To: Ted Pederson
From: Brent Eggenberger
Date: May 7, 2004
Subject: Google Sets

Introduction:

The amount of information available on the Internet today is growing very rapidly. Through the use of Google, people are able to query this information at an incredible speed. Although Google has an excellent search engine, the new Google Sets have been drawing quite a bit of attention. Google Sets are designed to return a list of related words from the user's set of input words. The output from Google Sets can be quite accurate. With this document, I propose to try to mimic Google Sets. I intend to use information from Google to identify sets of related words.

Statement of Problem:

It is often difficult to find the true meaning of a word in the English language. The definition of a word may vary between different dictionaries. In order to add a related word to a set, the general meaning of the other words must be known. Discovering a new set of related words is a useful tool. It helps produce a broader vocabulary for all writers. This project will be very similar to a thesaurus.

Finding a solid technique to compare related words can be tough. One idea of (Jeh and Widom 2002) is that "two objects are similar if they are related to similar objects." Through extensive work, these two have devised a general similarity measuring tool called SimRank. They present their algorithm for generating a SimRank score in their paper. I intend to use their idea of similarity in my algorithm. In a query from Google, I think related words to the query term(s) will often appear in the first few pages.

This paper will be a valuable resource for creating a powerful algorithm in finding sets of related words.

Producing an efficient algorithm will inevitably rely on Google's page ranking procedure. Understanding how it will affect my outputs is crucial. Although it is difficult to find Google's exact ranking algorithm, a group from Stanford discusses a general page ranking procedure (Page, Brin, Motwani, and Winograd 2001). Their method incorporates how much human attention each page has received. They demonstrate their procedure in a published paper. In general, a page receives a higher ranking when many other pages link to it. This is called backlinking. So, if there are multiple pages linking to page X, then page X will have a higher ranking. Having the knowledge of what to expect from Google will help when ranking my set of possible related words.

Plan of Action:

In order to mimic Google Sets to my best abilities, I will have to work in a series of steps. The very first thing that needs to be completed is setting up an account with Google to obtain a developers kit. This will give access to Google's services, such as the ability to perform a query inside a Perl script. The next step is to perform a query on each of the user's words that he/she has for input. The results from these queries will help me find text that may contain related words. My hypothesis is that most of the related words are going to appear in the same text as those initial words. Once I have the results of the queries I will be able to start searching through them.

In order to accurately find a set of related words, I propose to use the following algorithm:

1. Perform a query on the user’s input words using the Google API.
2. Store the results of the cached pages into one text file
3. Remove any associated HTML tags from the text file.
4. Continue to remove any garbage text from the text file.
5. Apply a part of speech tag to all remaining words in the text file.
6. Record the frequency of the types in the text.
7. Apply a weight to each word, based on the cached page the word occurs in.
8. Remove words from the text that are not the same part of speech as the user’s input words.
9. Based on the frequency and rank of each word, display the results.

To illustrate my algorithm, I will offer an example as a means of clarification. I use the words: red, brown, green for my input set of related words. Next, I perform a query on the combined words (red, brown, and green). From the results, I generate one large text file containing all of the cached pages. I proceed to remove all HTML tags. I then filter out any garbage text. At this point, I will use Eric Brill’s part of speech tagger and tag each word of my text. I then apply a rank to each word when recording the frequency. If a word appears in the first returned cached page, the word frequency is multiplied by 10. A word appearing in the second cached page will have the frequency of each word multiplied by 9 and so on down the 10 cached pages. Next, I remove words that are not the same part of speech as red, brown, green. I am now left with my results. I display the resulting words according to their frequency to standard out.

Evaluation Measure:

The task of finding related words can be very difficult using the English language. In order to judge the accuracy of my final project, I will need to execute my program multiple times on different sets of words. I intend to use two sources for a comparison tool. This will include the GoogleSets web page and WordNet 2.0. The GoogleSets web page allows a user to enter a set of related words and Google returns a new list of related words. The WordNet 2.0 software package will be incorporated into my program. Using a Perl module, I can access the huge database provided by WordNet. I intend to display all relevant hypernyms and hyponyms for each input word to the user. After I have run my program several times, I will manually test it for precision and recall on the top 25 words returned. To do this I will use the following formulas:

$$\bullet\text{Recall} = \frac{\text{\# of words appearing in my output that appear in GoogleSets or WN output}}{\text{total \# of words in the output of GoogleSets or WN}}$$

$$\bullet\text{Precision} = \frac{\text{\# of words that appear in my output that appear in GoogleSets or WN}}{\text{total \# of words that my program returned (25).}}$$

The higher the precision and recall, the more accurate my program performed.

Results:

I tested my program on a variety of different sets of words. To illustrate my first example, I ran my program on three different nouns: mud, gravel, and dirt. Table 1 summarizes my results.

Table 1: mud, gravel, dirt

	GoogleSets	WordNet
Recall:	25.00%	9.30%
Precision:	16.00%	16.00%

These numbers indicate, even though they are low, that my program can be effective at retrieving a set of related words. The lower numbers may be due to Eric Brill's part of speech tagger that I used. His tagger defaults to a noun when it can not determine a word's part of speech.

My second example displays how well my program can handle proper nouns. Tiger Woods, Phil Mickelson, and Ernie Els were used for my input set. The resulting set of words from my program, to an extent, all pertained to golf. Table 2 summarizes the results:

Table 2: Tiger Woods, Phil Mickelson, Ernie Els

	GoogleSets	WordNet
Recall:	35.42%	0.00%
Precision:	68.00%	0.00%

This set of words performed very well in comparison to GoogleSets. It was able to return many names of professional golfers. My program does not have the capability to retain two word phrases like Google, so it can not display a person’s whole name. Rather, the first and last names are separated, but usually not to far away on the list. WordNet did not prove to be a very good comparison for this set of data. The database from WordNet tends to only contain historical or famous names.

My final example illustrates the program’s effectiveness with verbs. Three verbs related to the action moving were given as the input set. They included walking, running, and swimming. Table 3 displays the results:

Table 3: walking, running, swimming

	GoogleSets	WordNet
Recall:	8.51%	7.41%
Precision:	16.00%	8.00%

Using a tagger, my program is limited to producing words that are the same part of speech as the input words. In this case, I felt like it produced an excellent set of words. The entire list was other verbs, usually words ending in ‘ing’. These numbers do not reflect my intuition. Google, more than WordNet, returned a list that contains nouns and adjectives. This may be the cause of the lower numbers.

Conclusion:

I have presented a moderately accurate program that produces sets of related words. It is able to retrieve information from Google, parse the text, apply a part of speech tag, and display the results according to the word's frequency. My experiments show that it holds the qualities to be an effective program. In order to make the recall and precision numbers higher, and better ranking algorithm needs to be devised. Although my program does not always produce the same results as GoogleSets, most of the time it is an effective tool for producing words that are related.

Works Cited

L.Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web, Stanford University, Computer Science Dept technical report, October 2001, http://citeseer.nj.nec.com/368196.html.

Jeh, G., and Widom, J. SimRank: a measure of structural-context similarity. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. July 23-26. 2002. Edmonton, Alberta, Canada