Web search engines) can be used to create a writing tool that helps users choose the most common or natural expression among different linguistic alternatives (within a given style or domain). Using WordNet we have created a test set of pairs of linguistic structures, which allowed us to evaluate our system and demonstrate the influence of the size and the style of the domain to which the search was restricted. We have then extended our system so that it proactively finds uncommon structures, and so that it allows to see how appropriate one phrase is in different styles. These extensions offer possibilities for future research.

References

[1] Google APIs, <http://www.google.com/apis>.

- [2] Växjö universitet – GramTime News, <http://www.hum.vxu.se/publ/gtn/>.
- [3] WordNet, a lexical database for the English language, <http://www.cogsci.princeton.edu/~wn>.
- [4] M. Géry, M. Hatem Haddad, and D. Vaufreydaz. Web as huge information source for noun phrases integration in the information retrieval process, 2000, citeseer.nj.nec.com/525316.html.
- [5] G. Grefenstette and J. Nioche. Estimation of english and non-english language use on the WWW. In *Proceedings of RIAO'2000, Content-Based Multimedia Information Access*, pages 237–246, Paris, 12–14 2000.
- [6] D. Vaufreydaz, M. Akbar, and J. Rouillard. Internet documents : a rich source for spoken language modeling, ASRU 99 Conference, 1999, citeseer.nj.nec.com/vaufreydaz99internet.html.
- [7] D. Vaufreydaz and M. Géry. Internet evolution and progress in full automatic French language modelling, ASRU, Madonna di Campiglio, Italy, 2001.