Identifying Sets of Related Words from the World Wide Web

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Pratheepan Raveendranathan

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

July 2005

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of master's thesis by

Pratheepan Raveendranathan

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Dr. Ted Pedersen

Name of Faculty Adviser

Signature of Faculty Advisor

Date

GRADUATE SCHOOL

Contents

List of Tables

List of Algorithms

Abstract

As the Internet keeps growing, the number of Web pages indexed by commercial search engines such as Google increases rapidly. Currently, Google reports that they index over 8 billion Web pages. The type of information available through the Web is very diverse, from publications to electronic encyclopedias to information about products. In short, the Web is vast and huge. Until recently, the Web has not been used to acquire information about words in order to better understand Natural Language. However, we believe that there is a need to develop methods that take advantage of the huge amount of information on the Web. Hence, this thesis focuses on finding sets of related words by using the World Wide Web.

This thesis presents three new methods for using Web search results to find sets of related words. We rely on the Google API to obtain search engine results, but in principle these methods can be used with any search engine. They rely on pattern matching techniques in addition to various measures or relatedness that we have developed.

In addition to finding sets of related words, we also explore the problem of Sentiment Classification. This was motivated by a desire to find a practical application for the sets of related words we discover. As such we extend the Pointwise Mutual Information - Information Retrieval (PMI-IR) measure described in (Turney, 2002) to be used with Google in order to discover sets of related words. These sets are then used as seeds in our Sentiment Classification algorithm.

1 Introduction

The overall goal of this thesis research is to use the World Wide Web as a source of information to identify sets of words that are related in meaning. As an example of a set of related words, consider the words *Toyota* and *Ford*. These names are related to each other as automobile manufacturers. Hence, if we take *Toyota* and *Ford* to be our input set, an expanded set of related words would be the set that includes names of other automobile manufacturers such as *Nissan, GM, Honda and Chevy*. As another example, consider the the initial set to be *January, February and May*. In this case, an expanded set of related words would be a set containing the names of the other months in a calendar year. Hence, a set containing *March, April, June, July* and other months would be the most related set of words. The goal here is to retrieve relevant information from the web, via a search engine such as Google, and extract enough information from the retrieved content to find sets of related words.

Since our research is very much dependant on both the quantity and quality of the Web content that is used to group words that are related in meaning, we used a commercial search engine such as Google to gather information about words from the Web. Google has a very effective ranking algorithm called PageRank which attempts to give more important or higher quality web pages a higher ranking [9]. One additional reason for selecting Google over any other search engine is based on the fact that Google has a very easy to use API that allows programs to interact with it's database of web pages. In addition, Google also indexes more than 8 billion web pages.

Using the World Wide Web (WWW) as a source of information has many advantages. As mentioned earlier, the information available in the web is vast, and the information is dynamic. The dynamic nature of the web can be advantageous. When experiments were conducted using the algorithms developed in this thesis a very interesting result was observed. Different periods of times produced different sets of related words. Often, these sets of related words reflected the current news associated with the given input set. As an example, when the search terms *George Bush* and *Bill Clinton* were used as input during the month of January 2005, the kind of words returned by the Google Hack was heavily induced by the unfortunate tsunami disaster in South East Asia. Words such as relief and aid were among the dominant terms. When the same search terms were issued a couple of months later, the sets of words changed to a more intuitive set of words such as *Jimmy Carter, President, Republican and Democrat*.

Sets of related words that are predicted based on information retrieved from the WWW can be used in many ways to identify the sentiment associated with certain text, and commercial products. For example, a brand such as *Toyota* or *Ford* can use sets of related words to identify the kind of words associated with their products [1]. For example, if for an input set such as *Toyota Corolla, Toyota Camry* the set of related words returned are *reliable, affordable, and gas efficient*, it can be concluded by a human that there exists a positive reaction on the web regarding the some of cars produced by Toyota. Another application of sets of related words would be the extension of the Point wise Mutual Information - Information Retrieval algorithm described in [10].

The overall contributions of this thesis are outlined as follows.

1. Developed an algorithm based on co-occurrence and frequency counts to identify sets of related words (Algorithm 1).
2. Developed an algorithm based on co-occurrence, frequency counts and a relatedness measure to identify sets of related words (Algorithm 2).
3. Developed an algorithm based on co-occurrence, frequency counts, a relatedness measure and log likelihood score to identify sets of related words (Algorithm 3).
4. Adapted the relatedness measure described in [6] to the WWW. The measure is effective in distinguishing noisy words, and is also effective in ranking words according to their relevance to a given set of search terms.
5. Extended the PMI-IR algorithm to use multiple sets of positive and negative connotations to classify the sentiment of reviews using Google.
6. Compared the sets of related words predicted by our Algorithms to other programs such as Google Sets that was developed by Google.

Other contributions of this thesis are as follows.

1. Developed and released an open source PERL package named Google Hack. The package is freely available through http://search.cpan.org/ prath/WebService-GoogleHack-0.15/

2. Developed a web interface through which can be used to interact with Google Hack.

2 Background

Our research is very much depedant on both the quantity and quality of the Web content that is used to group words that are related in meaning. Hence, we use the commericial search engine Google to gather information about words from thr Web. Google has a very effective ranking algortihm called PageRank which attempts to give more important or higher qualitiy web pages a higher ranking [9, 2]. In addition to the quality of the results returned by Google, our research is also dependant on interface that allows our algorithms to interact with a search engine. The Google API is such an interface that allows programs to interact with the Google search engine. In this section, we will first try give an idea of how PageRank works, and we will also try to give an idea of the kind of features allowed by the Google API.

2.1 Google and Page Rank

Google uses an algorithm called PageRank to determine the importance of a web page. The basic idea behind the PageRank algorithm is to use the number of links to a web page as a source for ranking [9, 2]. Highly linked pages are ranked more important than pages that do not have as many links to them. The links themselves are divided into back links and forward links. Back links are links that refer to a certain page. Though this is the basic idea behind the algorithm, the number of back-links alone cannot guarantee a good ranking. The ranking also depends on the rank page that is linking to it. If for example, a web page has only one back link, but the back link is from a credible or well linked web page such as the Stanford University homepage, then the web page would be given a high ranking.

A simplified PageRank function R, for a web page *u* can be defined as,

$$R(u) = c * \sum_{v \in B_m} \frac{R(v)}{N_m} + cE(u) \tag{1}$$

Where, F_u is the set of pages that u points to, and B_u is the set of pages that point to u. Let $N_u = | F_u |$ be the number of links from u and let c be a factor used for normalization. Let E(u) be some vector over the web pages that corresponds to a source of rank [9, 2].

As can be seen from the function, the rank of a web page is equally divided amongst its forward links. This ensures that the weights of a web page, is evenly distributed to the pages that it points to. The basic idea

behind page rank can be illustrated through the following example. If web pages A, B and C have have ranks 6, 6, and 9 respectively, a web page D that is linked to by A, B and C would receive an overall ranking of (6 + 6 + 9) / 3, which is 7.

To implement PageRank, Google uses the services of a web crawler, which constantly crawls through the web, finding web pages, and downloading them on to the Google servers. Each URL is then converted into a unique integer ID which can be stored in the Google database along with the hyper-link to the page. These web pages are then assigned a rank using the PageRank algorithm. Currently, Google has over 8 billion web pages in its database [1] .

The identity of a web page is often represented through its title. Keeping this fact in mind, Google uses a PageRank and a title based searching method to find web pages for a user query. When the user enters a query, Google retrieves web pages whose title matches the queried words. These web pages are then sorted using PageRank. This way, the search results are ensured high precision and quality.

PageRank is designed to handle common case queries well. An example of a common case query can be ”flower”. Querying for the term ”flower” in Google, will simply return popular commercial sites that allow you to buy flowers.

In conclusion, PageRank is a simple, but, a very clever algorithm which uses the link structure of the web to assign importance to web pages. This in turn allows Google to be a very successful and efficient search engine.The actual method used by Google has evolved considerably since Page Rank was proposed and the details are not known to us.

2.2 The Google API

The Google API is a simple programming interface through which software developers can query the Google web page database[2]. The API allows programs to retrieve information such as spelling suggestions, number of hits and cached web pages for queries. The Google API can also be used with various programming languages such as Java, Perl and VisualStudio.NET. However, the Google API does have a number of limitations. Firstly, the number of queries allowed per license key per day is a 1000 queries. Furthermore,

[1] $http://www.google.com$

[2] $http://www.google.com/apis$

each query can at most retrieve 10 web pages at a time. Hence, if a particular query returns a 1000 hits (web pages), then for a program to retrieve the first 30 web pages, 3 queries must be issued through the API. The queries have to be structured such that the first query returns the pages ranking from 1 to 10, the second query returning the pages ranked from 11 to 20, and finally a third query to retrieve the web pages ranked from 21 to 30. Other than these two limitations, one additional factor that must be considered is the number of hits returned by the API. According to Google, the Google API returns an estimated number of hits for a query instead of the actual number of hits. Hence, if a query returns 10000 as the number of hits, the actual number of hits might be higher or lower than the estimated number.

2.3 Web Stop Words

One of the biggest drawbacks to using the World Wide Web as a source of information is the amount of noisy data that is present throughout each web page that is used as a result. Specifically, certain words occur a very large number of times regardless of a relation to the search term, thereby creating noisy data. For example, words like *links*, *url* and *www* occur in almost every web page, and it was important to identify these words as not being related to the search terms.

One method of identifying these unrelated terms is to maintain a list of stop words that are specific to the web. Identifying and removing web stop words played an important role in reducing the noise in the results of Algorithm 1.

However, using a Web Stop List comes with some disadvantages. Words that occur in the Web Stop List cannot be used as input terms to the Google Hack Algorithms.

3 Methodology

3.1 Algorithm 1

The motivation of Algorithm 1 is to serve as a baseline method that uses direct pattern matching techniques to identify sets of related words using Google as a source of information. Algorithm 1 is based on the idea that words related in meaning tend to co-occur within the same context. Hence, the general idea behind the algorithm is to create various search engine queries to Google based on the given input terms and to retrieve the content of those web pages returned by Google for each query. The web page content retrieved for each Google query is then tokenized into list of words and frequencies. Finally, words that occur frequently and that are common to the different sets of web page content are assumed to be a set of related words to the input terms.

Pseudocode for Algorithm 1 is given in the following pages. Algorithm 1 is limited to two words in the initial input set. The Algorithm takes in as arguments the number of web pages to parse numofPages, the frequency cut off frequencyCutoff, and the number of iterations numIterations. Consider the following example trace of Algorithm 1. Let the initial input set S_1 be {gun, pistol}. Let numofPages be equal to 10, numIterations be equal to 1, and frequencyCutoff be equal to 10. The first step of the algorithm creates queries to Google based on the different possible permutations of the given two words. Hence, for the given set S_1, the following queries are created:

1. ”gun”
2. ”pistol”
3. ”pistol” AND ”gun”
4. ”gun” AND ”pistol”

Therefore, the initial set of queries $QuerySet_1$ is ”gun”,”pistol”, ”gun AND pistol”,”pistol AND gun”.

As documented by Google, the order of search terms in a query to Google actually matters. For example, queries 3 and 4 do not always produce the same results. Once the queries have been constructed, they are issued to Google one at a time. The idea is to retrieve a set of numofPages web page links for each query

and traverse the links to retrieve and parse the content of those pages to identify sets of related words. The set of links retrieved for each is query given in tables 1 through 4. In addition to traversing each link that Google returns, the Algorithm also parses the content within the particular web pages for more links. These links are also traversed for text.

Therefore, a set of related words for the query ”gun AND pistol” will be formulated from the text that is parsed from the top numofPages web pages that Google returns along with the text parsed from the web pages that are linked to by the top numofPages web pages that are returned by Google. Parsing a web page is basically removing HTML tags, JAVASCRIPT etc and retrieving only the plain text from the web pages. Once the web pages have been parsed, the Algorithm removes stop words from the plain text. The stop word list used in Google Hack consists of the words in the standard Smart Stop List along with stop words that are specific to web pages.

Once the web pages have been parsed and stop words removed, the plain text is then tokenized into a list of words and a frequency of occurrence count of each word is also maintained. Words that occur less than the given frequency cutoff *frequencyCutoff* are discarded, where in this example words occurring less than 10 times are discarded. Table 5 shows the list of words for the query “gun AND pistol” after discarding the low frequency words. It can be seen from column 4 of Table 5 that only words occurring at least 10 times are maintained in the list.

This process is repeated for each query in QuerySet_1. This would result in four individual sets of words, with each set representing the set of related words for a particular query. Table 5 shows the 4 sets of words retrieved for the current example. The next step in the algorithm is to find the set of intersecting words between these four individual sets. Let the four sets of related words be A, B, C and D. Hence, if a word occurs in set A, B, C or D and it occurs in at least one other set A, B, C, or D other than the set itself, the word is considered to be a related word. Formally, the first set of related words $\text{intersectingWords}_1$ would be:

$$\text{intersectingWords}_1 = (A \cap B) + (A \cap C) + (A \cap D) + (B \cap C) + (B \cap D) + (C \cap D)$$

Hence, if a word occurs in set A with frequency x, and the same word occurs in set D with frequency y, the frequency of that particular word in Word_1 would be x+y. Also note that if a word had occurred in set A with frequency x, and set B with frequency y, where both x and y are greater than the frequency cut off, and the

same word occurred in set D, however was discarded because its frequency z was less than the cut off, the final frequency for that particular word will be x+y+z. The set of words resulting from this set operations is given in Table 6. Let Word_1 equal to the set of related words in Table 6. Note that if either of the search terms occur as a result in either of the four set of words, it is discarded. Hence, the search terms of the current iteration should not appear as a resulting related word. Since the the number of iteration numIterations had been set to 1, $\text{intersectingWords}_1$ would be the set of related words for ”gun” and ”pistol”.

Now, consider the same the input set with different parameters. Let the number of web pages still remain 10, however, take the number of iterations numIterations to be 2, and the frequency cut off frequencyCutoff to be 20. The same process mentioned in the previous example is repeated, however, since the frequency cutoff is now 20, the number of words resulting in $\text{intersectingWords}_1$ is reduced to 4 terms. The results of iteration 1 is given in table 7. Since the number of iterations had been set to 2, the Algorithm continues to execute. In iteration 2, the Algorithm uses as input set the words in $\text{intersectingWords}_1$ which was retrieved from iteration 1. Hence, for all $i > 1$, if the iteration number is i, the set of input words would be from the set t of related words that were identified from the previous iteration i-1, which is $\text{intersectingWords}_{i-1}$. So in this example, for iteration 2, the input set S_2,

S_2 = {shooting, airsoft,guns,cases}

Now, the algorithm goes about creating queries to Google based on the different possible permutations of the 4 input words. The following set of queries QuerySet_2 are created:

QuerySet_2 = {*shooting, airsoft, guns, cases, shooting AND airsoft, shooting AND guns, shooting AND cases, airsoft AND shooting, airsoft AND guns, airsoft AND cases , gunsAND shooting, guns AND airsoft, guns AND cases, cases AND shooting, cases AND airsoft, cases AND guns*}

The process of issuing queries to Google, and retrieving web pages from Google is repeated for each query in QuerySet_2. The resulting links are traversed, and plain text is retrieved from these web pages and tokenized into lists of words in the same format as mentioned earlier. Finally, a second set of related words for *gun* and *pistol* is found by finding the set of intersecting words between all the sets of words for each query in Query_2. The resulting set of related words $\text{intersectingWords}_2$ is given in Table 8.

One important thing to note in iteration 2 is that, if it returns as a set of related words the actual initial input set, it gives a strong indication that the search terms and the current set of search terms are closely related. This concept can be generalized such that if the set of related words returned for iteration i actually contains the search terms of iteration i-1, then the words returned and search terms are tightly related.

The second iteration has considerably increased the number of related words for *gun* and *pistol*. It can also be seen that *gun* and *pistol* are returned as one of the top terms in intersectingWords$_2$. Note also that since the algorithm depends solely of the frequency of occurrence, iteration 2 has increased the number of noisy terms in the set of related words. Therefore giving enough motivation to develop Algorithm 2 which can distinguish between relevant terms and noisy terms by using some sort of relatedness measures.

Algorithm 1 Algorithm 1

1: **function** Algorithm1 ($searchStrings[], numSearchTerms, numofPages, frequencyCutoff, numIterations$)
2: $k \leftarrow 1$
3: **for all** $i \leftarrow 1$ to numSearchTerms **do**
4: $intersectingWords[k][i] \leftarrow searchStrings[i]$
5: **end for**
6: **for all** $k \leftarrow 1$ to numIterations **do**
7: Permute(QuerySet, intersectingWords, k, sizeof(intersectingWords[k]))
8: **for all** $i \leftarrow 1$ to sizeof(QuerySet) **do**
9: $results \leftarrow GoogleSearch(QuerySet[i])$
10: **for all** $j \leftarrow 1$ to R **do**
11: $content \leftarrow content + getWebPageContents(results \rightarrow url[i])$
12: getLinks(linksArray,content)
13: **for all** $n \leftarrow 1$ to sizeof(linksArray) **do**
14: $content \leftarrow content + getWebPageContents(linksArray[n])$
15: **end for**
16: $content \leftarrow removeHTML(content)$
17: $content \leftarrow removeStopWords(content)$
18: **end for**
19: getWordsAndFrequency(content: in, frequencyCutoff: in ,wordSetArray[i][]: out, wordFrequencyArray[i]: out)
20: **end for**
21: getIntersectingWords(wordSetArray: in, wordFrequencyArray: in, insertSectingWords[k+1]: out)
22: **end for**
23: return InterSectingWords
24: **end function**

```
1: procedure Permute(QuerySet[], intersectingWords[], numIterations, numofPages, numSearchTerms)
2: if I > 1 then
3:     start ← numofPages
4: else
5:     start ← 1
6: end if
7: k ← 1
8: for all i ← start to numSearchTerms do
9:     for all j ← start to numSearchTerms do
10:        if j! = i then
11:            QuerySet[k++] ← "intersectingWords[i]" AND "intersectingWords[j]"
12:        end if
13:    end for
14: end for
15: end procedure
```

Table 1: Resulting Top 10 Web Pages for query ”gun”

Set of Links for query “gun”
$http://www.thesmokinggun.com/$
$http://www.thesmokinggun.com/archive/1013043mackris1.html$
$http://www.gunbroker.com/$
$http://www.gunowners.org/$
$http://www.doublegun.com/$
$http://www.ithacagun.com/$
$http://www.imdb.com/title/tt0092099/$
$http://www.gunandgame.com/$
$http://www.gunaccessories.com/$
$http://www.guncite.com/$

Set of Links for query “pistol”
$http://www.idpa.com/$
$http://www.bullseyepistol.com/$
$http://www.zvis.com/dep/dep.shtml$
$http://www.crpa.org/$
$http://www.nysrpa.org/$
$http://www.auspistol.com.au/$
$http://en.wikipedia.org/wiki/Pistol$
$http://www.pistolgrip.net/$
$http://hubblesite.org/newscenter/newsdesk/archive/releases/1997/33/$
$http://www.pistolpeople.com/$

Table 2: Resulting Top 10 Web Pages for query ”pistol”

Table 3: Resulting Top 10 Web Pages for query ”pistol AND gun”

Set of Links for query “pistol AND gun”
$http://quizilla.com/users/ReverendDeWald/quizzes/What\%20Gun\%20Are\%20You\%3F/$
$http://www.bullseyegunaccessories.com/$
$http://www.greatoutdoors.com/ambackcom/opticssights/pistollasergunsights.html$
$http://fit4martialarts.com/store/airsoft_beretta_92f_92fs_auto_pistol_gun_spring_a62007_62121.php$
$http://fit4martialarts.com/store/airsoft_swat_seal_tactical_pistol_auto_gun_leg_holster_a62004_31688.php$
$http://www.safetysafeguards.com/site/402168/page/57959$
$http://www.docs.state.ny.us/DOCSOlympics/Combat.htm$
$http://www.peopleview.net/pistolbbgun.html$
$http://www.sail.qc.ca/catalog/detail.jsp?id=2880\&category=308$
$http://www.airgundepot.com/eaa-drozd.html$

Table 4: Resulting Top 10 Web Pages doe query ”gun AND pistol”

Set of Links for query “gun AND pistol”
$http://www.usgalco.com/$
$http://www.minirifle.co.uk/$
$http://www.bullseyegunaccessories.com/$
$http://www.gunsworld.com/frenchguns_home_us.html$
$http://www.soundrangers.com/category-results.cfm?storeid=1cat_id=0034$
$http://www.camping-hunting.com/$
$http://www.pelican-case.com/pelguncaspis.html$
$http://www.nimmocustomarms.com/$
$http://www.panelspecialties.com/guncab.htm$
$http://www.mommydreams.com/toy-pistol.html$

Table 5: Algorithm 1 Resulting 4 Sets for Example 1

“gun”		“pistol”	“pistol AND gun”		“gun AND pistol”	
accessories,51_	bipods,13	**shooting,25_**	main,217	false,15	**shooting,124_**	hobby,18
products,49	outdoor,13	eagle,20	subdesc,161	micon,15	**rifle,84_**	semi,18
knives,42	gear,13	desert,19	category,143	set,13	**guns,77_**	auto,18
cases,33_	night,13	dep,16	gray,92	image,13	practical,69	foam,18
control,31	clothing,13	links,15	hunting,88	shopping,12	**case,62_**	pictures,17
grips,30	sporting,12	crpa,15	catalog,88	mac,12	pelican,57	dealers,17
daily,29	sectlevel,12	bullseye,12	normal,61	offset,12	club,56	advertise,16
optics,29	**remington,12_**	**ammo,11_**	family,57	**holster,12_**	toy,51	shot,15
holsters,25_	updated,12	safety,10	level,56	settings,12	**bullet,42_**	chelmsford,15
tactical,25	cleaning,11		**airsoft,52_**	items,12	pistols,39	holster,15
systems,24	**bullets,11_**		**air,39_**	**accessories,12_**	ruger,38	blackpelican,15
stocks,23	targets,11		pro,38	match,11	**cases,33_**	custom,14
mounts,21	scopes,11		soft,36	imgcounter,11	cap,31	board,14
rifle,20_	amendment,11		**guns,34_**	**bullet,11_**	shoot,31	guest,14
care,20	sights,11		item,31	effect,11	essex,30	bulletin,14
shooting,19_	uncle,11		maindesc,29	**case,10_**	sport,28	holsters,14
shotgun,17	ak,11		option,20	defined,10	mini,26	book,14
scope,17	equipment,10		icon,19	hand,10	**shotgun,23_**	british,14
reloading,16	goa,10		paintball,18	small,10	trigger,23	save,14
military,16	**guns,10_**		safety,16	addtab,10	firearms,23	industry,13
protection,15	barrels,10		laser,15	safes,10	**ammo,23_**	chat,13
parts,15	handgun,10				target,23	room,13
sight,14	hearing,10				bullets,23	tuition,12
wood,13	calls,10				ukpsa,22	children,12
recoil,13	,				riffles,22	diecast,11
					airsoft,20_	flashlights,11
					vintage,19	accessories,11

Related Words and Frequency	
shooting , 169	air, 50
guns , 124	shotgun,46
rifle, 113	holsters, 46
case, 81	ammo, 37
cases , 74	bullets, 34
accessories, 74	safety, 32
airsoft , 72	holster, 27
products, 68	remington, 22
bullet, 53	

Table 6: Algorithm 1 Discussion - Example 1 - Related word set intersectingWords$_1$ from iteration 1 - Frequency Cutoff - 10

Related Words and Frequency
shooting , 169
guns , 124
cases , 74
airsoft , 72

Table 7: Algorithm 1 Discussion - Example 2 - Related word set intersectingWords$_1$ from iteration 1 - Frequency Cutoff - 20

Related Words and Frequency		
pistols,227	holster,118	tac,79
firearms,205	fits,118	radio,77
accessories,204	shoot,117	paintball,75
free,192	sport,115	assault,71
holsters,172	hours,109	teflon,70
club,170	usa,109	pouch,69
target,164	ammo,107	number,69
tactical,161	electric,107	shoulder,69
air,158	ships,106	leg,64
practical,152	spring,103	core,62
range,150	articles,96	essex,60
court,149	carry,95	nylon,57
uk,147	ruger,93	flash,55
sports,145	force,92	bullets,53
law,143	mp,90	trigger,50
price,142	remote,90	straps,46
full,140	car,89	helicopter,45
control,140	harlow,88	riffles,44
soft,124	magazines,87	coat,44
military,121	belt,86	ukpsa,44
custom,120	mini,82	

Table 8: Algorithm 1 Discussion -Example 2 - Related word set intersectingWords$_2$ from iteration 2 - Frequency Cutoff - 20

3.2 Algorithm 2

One of the biggest drawbacks of Algorithm 1 was the fact that it solely depended on frequency counts to identify sets of related words. This enabled some high frequency, but irrelevant words to be included in the set of related words. Another restriction of Algorithm 1 was that it can take only two search terms as input, and these search terms had to be single words. Finally, the algorithm was also limited to single words as results. Hence, certain 2-word collocations were split into individual words, which was not a desirable result. Collocations are phrases made up of two or more words that often co-occur, and the number of times that they co-occur is more than what would be expected by chance. For example the 2-word phrases, *United States* or *New York* would be both considered as collocations. For example, it would have been ideal to be able to search for *Bill Clinton* and *George Bush* and retrieve a related 2-word collocation such as *White House*, instead of having to search for *Clinton* and *Bush* and retrieve a separate pair of related words such as *white* and *house*. It can also be seen that the search terms *Clinton* and *Bush* are not specific enough to get good results from Google. Hence, Algorithm 2 was a refinement of Algorithm 1, where it accepts more than two terms as input, accepts bigrams as input, returns single words and bigrams as results, and finally, uses a relatedness measure to identify the most related words to the input set.

The new features of Algorithm 2 required some additional parameters as input to the Algorithm. Algorithm 2 takes in as arguments the number of web pages to parse numofPages, the frequency cut off frequencyCutoff, the number of iterations numIterations, the bigram frequency cutoff bigramCutoff, and the score cutoff relatednessCutoff . Bigrams occur at a much less frequency than individual words, hence, Algorithm 2 takes a separate argument for the bigram cutoff. In addition to the bigram cutoff, the algorithm also includes a relatedness measure cutoff which is used to discard words that are above a certain measure. Most of the new features in Algorithm 2 are self explanatory, however, the relatedness measure needs a little bit more attention. The formula for calculating the relatedness between two words is given below,

$$Relatedness(Word_1, Word_2) = \log(hits(Word_1)) + \log(hits(Word_2)) - 2 * \log(hits(Word_1 Word_2)) \quad (2)$$

The relatedness formula is based on the Jiang and Conrath measure [6]. The idea behind this formula is to find the total number of web pages in which Word_1 and Word_2 occur individually, and to subtract from that

total the number of web pages in which Word1 and Word2 occur together. Hence,The following queries to Google gives the required numbers,

- "$Word_1$"
- "$Word_2$"
- "$Word_1$" AND "$Word_2$"

Note that the number of hits returned for the queries "$Word_1$" AND "$Word_2$" and "$Word_2$" AND "$Word_1$" vary slightly, hence, only one combination is used for the calculations.

Hence, the best case for this formula would be if two words, $Word_1$ and $Word_2$ have the same number of hits, and each time they occur, they occur together. These words would then be most related, and would have a score of 0. For example, let hits($Word_1$) = 1000 , hits($Word_2$) = 1000 , and hits($Word_2 Word_2$) = 1000.

The relatedness measure for $Word_1$ and $Word_2$ would be,

Relatedness($Word_1$,$Word_2$) = $\log(1000) + \log(1000) - 2 * \log(1000)$

Relatedness($Word_1$,$Word_2$) = 0

Hence, it can be said that two words $Word_1$ and $Word_2$ tend to be more closely related as the relatedness score approaches 0. As the scores to go towards zero, the number of times the words co-occur tends to increase in relation to the number of times the words occur individually. Therefore, a lower score indicates that the two words are more related. Table 9 gives some example relatedness measures between two words. One problem with the Relatedness measure calculation is that it requires 3 Google queries for each time a measure has to be calculated.

It can be seen from Table 9 that the measure works reasonably well in distinguishing and ordering words that are most related.

In the case of Algorithm 2, the relatedness measure for the word "shooting" with the search terms "gun" and "pistol" would be calculated using the following formula,

Table 9: Algorithm 2 - Example Words and Relatedness Scores

$Word_1$	$Word_2$	Relatedness($Word_1$, $Word_2$)
knife	scissors	18.347
	cut	18.8966
	butcher	25.2893
	weapon	27.842
	murder	32.949
	measure	34.8966
George Bush	Bill Clinton	22.0451
	President	30.1983
	Texas	31.4737
	General	36.4974
	Minnesota	44.4027

= (Relatedness(“shooting”, “gun”) + Relatedness(“shooting”, “pistol”)) / 2

This can be generalized such that, for n search terms, the relatedness score between a particular word and the search terms is the average of the relatedness scores between the word and each search term.

Algorithm 2 essentially follows the same steps as Algorithm 1, however, once it has found a set of related words, Algorithm 2 calculates the relatedness score between each word in the set of related words with the search terms, and discards any word whose relatedness measure is greater than the score cut off relatedness-Cutoff. In addition, the set of related words are also sorted on the relatedness measure. One distinct advantage of using a relatedness measure to rank the set of related words is that high frequency words do not necessarily end up at the top of the list. To illustrate the effectiveness of Algorithm 2, take as an example the same input set used in Example 1 of the Algorithm 1 discussion. Let the algorithm parameters take the following values:

numofPages = 10, numIterations = 1, frequencyCutoff = 10, bigramCutoff = 4 and relatednessCutoff = 30

As you can see from the resulting related set of words in Table 10 that Algorithm 2 is not solely biased towards high frequency words. It can also be seen that reducing the score cutoff relatednessCutoff to around 22, would actually produce a very good set of related words. However, the inclusion of bigrams has also

caused more noise. Take for example the bigram “gun guns”. This is obviously not a good collocation. Hence, this new problem introduces requirement for further refinement.

Now, if we had set the number of iterations numIterations to be 2, the search terms resulting from itertaion 1 would be used as the input set as in Algorithm 1. For more results from Algorithm 2, refer to the section on Experimental Results for Algorithm 2.

Algorithm 2 Algorithm 2

1: **function** Algorithm2 (searchStrings[], numofPages, frequencyCutoff, numIterations, bigramCutoff, relatednessCutoff)
2: $k \leftarrow 1$
3: **for all** $i \leftarrow 1$ to sizeof(searchStrings) **do**
4: $intersectingWords[k][i] \leftarrow searchStrings[i]$
5: **end for**
6: **for all** $k \leftarrow 1$ to numIterations **do**
7: Permute(QuerySet, intersectingWords, k, sizeof(intersectingWords[k]))
8: **for all** $i \leftarrow 1$ to sizeof(QuerySet) **do**
9: $results \leftarrow GoogleSearch(QuerySet[i])$
10: **for all** $j \leftarrow 1$ to R **do**
11: $content \leftarrow content + getWebPageContents(results \rightarrow url[i])$
12: getLinks(linksArray,content)
13: **for all** $n \leftarrow 1$ to sizeof(linksArray) **do**
14: $content \leftarrow content + getWebPageContents(linksArray[n])$
15: **end for**
16: $content \leftarrow removeHTML(content)$
17: $content \leftarrow removeStopWords(content)$
18: **end for**
19: getWordsBiGramsAndFrequency(content: in, frequencyCutoff: in, bigramCutoff: in, wordSetArray[i]: out, wordFrequencyArray[i]: out)
20: **end for**
21: getIntersectingWords(wordSetArray: in, wordFrequencyArray: in, insertSectingWords[k+1][]: out)
22: **end for**
23: **for all** $k \leftarrow 1$ to numIterations **do**
24: calculateRelatednessMeasure(insertSectingWords[k][]: in,searchStrings: in, relatednessCutoff: in, relatedSet[k][]: out)
25: **end for**
26: return resultsArray
27: **end function**

Table 10: Algorithm 2 - Example Words and Relatedness Scores

Related Words from Algorithm 1 (Frequency)	Related Words from Algorithm 2 (Relatedness Score, Frequency)	
shooting , 169	shotgun , 16.40,	50
guns , 124	rifle , 18.01,	112
rifle , 113	holster , 19.31,	26
case , 81	ammo , 19.61,	36
accessories , 74	shooting , 22.21,	171
cases , 74	bullets , 22.80,	32
airsoft , 72	air , 24.88,	44
products , 68	holsters , 25.04,	46
bullet , 53	airsoft , 25.79,	69
air , 50	gun cases , 26.02,	43
shotgun , 46	accessories , 26.99 ,	69
holsters , 46	guns , 28.42,	119
ammo , 37	equipment , 29.32,	27
bullets , 34	remington , 29.37,	22
safety , 32	case , 30.54,	85
holster , 27	gun accessories , 32.52,	10
remington , 22	safety , 34.14,	65
	auto , 34.28,	39
	flashlights , 34.95,	36
	air soft , 36.01,	40
	don hume , 37.77,	8
	cases , 38.73,	114
	gun guns , 39.47,	23

1: **procedure** calculateRelatednessMeasure(insertSectingWords[][],searchStrings,iteration,relatednessCutoff, relatedSet[][])
2: **for all** $i \leftarrow 1$ to sizeof(intersectingWords[iteration][]) **do**
3: $Total \leftarrow 0$
4: **for all** $k \leftarrow 1$ to sizeof(searchStrings[]) **do**
5: $results1 \leftarrow$ GoogleSearch("searchStrings[k]")
6: $results2 \leftarrow$ GoogleSearch("intersectingWords[iteration][i]")
7: $results3 \leftarrow$ GoogleSearch("intersectingWords[iteration][i] AND searchStrings[k]")
8: $Score \leftarrow log(results1 \rightarrow hitcount) + log(results2 \rightarrow hitcount) - 2 * log(results3 \rightarrow hitcount)$
9: $Total \leftarrow$ Total + Score
10: **end for**
11: $Average \leftarrow$ Total/ sizeof(searchStrings[])
12: **if** $Average < relatednessCutoff$ **then**
13: $relatedSet[iteration][\text{"}intersectingWords[iteration][i]''] \leftarrow$ Average
14: **end if**
15: **end for**
16: **end procedure**

3.3 Algorithm 3

Though the relatedness score approach in Algorithm 2 helped identify and discard noisy or irrelevant terms, the introduction of 2-word collocations as results introduced a new problem. Some of the bigrams returned by Google Hack were not actually collocations. Collocations are phrases made up of two or more words that often co-occur, and the number of times that they co-occur is more than what would be expected by chance. For example the 2-word phrases, *United States* or *New York* would be both considered as collocations. Once again, consider the example input set *gun* and *pistol*. Table 18 of the section Experimental Results shows the set of words returned by Google Hack for the above input set. One of the bigrams in the related set of words is *guns guns*. Obviously, this is not a good collocation.

Consider another example, where one of the bigrams returned for the query *maytag* and *kenmore* was *ge jenn*. This bigram is the combination of the brand names *ge* and *jenn air*. This is obviously not a collocation. The reason why this particular bigram was not discarded by the relevance measure is because *ge* and *jenn air* are both brand names of appliance manufacturers. Hence, there is a relevance between the search terms and the result, however, the result as a collocation is not a good one. In this particular example the bigram was somewhat relevant. However, in many cases the bigrams returned are quite bad. Examples of such bigrams include *forecast text* and *bulletin fcpn* that are returned as results for the input set *sunny* and *cloudy*.

To solve this problem, there was a need for some sort of measure to indicate if a bigram is a collocation or not. Hence, the log likelihood measure was slightly modified to adapt to the WWW, and was used to identify words that occur more often than expected by chance. The basic idea was to use the number of hits returned for a particular bigram as a means of calculating the log likelihood score. Hence, if the bigram to be validated is "Word_1 Word_2", the following queries are issued to Google, and the respective hit counts are retrieved.

- "of the"
- "Word_1 *"
- "* Word_2"
- "Word_1 Word_2"

Table 11: Algorithm 3 - Calculation of Observed Values

	$Word_2$	Not $Word_2$	
$Word_1$	n11	n12 = n1p-n11	n1p
Not $Word_1$	n21 = np1 - n11	n22 = n2p - n21	n2p = npp - n1p
	np1	np2 = npp - np1	npp

Table 12: Algorithm 3 - Calculation of Expected Values

	$Word_2$	Not $Word_2$
$Word_1$	e11 =(n1p*np1/npp)	e12 =(n1p*np2/npp)
Not $Word_1$	e21 =(np1*n2p/npp)	e21 =(np2*n2p/npp)

The number of hits returned for the query "of the" serves as the sample size npp. The assumption here is that "of the" must be one of the, if not, the most common bigram occuring on the web. Also note that the queries are issued with quotes surrounding the search terms. This ensures that Google searches for the particular string as a phrase. To get the hit count for the number of times $Word_1$ occured as the first word of a bigram, the wild card operator "*" was used. Hence, the query "$Word_1$ *" should ideally return the number of times $Word_1$ occured as the first word in a phrase. Let this value be n1p. Similarly, to find the number of times $Word_2$ occured as the second word of a 2-word phrase, the query "* $Word_2$" was issued to Google. Let this value be np1. Finally, the query "$Word_1$ $Word_2$" was issued to find out the number of hits for this particular phrase. Let this value be n11.

Given the hit counts for these particular queries, the values in Table 11 can be calculated by some simple algebra. The values calculated here are the observed values for each cell in Tabl 11. Hence, the value in cell n11 of Table 11 represents the observed value for the number of times the bigram "$Word_1$ $Word_2$" occurred, whereas n12 represents the number of times $Word_1$ occurred as the first word of a bigram where the second word of the bigram was not $Word_2$.

Once the values for each cell in table has been calculated, the next step is to calculate the expected values for each cell. The expected values are calculated as shown in Table 12, where e11, e12, e21 and e22 are the respective expected values for each cell. The expected value signifies the idea of independence between two words, where the probability of two words occurring together is equal to the product of the probability of the words occurring individually.

Finally, the log likelihood measure can be calculated for the bigram "$Word_1$ $Word_2$". The use of the log likelihood measure to predict if two or more words occurred by chance was introduced by [4]. In this case, the log likelihood score tries to identify if two words occurred together by chance, and are independent, or if they occur more often than expected by chance. The higher the log likelihood score, the less likely that the two words occurred together by chance. However, there are some important cases to note here. The number of hits for the phrase "$Word_1$ $Word_2$", in this case the value in cell n11 has to be greater than 0. If the value is 0, then this particular bigram is not a collocation. The next case would be, if the observed value for the phrase "$Word_1$ $Word_2$" is less than the expected value, that is if n11 is less than e11, then this particular bigram is also not a collocation. Hence, for a particular bigram to be considered as a bad collocation, one of the above conditions must be met. The results are first sorted by the relatedness score, and then the log likelihood score.

Also note that the log likelihood score is calculated only for those bigrams which appear in the final set of related words.

Algorithm 3 remains similar to Algorithm 2 except for the change in the calculateRelatedness function, which now calls the calculateLoglikelihoodScore function if a related term is a bigram before calculating the relatedness measure. Example bigrams and log likelihood scores are given in Table 3.

Algorithm 3 Algorithm 3

procedure calculateRelatednessMeasure(resultsArray[][],insertSectingWords[][],searchStrings,iteration,cutoff)
for all $i \leftarrow 1$ to sizeof(intersectingWords[iteration][]) **do**
  $flag \leftarrow 0$
  if isCollocation("intersectingWords[iteration][i]") **then**
    $llscore \leftarrow$calculateLogliklihoodScore("intersectingWords[iteration][i]")
    if llscore == 0 **then**
      continue
    end if
  end if
  $Total \leftarrow 0$
  for all $k \leftarrow 1$ to sizeof(searchStrings[]) **do**
    $results1 \leftarrow$ GoogleSearch("searchStrings[k]")
    $results2 \leftarrow$ GoogleSearch("intersectingWords[iteration][i]")
    $results3 \leftarrow$ GoogleSearch("intersectingWords[iteration][i] AND searchStrings[k]")
    $Score \leftarrow log(results1 \rightarrow hitcount) + log(results2 \rightarrow hitcount) - 2 * log(results3 \rightarrow hitcount)$
    $Total \leftarrow$ Total + Score
  end for
  if $(results1 \rightarrow hitcount = 0)$ OR $(results2 \rightarrow hitcount = 0)$ OR $(results3 \rightarrow hitcount = 0)$ **then**
    continue
  end if
  $Average \leftarrow$ Total/ sizeof(searchStrings[])
  if $Average < cutoff$ **then**
    $resultsArray[iteration][\text{“}intersectingWords[iteration][i]''] \leftarrow$ Average
  end if
end for
end procedure

1: **procedure** calculateLogliklihoodScore(bigram)
2: $word1$ ←getFirstWord(bigram)
3: $word2$ ←getSecondWord(bigram)
4: $result1$ ← GoogleSearch(“bigram”)
5: $n11$ ← result1 → hitcount
6: **if** $n11 == 0$ **then**
7: return 0;
8: **end if**
9: $result$ ← GoogleSearch(“of the”)
10: npp ← result → hitcount
11: $result2$ ← GoogleSearch(“word1 *”)
12: $n1p$ ← result2 → hitcount
13: $result3$ ← GoogleSearch(“* word2”)
14: $np1$ ← result3 → hitcount
15: $n2p$ ← npp - n1p
16: $np2$ ← npp - np1
17: $n12$ ← n1p - n11
18: $n21$ ← np1 - n11
19: $n22$ ← n2p - n21
20: $e11$ ← (n1p * np1) / npp
21: $e12$ ← (n1p * np2) / npp
22: $e21$ ← (n2p * np1) / npp
23: $e22$ ← (n2p * np2) / npp

Table 13: Algorithm 3 - Example Bigrams and log liklihood scores

Bigram	log likelihood score	n11	e11	Discarded
new york	50395555	63100000	40851146	No
star wars	26776472	6990000	638077	No
george bush	612272	1360000	664288	No
atomic bomb	503856	230000	34156	No
star gazing	12182	39100	21620	No
maytag kenmore	1321	520	60	No
atomic nuclear	0	13500	135873	Yes
appliance parts	0	69300	104550	Yes
fork back	0	1000	456490	Yes
chocolate good	0	2060	404516	Yes
running there	0	27100		Yes

3.4 Sentiment Classification Algorithm

The final addition to this thesis is a Sentiment Classification Algorithm that tries show the value of sets of related words. This algorithm is simply an extension of the Point wise Mutual Information -Information Retrieval (PMI-IR) algorithm described in [10]. The purpose of the PMI-IR algorithm was to predict if a particular phrase has an overall positive or negative orientation. For example, the phrase *excellent movie* has a positive semantic orientation, and the phrase *horrible movie* has a negative semantic orientation or sentiment. This algorithm is then used to classify if a review has an overall positive or negative recommendation. These reviews can be for automobiles, movies, banks and even travel destinations. Certain 2-word phrases are extracted from a review and the PMI-IR algorithm is used to predict the sentiment of each phrase by using information retrieved from the WWW. If the sentiment of the extracted phrases lean toward a positive sentiment, the review is classified as being a positive review, and negative otherwise.

The first step in the PMI-IR algorithm is to run a part-of-speech (POS) tagger to identify words in the review as being nouns, adjectives or verbs. The POS tagger used in both the PMI-IR algorithm and the Sentiment Classification Algorithm of Google Hack is the Brill Tagger [10]. Once the text has been POS tagged, certain two word phrases are extracted from the tagged text. There are particular combinations and rules for

1: **if** $n11 < e11$ **then**
2: return 0
3: **end if**
4: $llscore \leftarrow 0$
5: **if** $n11$ **then**
6: $llscore \leftarrow$ llscore + (n11 * log (n11 / e11))
7: **end if**
8: **if** $n12$ **then**
9: $llscore \leftarrow$ llscore + (n12 * log (n12 / e12))
10: **end if**
11: **if** $n21$ **then**
12: $llscore \leftarrow$ llscore + (n21 * log (n21 / e21))
13: **end if**
14: **if** $n22$ **then**
15: $llscore \leftarrow$ llscore + (n22 * log (n22 / e22))
16: **end if**
17: $llscore \leftarrow$ llscore * 2
18: return llscore
19: **end procedure**

extracting phrases. For example, a phrase which contains a noun as the first word and second word is always extracted. The idea here is to extract phrases which reveal the most about the Semantic Orientation of the review. Table 14 identifies the different possible combinations of words that require a phrase to be extracted from a review. The rules to extract phrases in the Sentiment Classification Algorithm are the same as the set of rules used in the PMI-IR algorithm. The PMI-IR algorithm then queries the search engine AltaVista with the combination of positive and negative words with each of these phrases to find the Semantic Orientation (SO) of that particular phrase. The SO measure is calculated as follows:

$$SO(\textit{phrase}) = \log_2 \frac{hits(\textit{phrase}\ NEAR\ \text{“}excellent''))\ hits(\text{“}poor'')}{hits(\textit{phrase}\ NEAR\ \text{“}poor'')\ hits(\text{“}excellent'')} \tag{3}$$

In the PMI-IR algorithm, the words “excellent” and “poor” were used as the positive and negative connotations. So, $hits(\textit{phrase} NEAR \text{“}excellent'')$ is the number of web pages in which the phrase occurred near the word “excellent”, and $hits(\textit{phrase} NEAR \text{“}poor'')$ is the number of web pages in which the phrase occurred near the word “poor’ . The value of hits(“poor”) and hits(“excellent”) are the number of hits returned for the queries “poor” and “excellent” respectively. Therefore, a phrase is positively oriented if it is often associated with the positive word “excellent”, and negatively oriented otherwise. A positive SO value signifies that the phrase is positively oriented, and a negative measure signifies that the phrase is negatively oriented.

The results from the PMI-IR algorithm are quite good. However, the ambiguity seen when using a polyseme such as “poor”, which at minimum has two senses, “poor” as in “poverty” and “poor” as in “bad” in turn affects the SO of a phrase. For example, the word “Hawaii” has a negative sentiment because the “poverty” sense of “poor” creates a negative association with the word.

To address this problem, Turney and Litman proposed the approach of using multiple pairs of psoitive and negative connotations. The Sentiment Classification Algorithm extends this approach to use Google instead of AltaVista, and the Semantic Orientation of a particular phrase is calculated using the following two equations:

$$Total\ SO(\textit{phrase}) = \sum_{i=1}^{n} \log_2 \frac{hits(\textit{phrase}\ AND\ \text{“}positiveWord_i'')\ hits(\text{“}negativeWord_i'')}{hits(\textit{phrase}\ AND\ \text{“}negativeWord_i'')\ hits(\text{“}positiveWord_i'')} \tag{4}$$

$$SO(\textit{phrase}) = \frac{Total\ SO}{n} \tag{5}$$

As it can be seen from the equations, instead of using a single pair of positive and negative connotations as in (Turney 2002), the Sentiment Classification Algorithm uses a set of positive and negative words as suggested in (Turney and Littman 2003) [11]. Hence, if a particular positive or negative word takes on the wrong sense when used as a search term in a search engine query, its effects can be reduced by weighting the overall SO over multiple pairs of positive and negative words. The most important difference between the PMI-IR algorithm and the Sentiment Classification Algorithm is the use of the "NEAR"operator in the PMI-IR algorithm, and the use of the "AND" operator in the Sentiment Classification Algorithm. Google does not provide the "NEAR" feature that was provided by AltaVista. Google also states that the use of the "AND" operator is not necessary, however, the "AND" operator is used here as means of clarifying the query strings.

Phrases and their Semantic Orientation values are given in Table 15 for a movie review which was classified as being a negative review by Sentiment Classification Algorithm. In this particular example, the words {{*excellent, bad*}, {*great, awful* }} were used as the positive and negative pairs of words. Table 16 gives a better idea of the difference between using single pairs of positive and negative connotations to multiple pairs of positive and negative connotations. The review used for Table 16 is a positively oriented review for an automobile. In this particular example, the words {{*excellent, bad*}, {*great, awful* }} were used as the positive and negative pairs of words for the second column of results, and the words {*excellent, bad*} were used for the second column of results. The Sentiment Classification Algorithm classified the automobile review as *positive* when used with the multiple pairs of positive and negative words, and it classified the review as negative with a single pair of positive and negative words.

Eventually, the goal is to use the set of related words predicted by Google Hack as seeds to the Sentiment Classification Algorithm.

The Sentiment Classification Algorithm is given in the following pages.

Table 14: Sentiment Classification Algorithm Discussion - Patterns of tags

	First Word	Second Word	Third Word (Not Extracted)
1.	JJ	NN or NNS	anything
2.	RB, RBR, or RBS	JJ	not NN or NNS
3.	JJ	JJ	not NN or NNS
4.	NN or NNS	JJ	not NN or NNS
5.	RB, RBR, or RBS	VB, VBD, VBN, or VBG	anything

NN - Noun, VB - Verb, and JJ - Adjective (Turney 2002)

Table 15: Sentiment Classification Algorithm - Negative Movie Review

Phrase	Semantic Orientation(phrase) {{*excellent, bad*}, {*great, awful* }}
”disconcerting elements”	-0.2468
”accidental death”	-0.2773
”numbing plot”	-0.9334
”poorly written”	-1.1067
”brutal killing”	-1.5943
”lukewarm reception”	-1.8782
”despicable movie”	-1.9241
”hapless victims”	-2.0050
”embarrassing lines”	-2.0792
”bad wardrobe”	-2.3723

Table 16: Sentiment Classification Algorithm - A Positive Car Review with Sentiment Classification Algorithm

Phrase	Semantic Orientation(phrase) {{*excellent, bad*}, {*great, awful* }}	Semantic Orientation(phrase) {*excellent, bad*}	{great, awful }
”4Matic model”	13.7910	0.1217	27.49
”high performance”	2.1544	1.50	2.72
”standard features”	1.7077	1.1799	1.97
”refined styling”	1.2271	1.18	0.87
”full size”	0.8893	0.3960	0.89
”classical guitar”	0.7539	0.0159	1.31
”high speeds”	0.5058	-0.0390	0.63
”new standard”	0.4136	-2.0948	0.52
”quick maneuvers”	0.2963	0.6811	-0.06
”sophisticated user”	0.1704	-0.15	0.50
”luxurious interior”	-0.0052	1.1882	-1.19
”extraordinary comfort”	-0.027	0.13	-0.10
”very intuitive”	-0.0633	0.1965	-0.41
”electronic glitches”	-0.0633	-0.6154	-1.69
”technical sophistication”	-0.7824	-0 .02	-1.45
”persistent fault”	-0.9381	-0.95	-0.86
”electronically limited”	-1.0142	-0.5941	-1.40
”back seat”	-1.2352	-0.8376	-1.69

Algorithm 4 Sentiment Classification Algorithm

```
1: function predictSemanticOrientation(review, posConnotations[], negConnotations[])
2: numPos ← sizeof(posConnotations)
3: numNeg ← sizeof(negConnotations)
4: if numPos! = numNeg then
5:     return
6: end if
7: taggedText ← runBrillTagger(review)
8: getPhrases(taggedText:in, phrases[]: out)
9: for all i ← 1 to numPos do
10:    result1 ← GoogleSearch("posConnotations[i]")
11:    posCounts[i] ← result1 → hitcount
12:    result2 ← GoogleSearch("negConnotations[i]")
13:    negCounts[i] ← result2 → hitcount
14: end for
15: totalSO ← 0
16: for all i ← 1 to sizeof(phrases) do
17:    tempSO ← 0
18:    for all k ← 1 to numPos do
19:       result1 ← GoogleSearch("phrases[i] AND posConnotations[k]")
20:       pos ← result1 → hitcount
21:       result2 ← GoogleSearch("phrases[i] AND negConnotations[k]")
22:       neg ← result2 → hitcount
23:       score ← (pos * negCounts[k]) / ( neg * posCounts[k])
24:       if score > 0 then
25:          tempSO ← tempSO + log(score)
26:       end if
27:    end for
28:    totalSO ← totalSO + (tempSO / numPos)
29: end for
30: return totalSO/sizeof(phrases)
31: end function
```

4 Experimental Results

4.1 Evaluation Methods

Algorithms 1,2 and 3 were evaluated by using as the gold standard the sets of related words predicted by Google Sets and the Human Subject experiments. A description of the two sources used to evaluate the Algorithms are given in sections 4.1.1, and 4.1.2. One of the problems in evaluating the results of Google-Hack was the notion of a related set of words. Hence, the Algorithms were evaluated by using the results from Google Sets and the results from the Human Subject experiments as a gold standard.

4.1.1 Human Subject Experiments

The Human Subject Experiments were conducted by prompting human subjects to expand a given set of words. As mentioned earlier, the idea of a related set of words varied amongst different sources. This was also reflected in the results attained through the Human Subject experiments. For example, when one human subject was asked to expand the initial set {bush, clinton }, the human subject predicted the words *carter, reagan, ford, nixon, johnson, kennedy* as being in the same set as the words bush and clinton. However, when another human subject was prompted with the same initial set of words, the set of words predicted by that human subject was *Parliament, President, United States, Senate, war, peace, United Nations, September 11, India, Iraq, Administration* . The first set clearly contains the last names of past American presidents. This seems quite intuitive given the fact that the input set contains the last names of two American presidents. However, the second set is also quite good. The relationship assumed here is not very clear, however, the words in the set are often associated with the Presidents of the United States. A third subject entered the following words *Elections, Democrats, Republicans ,Second Term, Presidents, White House*. It can be seen here that the set of words predicted by this human subject also contains words that do have a relation with presidents of the United States. These examples illustrate the fact that the notion of a set of related words can be defined in a very diverse manner.

4.1.2 Google Sets

Google Sets is a tool developed by Google that tries to predict sets of related words. However, it is not known if Google Sets uses information from the WWW or a standard static corpora of text. In fact, the results returned by Google Sets over time for the same set does not change drastically, suggesting that Google must be using a more standard form of text. Google Sets has two main features. The first feature allows the user to retrieve a smaller, more tightly related set of 15 words or less. The second feature allows the user to retrieve a larger more loosely related set of words. Almost all the experiments conducted for the purpose of evaluation uses the small set feature feature of Google Sets. The large set feature was used only when Google Sets was unable to predict a smaller set of related words. In contrast to the manual experiments, Google Sets was most often able to identify a very clear and intuitive relationship between the input terms. Google Sets works very well for Proper Nouns in particular. For example, when Google Sets was prompted with the initial set {bush, clinton} it returned *reagan, nixon, ford, eisenhowever, kennedy, johnson* as the expanded small set. The set clearly contains the last names of past American presidents. However, when Google Sets was prompted with the initial set {jordan, chicago} it returned the results *Israel, JOHNSON, Jackson, Kuwait, JANESVILLE, Iraq, Japan, Lebanon, Egypt, Springfield* . The relationship assumed here is very vague, but the terms returned are either cities from the United States or countries from the Middle East region. The only exception being Japan. However, for most fans of the sport Basketball, the set {jordan, chicago} clearly represents Micheal Jordan the basketball player of the NBA basketball team, Chicago Bulls.

Google Sets can be accessed by going to the following link http://labs.google.com/sets.

4.2 Experimental Results of Algorithm 1

The sets of related words predicted by Algorithm 1 are given in Table 18 to Table 28. The caption at the top of each table gives the input set for which the experiments were run. The columns with the heading GoogleHack(numofPages,frequencyCutoff,numIterations) contain the sets of related words predicted by Google Hack, where numofPages, frequencyCutoff and numIterations are the number of web pages that were parsed, the frequency cutoff that was used, and the number of iterations respectively. Each table tries to emphasize a different type of set of related words, where the input set contains either nouns, proper nouns, or adjectives. The experiments try to express the strengths and weaknesses of Google Hack, and the ad-

vantages and disadvantages of using the WWW as a source of information. Each word in the Google Hack column is accompanied by its frequency of occurrence in the web pages. Hence, the results in this column are ranked by the highest frequency to the lowest. The column with the heading Google Sets contains the results returned by Google Sets for the given input set. The order of the words in the Google Sets columns are represented exactly the way the results were returned by Google Sets. The final column with the heading Human Subject Experiments contains the set of related words that were retrieved from the Human Subject experiments. The words in that column are sorted by alphabetical order.

Also note that Algorithm 1 accepts only single words as input terms. Furthermore, Algorithm 1 takes in only two terms as input.

4.3 Experimental Results of Algorithm 1

The set of related words predicted by Google Hack, Google Sets, and the Human Subject experiments for the initial set S_1 = {gun, pistol} is given in Table 19. The first two columns of Table 19 represent the performance of Algorithm 1 over different parameters. In both columns, the algorithm parsed only the top 10 web pages and the pages linked to by the top 10 pages. However, the frequency cut off used in the first run was 10, and the algorithm was run for 1 iteration. The words in the second column represent the set of words predicted by Algorithm 1 when it was executed with a frequency cutoff of 15 and an iteration value of 2. As expected, the number of words returned drastically increased from iteration 1 to iteration 2. The results in the first column seem to be quite good. The only words that seem to be out of place are the words *accessories,72* and *products,68*. The high frequency counts for both *accessories* and *products* could be explained by the fact that Google is a very commercialized search engine, and a search for words such as *gun* and *pistol* retrieve web pages which sell the two items. Table 16 gives the URLs returned by Google for the query “gun”. It can be seen from the table that, links 1,3 and 4 are links to web sites that sell guns and gun accessories.

Table 29 gives the Precision, Recall and the F-measure for all the results from Google Hack for the initial input sets S_1 - S_9 in relation to the gold standard. The Precision, Recall and F-measure for the sets were calculated as follows:

Table 17: Algorithm 1 Experimental Results - Resulting Top 10 Web Pages for query ”gun”

Web page #	Set of Links for query “gun”
1.	$http://www.thesmokinggun.com/$
2.	$http://www.gunbroker.com/$
3.	$http://www.gunowners.org/$
4.	$http://www.doublegun.com/$
5.	$http://www.ithacagun.com/$
6.	$http://www.imdb.com/title/tt0092099/$
7.	$http://www.gunandgame.com/$
8.	$http://www.gunaccessories.com/$
9.	$http://www.bradycampaign.org/$
10.	$http://www.guncite.com/$

Table 18: Algorithm 1 Experimental Results - Resulting Web Pages for query ”red”

Web page #	Set of Links for query “red”
1.	$http://www.redhat.com/$
2.	$http://www.redcross.org/$
3.	$http://www.redherring.com/$
4.	$http://www.redmeat.com/$
5.	$http://www.redbull.com/$

$$Precision = \frac{\#\ Words\ common\ to\ Google\ Hack\ and\ Google\ Sets}{\#\ Words\ Returned\ By\ Google\ Hack} \tag{6}$$

$$Recall = \frac{\#\ Words\ common\ to\ Google\ Hack\ and\ Google\ Sets}{\#\ Words\ Returned\ By\ Google\ Sets} \tag{7}$$

$$Fmeasure = \frac{2\ *\ (Precision\ *\ Recall)}{Precision\ +\ Recall} \tag{8}$$

It can be seen that the type of words predicted by Google Hack closely matches the type of words in the manual experiments. However, in this particular example, the results of Algorithm 1 look similar to the type

of results returned by Google Sets. Google Sets returned an empty set when it was asked to return a small set for *gun* and *pistol*.

Though the type of words predicted by Google Hack seem similar to the words in Google Sets and the Human Subject experiments, overall, the f-measure is very low. There are many reasons for this. Firstly, words such as *ammo, paintball, airsoft, bullets* that are related words do not appear in either Google Sets or the Human Subject experiment results. In addition, some of the words returned by Google Sets such as *Trainer, Binocular, Shop Set, Carbine, Bayonet, Kit Mod, Mount, Telescope, Storage Racks, and Gage* seem to be very irrelevant and noisy. This in turn affects the recall for Google Hack, which in turn affects overall F-measure for Algorithm 1. The second iteration of Google Hack produced more noisy terms than relevant terms. This emphasizes the fact that raw frequency counts alone would not be sufficient to predict if a word is relevant or not to the given search terms.

One other interesting thing to note when comparing the results from Google Hack to the results in the gold standard is the difference due to the singular and plural forms of words. For example, a word such as *bullets* returned by Google Hack is not in the set of related words in the Human Subject experiments, however, the Human Subject experiment does contain the singular form of the word *bullet* in its set of related words. Another point to note would be Algorithm 1 is restricted to single words as results. However, a quick glance at both the gold standard experimental results reveal that a related word can also be a collocation. This would possibly be one of the improvements for Algorithm 2.

The next input set, $S_2 = \{\text{bush, clinton}\}$ tries to show the dynamic nature of Google Hack. The underlying relationship assumed here is that these are the last names of American presidents. The first column represents the experiments conducted the 9th of March 2005. At the time of those experiments, the news around the world was focused on matters such as relief and aid to the people and countries effected by the tsunami. Furthermore, President George Bush and President Bill Clinton had been trying to aid with the relief as well. The words such as tsunami, world, disaster, relief and village that were returned by Google Hack reflect the period in which the experiments were conducted. However, certain words such as president, presidents, george are obviously words that are related to the words bush, clinton. Another interesting fact to note here are the words sri and lanka. This is obviously Sri Lanka, one of the South Asian countries that was affected by the recent tsunami disaster in December 2004.

Comparing these results to the words returned by Google Hack (with the same parameters) on May 6th

2005, the results changed by quite a bit. In additional to the results changing, there are more noisy terms than before. However, words such as democratic, election, republican and republicans seem to match with the set of related words in the Human Subject Experiments column.

Once again, in relation to Google Sets, the set of words predicted by Google Hack is very different. Google Sets gives a more specific set, with only relation identifying all the terms in the set. All the terms are last names of presidents of America. However, given the nature of the web, and the particular approach taken to this problem of identifying sets of related words, the words predicted by Google Hack seem reasonable for a baseline approach.

Another interesting example to look at would be S_4 ={jordan, chicago}. The results given by Google Hack and Google Sets is very different for these two terms. The relationship assumed by Google Hack should be very clear for any fan of Basketball. Google Hack correctly identified jordan as being Michael Jordan, and Chicago as being the city in which Micheal Jordan played basketball. When conducting this particular experiment, the relationship assumed was the above mentioned relationship. When the same initial set was input to Google Sets, the resulting set of words were names of cities and countries around the world. The cities were from the United States, and the countries in the list were from the Middle East. This can be explained from the fact that Jordan is a country in the Middle East, and Chicago is a city in the United States of America.

Close examination of the set of words returned by Google Hack for a frequency cutoff of 5, which is the first column in Table 23 reveals what seems to be a very nice set of related words. The first two words in the set, Michael, and bulls, are the first name of the basketball player Jordan, and the name of the team that Michael Jordan used to play for. There are 4 terms in the first list photos, picture,type, and time that seem to be irrelevant. The second column of Table 23 represents the set of words returned for a slightly higher frequency cutoff. All the words in the set seem to be very intuitive and good.

The next set of results to look at would be S_6 ={yellow, red}. This is one of the harder set to expand. The two main reasons being, the words in the set are 1) adjectives, and 2) not specific. A search for the terms red returns the URLs given in Table 18. As it can be scene from the URL's in Table 10, the resulting web pages seem to be very diverse. This is one of the problems in using a commercial search engine such as Google. The results returned are targeted towards commercial products. Hence, it can be seen through Table 24 that the set of words predicted by Google Hack is very noisy except for the two terms, black and blue.

However, a slight increase of the frequency cutoff to 20 results in the set of words in column 2 of Table 25. The unfortunate result here is that the two related words black and blue have been discarded.

The results returned by Google Sets for the input set, {yellow,red} is extremely good. The set contains other colors. The sets of words in the Human Subject Experiments closely match the results in Google Sets, however, it also contains some words that are not as closely related, such as flower, frequency, pattern etc. This words are often associated with colors, and the word spectrum which is at the bottom of the list also makes sense.

The final set to be examined is S_7 ={bank, money} (Table 26). Google Hack performed reasonably well for this input set. However, once again, in comparison to Google Sets, the f-measure seems very low. However, closer introspection of the set of words predicted by Google Hack seems to have returned some important words that were missed by both Google Sets, and the Human Subject Experiments. In this particular example, Google Sets seemed to have performed poorly in comparison to previous results. Other than the word Invest, all the other words seem to be quite irrelevant. On the contrary Google Hack returned some really interesting words such as, account, services, funds, banking, currency, credit and cards. It can also be seen that the actual collocation credit cards, has been split into credit and cards. The frequency of the two words also seem to be close enough, that is credit occurred 39 times, and cards occurred 38 times. This also seems to strongly suggest the inclusion of bigrams in results.

Table 19: Algorithm 1 Results - S_1 = {gun, pistol}

GoogleHack(10,10,1)	GoogleHack(10,15,2)			Google Sets (Small Set)	Human Subject Experiments
03/09/2005	03/09/2005			05/16/2005	01/01/2005
shooting , 169	pistols,227	holster,118	tac,79	gun	air
guns , 124	firearms,205	fits,118	radio,77	pistol	ak47
rifle , 113	accessories,204	shoot,117	paintball,75	ordinance	arm
case , 81	free, 192	sport,115	assault,71	heavy weapon	arms
accessories , 74	holsters,172	hours,109	teflon,70	artillery	automatic
cases , 74	club,170	usa,109	pouch,69	machine gun	barrel
airsoft , 72	target,164	ammo,107	number,69	launcher	Bomb
products , 68	tactical,161	electric,107	shoulder,69	shotgun	bullet
bullet , 53	air,158	ships,106	leg,64	rifle	bull's eye
air , 50	practical,152	spring,103	core,62	shooting iron	case
shotgun , 46	range,150	articles,96	essex,60	handgun	colt
holsters , 46	court,149	carry,95	nylon,57	side arm	fire
ammo , 37	uk,147	ruger,93	flash,55	target	fire arms
bullets , 34	sports,145	force,92	bullets,53	munitions	firearm
safety , 32	law,143	mp,90	trigger,50	repeating firearm	hand
holster , 27	price,142	remote,90	straps,46	rocket launcher	handgun
remington , 22	full,140	car,89	helicopter,45	arsenal	holster
	control,140	harlow,88	riffles,44	22	Knives
	soft,124	magazines,87	coat,44	scatter gun	machine gun
	military,121	belt,86	ukpsa,44	armory	muzzle
	custom,120	mini,82		repeater	Police

Table 20: S_1 = {gun, pistol} Continued

GoogleHack(10,10,1) 03/09/2005	GoogleHack(10,15,2) 03/09/2005	Google Sets (Large Set) 05/16/2005	Human Subject Experiments 01/01/2005
		torpedo	recoil
		armoury	revolver
		broadside	rifle
		twenty two	safe
		muzzle loader	semi-automatic
		auto loader	shoot
		ordinance stores	Terrorism
		Trainer	thirty-eight
		Binocular	trajectory
		Shop Set	Violence
		Submachine gun	
		Carbine	
		Bayonet	
		Kit Mod	
		Mount	
		Telescope	
		Storage Racks	
		Gage	

Table 21: Algorithm 1 Results - S_2 = {bush, clinton}

GoogleHack(10,10,1) 03/09/2005	GoogleHack(10,10,1) 05/06/2005	Google Sets (Large Set) 05/06/2005	Human Subject Experiments 01/01/2005
tsunami , 521	president , 1578	reagan	adams
world , 315	security , 749	nixon	Administration
mr , 195	house , 550	ford	carter
sri , 195	national , 473	eisenhower	Deomocrats
people , 172	people , 294	kennedy	Elections
lanka , 160	april , 224	johnson	ford
president , 144	american , 217		India
children , 139	apr , 166		Iraq
disaster , 136	bill , 143		jefferson
aceh , 122	time , 136		johnson
tour , 101	relief , 135		kennedy
men , 87	women , 132		lincoln
relief , 86	ve , 129		nixon
thai , 77	support , 123		Parliament
survivors , 74	don , 116		peace
presidents , 65	pm , 115		president
george , 65	edt , 114		Presidents
affected , 58	good , 112		reagan
village , 57	george , 94		republicans
	election , 92		Second Term
	money , 71		Senate
	presidential , 69		United Nations
	vote , 68		United States
	political , 67		war
	democratic , 66		washington
	republican , 66		White house
	senator , 46		

Table 22: Algorithm 1 Results - S_3 = {toyota, ford}

GoogleHack(10,5,1) 03/09/2005	GoogleHack(10,10,1) 03/09/2005	Google Sets (Small Set) 05/06/2005	Human Subject Experiments 01/01/2005
truck , 66	car, 61	nissan	benz
car , 61	sales, 59	honda	buick
sales , 59	cars, 35	mazda	car
parts , 46	auto .31	subaru	chevrolet
vehicles , 45		mitsubishi	chrysler
year , 43		dodge	dodge
cars , 35		chevrolet	fast
auto , 32		jeep	ferrari
motors , 30		volvo	gasoline
general , 27		buick	general motors
company , 24		pontiac	honda
honda , 20		suzuki	jeep
service , 20		holden	mercedes
automotive , 18		mitsubishi	
nissan , 18			
trucks , 17			
consumer , 17			
detroit , 13			
marketing , 13			
volvo , 12			
media , 12			
buyers , 12			
focus , 11			

Table 23: Algorithm 1 Results - S_4 = {jordan,chicago }

GoogleHack(10,5,1) 05/06/2005	GoogleHack(10,10,1) 05/06/2005	Google Sets (Small Set) 05/06/2005
michael , 173	michael , 174	Chicago
bulls , 148	bulls , 148	Jordan
nba , 97	nba , 97	Israel
game , 56	game , 56	JOHNSON
type , 52	jersey , 43	Jackson
team , 46		Kuwait
player , 43		JANESVILLE
jersey , 43		Iraq
time , 35		Japan
points , 30		Lebanon
basketball , 26		Egypt
photos , 25		Springfield
picture , 22		

Table 24: Algorithm 1 Results -S_5 = {kenmore, maytag }

GoogleHack(10,15,1) 05/06/2005	GoogleHack(10,20,1) 05/06/2005	Google Sets (Small Set) 05/06/2005
service , 82	ge , 109	Kenmore
appliance , 75	service , 82	Maytag
appliances , 69	appliance , 75	Whirlpool
volt , 60	appliances , 69	Frigidaire
dishwasher , 54	volt , 60	GE
air , 50	amana , 46	Amana
amana , 46		KitchenAid
electric , 43		Inglis
kitchen , 38		HOTPOINT
general , 37		Westinghouse
frigidaire , 36		Norge
		Speed Queen
		Magic Chef
		General Electric
		Gen Electric

Table 25: Algorithm 1 Results -S_6 = {yellow,red}

GoogleHack(10,15,1) 05/06/2005	GoogleHack(10,20,1) 05/06/2005	Google Sets (Small Set) 05/06/2005	Human Subject Experiments 01/01/2005
enterprise , 411	enterprise , 406	Yellow	black
software , 257	management , 137	Red	blue
solutions , 151	system , 94	Green	colors
management , 142	snake , 70	Blue	cyan
technology , 141	applications , 67	White	flower
system , 96	scarlet , 62	Black	frequency
services , 89	touch , 46	Orange	green
netherlands , 84	snakes , 38	magenta	grey
fellow , 76	florida , 36	cyan	leaves
applications , 71	bands , 34	Gray	light
snake , 70		Purple	orange
performance , 64		Browser	pink
scarlet , 62			pattern
design , 52			spectrum
model , 47			
service , 47			
touch , 46			
work , 46			
tools , 44			
read , 41			
order , 39			
includes , 38			
snakes , 38			
florida , 36			
bands , 34			
coral , 28			
black , 28			
blue , 27			

Table 26: Algorithm 1 Results -S_7 = {bank,money}

GoogleHack(10,10,1) 05/07/2005	GoogleHack(20,10,1) 05/07/2005	Google Sets (Small Set) 05/07/2005	Human Subject Experiments 01/01/2005
currency , 170	state , 312	People	cash
collect , 114	state , 312	Shopping	Cashier
note , 89	npaper , 216	Maps	cent
financial , 83	world , 182	Freebies	check
member , 73	currency , 172	Education	checking account
selection , 70	select , 170	Invest	commercial
banking , 70	financial , 120	Insure	Credit
funds , 66	collect , 117	Coupons	Credit Card
services , 62	features , 117	Contests	credit union
provide , 62	note , 96		currency
account , 55	banking , 95		deposit
canada , 44	services , 90		dollar
market , 43	policy , 87		draft
accounts , 39	years , 86		exchange
credit , 39	privacy , 86		finances
loans , 38	member , 83		loan
banks , 38	account , 82		mortgage
cards , 38	funds , 71		national
business , 35	selection , 71		overdraft
military , 28	credit , 71		penny
check , 27	rate , 69		rupee
service , 26	provide , 69		safe
enc , 20	loans , 65		savings account
	accounts , 62		savings and loan
	list , 59		sensex

Table 27: Algorithm 1 Results -S_8 = {bank,money} Continued

GoogleHack(10,10,1) 05/06/2005	GoogleHack(20,10,1) 05/06/2005	Google Sets (Small Set) 01/01/2005	Human Subject Experiments
	education , 56		sterling
	business , 55		stock
	banks , 54		Teller
	cards , 54		trade
	loan , 49		withdraw
	market , 48		
	canada , 47		
	exchange , 47		
	savings , 46		
	check , 43		
	national , 42		
	rates , 42		
	indexof , 42		
	navigator , 39		
	card , 38		
	american , 37		
	federal , 35		
	center , 35		
	service , 33		
	gold , 33		
	books , 32		
	military , 28		
	annual , 28		
	checking , 27		
	status , 26		
	enc , 20		

Table 28: Algorithm 1 Results -S_9 = {buddhism,islam}

GoogleHack(10,35,1) 05/07/2005	Google Sets (Small Set) 05/07/2005
god , 490	Buddhism
life , 328	Islam
al , 268	Christianity
qur , 267	Hinduism
buddhist , 266	Judaism
religion , 257	
people , 246	
islamic , 207	
muslim , 202	
world , 192	
buddha , 142	
allah , 128	
prophet , 126	
human , 109	

Table 29: Algorithm 1 Results - Precision, Recall and F-measure

Input Set	Google Sets (Small Set)			Human Subject Experiments		
	P	R	F-Measure	P	R	F-measure
S_1 = {gun, pistol} GoogleHack(10,10,1)	2/17	2/39	0.06	6/17	6/31	0.24
S_2 = {bush, clinton} GoogleHack(10,10,1) 01/09/2005	0	0	0	2/19	2/26	0.08
S_2 = {bush, clinton} GoogleHack(10,10,1) 05/06/2005	0	0	0	4/29	4/26	0.14
S_3 = {toyota, ford} GoogleHack(10,5,1)	3/23	3/14	0.16	2/23	2/13	0.104
S_4 = {jordan, chicago} GoogleHack(10,5,1)	0	0	0	NA	NA	NA
S_5 = {kenmore, maytag} GoogleHack(10,15,1)	3/11	3/13	0.24	NA	NA	NA
S_6 = {yellow, red} GoogleHack(10,15,1)	2/31	2/10	0.09	2/31	2/14	0.084
S_7 = {bank, money} GoogleHack(10,10,1)	0	0	0	4/25	4/30	0.14
S_8 = {buddhism, islam} GoogleHack(10,35,1)	0	0	0	NA	NA	NA

4.4 Experimental Results of Algorithm 2

The sets of results from Algorithm 2 were evaluated similar to the sets of results from Algorithm 1. The basic idea was to compare the results of the Google Hack algorithms to the results from Google Sets and the Human Subject Experiments. However, since Algorithm 2 can take in more than two words as input, certain sets predicted by Algorithm 2 were compared only to the results from Google Sets. Overall, Algorithm 2 performed much better than Algorithm 1. The relatedness measure approach used in Algorithm 2 helped discard much of the noisy terms. Note that each word in the Google Hack column contains a relatedness score as well. Some of the sets were compared to the Small Set, and Large Set feature of Google Sets.

The first example input set is $S_1 = \{\text{kenmore, maytag}\}$. The first column of Table 30 shows the set of words predicted by Algorithm 2 over a frequency cutoff of 10, and bi-gram cutoff of 4, a score cutoff of 30, and for 1 iteration for the given input set. The resulting set of words are quite good. All the words in the set are brand names of appliance manufacturers. The results also show that the inclusion of bi-grams has resulted in the bi-gram '*jenn air* being included in the set of related words predicted by Google Hack. The precision, recall and f-measure for all th example are given in Table 48. The second and third column of Table 30 represent the results for running Algorithm 2 over 2 iterations, and a frequency cutoff of 15, a relatedness score.cutoff of 30. Bi-grams were not considered for this particular example. The first iteration results in *whirlpool* and *ge*. The terms *frigidaire* and *kitchenaid* which were par to the related set in the first column have been discarded due to the higher frequency cutoff used in column 2. A higher frequency was used show.

As expected, iteration was able to get one addition brand, *frigidaire* in its set of related words. This illustrates the point that increasing the number of iterations can result in a better set of related words than having just one iteration. However, in most cases, increasing the number of iterations result in more noise.

Table 31, shows the set of related words predicted by Algorithm 2 for the input set $S_2 = \{\text{toyota, ford}\}$ with a frequency cutoff of 10, and a relatedness score cutoff of 30. Any term with frequency less than 10, or any term with a relatedness score greater than 30 is automatically discarded. Big rams were not considered as part of the expanded set for this particular example. Algorithm 2 returned *gm, nissan and car* as the set of related words to the input terms *toyota, ford*. Though the small set predicted by Google Sets does not contain any of the terms predicted by Google Hack, it is very clear that the terms predicted by Google Hack are in

fact related. *gm* and *nissan* are manufacturers of automobiles. The word *car* is in the set of words predicted by the Human Subject Experiments. Though the precision of Google Hack for this particular example is quite good, the recall is very poor. The number of words returned by Google Hack seems too less. Hence, to expand the set further, the term *nissan* was added to the initial input set. This exploits Algorithm 2's new feature which allows more than two terms as input. The resulting set of words predicted by Google Hack is given in the first two columns of Table 32. The score cutoff used in the first column was30, whereas the score cutoff used in the second column of Table 32 was 40. This slight variation was intentional, since it shows the kind of words that were discarded by the relatedness score, words which would have otherwise been included in the set of related words as they pass the frequency cutoff phase. The set predicted by Google Hack for this input set is extremely good. All the terms in the set are names of automobile manufacturers, in-fact only the last term in column 1, which is *vehicles* seems to be out of place. However, even *vehicles* can be considered as a related word to S_3 = {toyota, ford, nissan} since all the input terms are names of manufacturers of *vehicles*. Once again, some of the terms predicted by Algorithm 2 do not exist in the set predicted by Google Sets, but it is clearly not incorrect. Furthermore, the terms in column 2 that had a relatedness score greater than 30 are still words that are related to automobile manufacturers. At first glance, the only word that seems to be out of place in the set is *sheehy*, however, upon further investigation, *sheehy* was found to be an automobile dealer. However, a score cutoff of 30 reveals a much tight set of related words.

The number of terms predicted by Google Hack for S_3 = {toyota, ford, nissan} is much greater than the number of terms predicted by Google Hack for S_2 = {toyota, ford}. This is justifiable result, since the number of queries to Google increases from 4 queries to 9 queries when moving from an initial set of 2 words to an initial set of 3 words. In turn 50 more web pages are parsed for text. As a result, the set of words predicted by Google Hack drastically increases. The quality of the terms in the resulting set is also quite good. This is because, as the number of web pages parsed increases, a clearer relationship can be identified within the input terms.

Another interesting result that is worth looking at in Table 32 is that, the top two terms returned by Google Hack are *mazda, 19.59* and *honda, 19.92*. The top two terms in the set predicted by Google Sets is also *honda* and *mazda*. This same result can also be noticed in some of the other result tables.

The next input set to be looked is S_4 = {january,february,may}. Table 33 shows the set of words predicted

by Google Hack for this set. The set of words predicted by Google for this particular input set is very good. In fact, the precision and recall for this particular set is 1. Google Hack was clearly able to identify that *january, february* and *may* are all months of there year, and this is reflected in the set of words returned by Algorithm 2. The terms that were discarded by Google Hack, which had a relatedness score greater than 30 were *friday, wednesday , thursday , work , people , bush , year , president , american , house , northern* and *saddam*. Other than *friday, wednesday , thursday* and *year*, the other words with a relatedness measure greater than 30 look very random. Hence, this shows that the relatedness measure is quite effective in pruning out noisy terms.

The next set to be considered is $S_6 = \{\text{red, yellow}\}$, which is one of th most difficult set of input terms. This is mainly due to the non-specific nature of the terms. Algorithm 1 performed very poorly for this input set, with the most relevant terms, *blue* and *black* appearing at the bottom of the set of related words (Table 25 in Algorithm 1 Experimental Results). However, with the relatedness measure approach, the words *blue* and *black* have been pushed to the top of the set, and with a score cutoff of 30, almost all non-relevant terms have been discarded. This can be seen by comparing the results in column 1 to the results inc column 2 of Table 35. Column 1 also shows the words *scarlet* and *coral* as being related to *red* and *yellow*. This is correct, since *scarlet* and *coral*are colors. Some of the words discarded by the relatedness measure include *power, call, includes* and *work* etc. These are obviously noisy and irrelevant terms. Once again, the precision and recall for this input set is low. This illustrates the fact that Google Hack performs significantly better with specific, and topic focused nouns such as brand names, and performs rather poorly with more general terms like adjectives that are used to describe nouns.

The next set to be examined is $S_9 = \{\text{artificial intelligence, machine learning}\}$. This set illustrates Algorithm 2's new feature which allows bi-grams to be used as input terms to Google Hack. Overall, the precision, recall and f-measure calculated for Google Hack over a score cutoff of 40 seems low. However,. examination of the set reveals a lot of really good terms that are missed by Google Sets. For example, the terms *knowledge discovery* and *reinforcement learning* are obviously terms that are relevant to *artificial intelligence* and *machine learning* where *reinforcement learning* is a machine learning technique. This particular example illustrates two more deficiencies of Google Hack. Words such as *genetic, algorithms, reasoning, expert, neural* and *natural* are words that were obviously part of a 2-word or 3-word collocation. For example, *expert* from *expert systems*, and *natural* from *natural language processing*. The latter example shows the

second deficiency. Since Google Hack considers only unigrams and bi-grams, terms such as *cased based reasoning* and *natural language processing* are split into either unigrams or bi-grams or both. The same type of problem is once again revealed in Table 41, for the input set S_{10} = {versace,armani}. Words such as *dior* and *hugo* are obviously from the collocations *christian dior* and *hugo boss* respectively. Overall, this results in a lower precision and recall for Google Hack.

The next input set S_{11} = {sunny,cloudy} reveals a new problem that did not exist in Algorithm 1. This problem arose due to the inclusion of bi-grams in the Google Hack expanded sets. column 2 of Table 43 contains the set of words predicted by Google Hack for S_{11} = {sunny,cloudy}. The set contains 4 bi-grams, of which, two of the bi-grams *partly cloudy* and *partly sunny* are good collocations, whereas *forecast text* and *bulletin fpcn* make no sense as collocations. Hence, a possible refinement to Algorithm 2 might be the inclusion a function to verify if a particular bi-gram is a good collocation or not.

One final point worthy of discussion is the low precision and recall for Google Hack when comparing to Google Sets in general. Take for example Table 32, where some of the words returned by Google Hack, specifically *bmw,lexus, gmc* are names of automobile brands. However, these brand names are not included in the set returned by Google Sets or the Human Subject Experiments. Now, this arises the question as to whether the set of related words returned by Google Hack can be compared to a set of gold standard files. However, for the purpose of this thesis research, the results are compared to Google Sets, and the Human Subject Experiments.

Table 30: Algorithm 2 Results - S_1 = {kenmore, maytag}

GoogleHack(10,10,4,1,30)	GoogleHack(10,15,0,2,30) Iteration 1	GoogleHack(10,15,0,2,30) Iteration 2	Google Sets (Small Set)
whirlpool , 16.04 frigidaire , 19.60 ge , 21.67 jenn air , 22.37 kitchenaid , 23.70	whirlpool 17.18 ge , 23.38	frigidaire,20.54	Whirlpool Frigidaire GE Amana KitchenAid Inglis Hotpoint Westinghouse Norge Speed Queen Magic Chef General Electric Econo Wash

Table 31: Algorithm 2 Results - S_2 = {toyota, ford}

GoogleHack(10,10,0,1,30) 05/25/2005	Google Sets (Small Set) 05/25/2005	Human Subject Experiments 05/25/2005
gm , 19.09	HONDA	benz
nissan , 20.15	MAZDA	buick
car , 29.77	SUBARU	car
	MITSUBISHI	chevrolet
	DODGE	chrysler
	CHEVROLET	dodge
	Jeep	fast
	Volvo	ferrari
	Buick	gasoline
	Pontiac	general motors
	Suzuki	honda
	Holden	jeep
		mercedes

Table 32: Algorithm 2 Results - S_3 = {toyota, ford, nissan}

GoogleHack(10,10,0,1,30) 05/25/2005	GoogleHack(10,10,0,1,40) 05/25/2005	Google Sets (Small Set) 05/25/2005
mazda , 19.59	mazda , 19.59	HONDA
honda , 19.92	honda , 19.92	MAZDA
chevrolet , 21.37	chevrolet , 21.37	SUBARU
bmw , 22.47	bmw , 22.47	MITSUBISHI
dodge , 22.83	dodge , 22.83	DODGE
lexus , 23.05	lexus , 23.05	CHEVROLET
mitsubishi , 23.17	mitsubishi , 23.17	Jeep
pontiac , 23.89	pontiac , 23.89	Volvo
mercedes , 24.56	mercedes , 24.56	Buick
gmc , 25.14	gmc , 25.14	Pontiac
vehicles , 27.77	vehicles , 27.77	Suzuki
	truck , 30.42	Holden
	car , 30.71	
	parts , 30.73	
	cars , 31.75	
	auto , 31.84	
	suv , 32.86	
	dealer , 33.74	
	performance , 34.64	
	prius , 34.99	
	vehicle , 36.74	
	sheehy , 36.93	
	sales , 37.38	
	rear , 37.58	
	coupe , 38.18	
	sedan , 39.56	

Table 33: Algorithm 2 Results - S_4 = {january,february,may}

GoogleHack(10,15,0,1,30)	Google Sets (Small Set)
june , 22.90 , 215	March
july , 24.39 , 176	April
august , 25.33 , 212	June
september , 25.50 , 260	October
march , 25.71 , 236	November
october , 26.21 , 192	December
november , 27.09 , 214	September
april , 27.49 , 243	July
december , 27.61 , 263	August

Table 34: Algorithm 2 Results - S_5 = {george bush, bill clinton, ronald reagan}

GoogleHack(10,10,0,1,40)		GoogleHack(10,10,5,1,30)	Google Sets (Small Set)
john , 25.89	federal , 36.49	jimmy carter , 21.95	Jimmy Carter
democratic , 26.49	people , 36.89	john , 25.92	John F Kennedy
republican , 27.50	anti , 37.00	democratic , 26.59	Gerald Ford
congress , 27.63	white , 37.10	vice president , 27.04	Richard Nixon
william , 28.06	civil , 37.24	republican , 27.83	George W Bush
president , 28.21	south , 37.45	president , 28.33	Ross Perot
james , 28.65	war , 37.71		Herbert Hoover
republicans , 29.20	government , 37.74		Hillary Clinton
soviet , 30.61	political , 37.92		Richard M Nixon
democrats , 32.13	great , 38.75		Harry Truman
america , 32.22	today , 38.18		Lyndon Johnson
presidential , 32.59	state , 38.82		
iraq , 33.52	durbin , 39.11		
nixon , 33.72	world , 39.23		
house , 34.08	carter , 39.37		
american , 35.04	jefferson , 39.45		
freedom , 35.25	washington , 39.50		
united , 35.73	elected , 39.67		
national , 35.90	ended , 39.67		
supreme , 36.22	good , 39.74		
won , 36.33	election , 39.77		

Table 35: Algorithm 2 Results - S_6 = {red, yellow}

GoogleHack(10,10,0,1,30) 05/22/2005	GoogleHack(10,10,0,1,40) 05/22/2005	Google Sets (Small Set) 05/22/2005	Human Subject Experiments 05/22/2005
blue , 16.77	blue , 16.77	Red	black
black , 17.07	black , 17.07	Green	blue
scarlet , 24.91	scarlet , 24.91	Blue	colors
coral , 28.97	coral , 28.97	White	cyan
	power , 32.79	Black	flower
	call , 33.58	Orange	frequency
	includes , 35.45	magenta	green
	read , 35.67	cyan	green
	kill , 35.85	Gray	grey
	work , 37.21	Purple	leaves
	kingsnake , 37.59	Browser	light
	touch , 39.36		orange
	post , 39.50		pink
	snake , 39.83		spectrum

Table 36: Algorithm 2 Results - S_7 = {bank, money}

GoogleHack(10,10,5,1,30) 05/22/2005	GoogleHack(10,10,5,1,40) 05/22/2005	Google Sets (Small Set) 05/22/2005	Human Subject Experiments 05/22/2005
banking , 21.16	banking , 21.16	Bank	cash
credit , 21.62	credit , 21.62	Money	Cashier
financial , 25.15	financial , 25.15	People	cent
credit cards , 25.79	credit cards , 25.79	Shopping	check
currency , 26.58	currency , 26.58	Maps	checking account
loans , 28.81	loans , 28.81	Freebies	commercial
business , 29.40	business , 29.40	Education	Credit
	account , 31.47	Invest	Credit Card
	canada , 31.55	Insure	credit union
	accounts , 32.14	Coupons	currency
	services , 33.47	Contests	deposit
	note , 34.68		dollar
	msn money , 38.58		draft
	center , 39.84		exchange
			finances
			loan
			mortgage
			national
			overdraft
			penny
			rupee
			safe
			savings account
			savings and loan
			sensex

Table 37: Algorithm 2 Results - S_8 = {buddhism, islam}

GoogleHack(10,15,5,1,30) 05/22/2005	GoogleHack(10,15,5,1,40) 05/22/2005	Google Sets (Small Set) 05/22/2005
christianity , 14.89	christianity , 14.89	Buddhism
buddha , 21.44	buddha , 21.44	Islam
muslims , 21.48	muslims , 21.48	Christianity
tibetan , 21.79	tibetan , 21.79	Hinduism
islamic , 23.21	islamic , 23.21	Judaism
buddhist , 23.68	buddhist , 23.68	
muslim , 23.74	muslim , 23.74	
buddhists , 24.00	buddhists , 24.00	
god , 25.70	god , 25.70	
religion , 25.99	religion , 25.99	
women , 26.10	women , 26.10	
religions , 26.23	religions , 26.23	
peace , 26.50	peace , 26.50	
human , 26.59	human , 26.59	
jesus , 27.35	jesus , 27.35	
death , 27.45	death , 27.45	
muhammad , 28.87	muhammad , 28.87	
teachings , 28.93	teachings , 28.93	
life , 29.29	life , 29.29	
quran , 29.32	quran , 29.32	
worship , 29.38	worship , 29.38	
allah , 29.43	allah , 29.43	
jihad , 29.72	jihad , 29.72	
	world , 30.06	
	india , 30.16	
	faith , 30.22	
	qur , 30.37	
	prophet , 30.62	
	love , 31.03	

Table 38: Algorithm 2 Results - S_8 = {buddhism, islam} Continued

GoogleHack(10,15,5,1,30) 05/22/2005	GoogleHack(10,15,5,1,40) 05/22/2005	Google Sets (Small Set) 05/22/2005
	west , 31.38 history , 31.48 living , 31.78 people , 31.89 give , 32.00 central asia , 32.40 called , 32.51 al , 32.72 belief , 32.73 followers , 32.74 holy , 32.77 earth , 32.82 buddhism islam , 32.87 central , 32.91 man , 33.05 law , 33.24 basic , 33.62 al insan , 33.82 insan al , 33.82 dalai lama , 33.97 means , 34.39 good , 34.53 beliefs , 34.76 al kamil , 34.88 children , 35.10 mind , 35.83 truth , 35.98 great , 36.21 human beings , 36.98 guide , 37.22 books , 37.44	

Table 39: Algorithm 2 Results - S_9 = {artificial intelligence, machine learning}

GoogleHack(10,15,0,1,30) 05/22/2005	GoogleHack(10,15,5,1,40) 05/22/2005	Google Sets (Large Set) 05/22/2005
robotics , 21.14	neural networks , 20.88 , 25	Neural Networks
neural , 21.60	robotics , 21.14	Robotics
expert , 24.24	neural , 21.60	Knowledge Representation
reasoning , 24.40	data mining , 22.84	Natural Language Processing
intelligent , 25.68	expert systems , 22.90	Pattern Recognition
knowledge , 25.89	expert , 24.24	Machine Vision
logic , 26.18	genetic algorithms , 24.30	Programming Languages
data , 26.21	reasoning , 24.40	Data Mining
natural , 26.23	logic programming , 24.40	Genetic Programming
genetic , 26.33	natural language , 24.87	Vision
applications , 26.60	intelligent , 25.68	Natural Language
computer , 27.91	knowledge , 25.89	Intelligent Agents
ai , 29.16	logic , 26.18	People
language , 31.44	data , 26.21	Publications
software , 31.64	natural , 26.23	Philosophy
programming , 32.01	genetic , 26.33	Qualitative Physics
game , 32.22	applications , 26.60	Speech Processing
algorithms , 32.44	computer , 27.91	Expert Systems
reinforcement , 32.68	knowledge discovery , 28.91	Genetic Algorithms
lisp , 33.57	ai , 29.16	Computer Vision
prolog , 34.05	case based , 29.83	Computational Linguistics
agents , 34.22	computer science , 30.21	Cognitive Science
aaai , 35.06	reinforcement learning , 31.17	Fuzzy Logic

Table 40: Algorithm 2 Results - S_9 = {artificial intelligence, machine learning} Continued

GoogleHack(10,15,0,1,30) 05/22/2005	GoogleHack(10,15,5,1,40) 05/22/2005	Google Sets (Large Set) 05/22/2005
research , 35.44	language , 31.44	Knowledge Acquisition
games , 36.38	software , 31.64	Artificial Life
systems , 36.47 ,	programming , 32.01	Knowledge Based Systems
recognition , 37.05	game , 32.22	Case Based Reasoning
analysis , 37.08	algorithms , 32.44	Image processing
discovery , 38.13	reinforcement , 32.68	Software engineering
networks , 38.56	lisp , 33.57	Planning and Scheduling
system , 38.62	prolog , 34.05	Information Retrieval
machines , 39.01	agents , 34.22	Virtual Reality
science , 39.05	research , 35.44	Automated Reasoning
methods , 39.15	ai programming , 36.19	Speech recognition
iit , 39.585	games , 36.38	Knowledge Engineering
java , 39.67	systems , 36.47	Model based Reasoning
	discovery , 38.13	Speech
	ai magazine , 38.22	Agents
	system , 38.62	Dynamical systems
	morgan kaufmann , 38.87	Fuzzy Systems
	machines , 39.01	Expert system
	science , 39.05	ALGORITHMS
	methods , 39.15	Computer Graphics
	iit , 39.58	Complexity Theory
	learning systems , 39.62	Fuzzy Sets
	java , 39.67	Logic Programming
		AI Magazine
		Neural Nets

Table 41: Algorithm 2 Results - S_{10} = {versace,armani}

GoogleHack(10,15,0,1,30)	GoogleHack(10,15,5,1,40)	Google Sets (Small Set)	Google Sets (Large Set)
prada , 18.17	prada , 18.17	GUCCI	Gucci
moschino , 18.45	moschino , 18.45	Ferragamo	Chanel
gucci , 18.60	gucci , 18.60	Chanel	Calvin Klein
dkny , 19.00	dkny , 19.00	Valentino	Prada
valentino , 19.72	valentino , 19.72 ,	Prada	Dolce Gabbana
chanel , 19.93	chanel , 19.93	Biagiotti	Fendi
gianni , 20.12	gianni , 20.12	Calvin Klein	Hugo Boss
calvin , 21.97	hugo boss , 20.17	Hugo Boss	Christian Dior
dior , 22.37	calvin klein , 20.29	Dolce Gabbana	Hermes
yves , 22.62	gianni versace , 20.46		Moschino
hugo , 23.06	dolce gabbana , 21.76		Donna Karan
fendi , 24.12	calvin , 21.97		Ralph Lauren
giorgio , 24.64	yves saint , 22.10		Valentino
boss , 30.00	dior , 22.37		Louis Vuitton
	yves , 22.62		Giorgio Armani
	giorgio armani , 23.04		DKNY
	hugo , 23.06		Escada
	fendi , 24.12		Tommy Hilfiger

Table 42: Algorithm 2 Results - S_{10} = {versace,armani} Continued

GoogleHack(10,15,0,1,30)	GoogleHack(10,15,5,1,40)	Google Sets (Small Set)	Google Sets (Large Set)
	giorgio , 24.64 christian dior , 24.86		Tiffany Givenchy Cartier Ferragamo Vivienne Westwood Elizabeth Arden Estee Lauder Miu Miu Salvatore Ferragamo Guess Carolina Herrera Pucci Joop Coach DIESEL Dana Buchman Cour Carre Bally David Hayes Bernini Agnes b ADIDAS Davidoff Aramis OAKLEY

Table 43: Algorithm 2 Results - S_{11} = {sunny,cloudy}

GoogleHack(10,10,0,1,40)	GoogleHack(10,15,4,1,30)	Google Sets (Small Set)
clear , 24.35 , 68	clear , 24.35 , 38	partly cloudy
partly , 25.88 , 87	partly cloudy , 25.85 , 56	Sleet
light , 27.33 , 166	forecast text	forecast , 26.66 , 11
showers , 28.62 , 303	partly sunny , 26.92 , 21	Freezing Rain
wind , 28.84 , 209	light , 27.33 , 128	Dew Point
cloud , 29.11 , 61	bulletin fpcn , 28.33 , 11	Hail
winds , 29.22 , 121	wind , 28.84 , 115	Heavy Snow
low , 30.11 , 308	winds , 29.22 , 96	Clear
moderate , 30.77 , 164		Weather Warning
midnight , 31.23 , 43		Partly Sunny
high , 31.30 , 621		windy
fog , 31.36 , 52		Foggy
fpcn , 31.90 , 40		rainy
upper , 32.38 , 35		
tonight , 33.29 , 240		
clearing , 33.94 , 32		
chance , 34.00 , 291		
lower , 34.40 , 33		
clouds , 34.67 , 52		
today , 34.80 , 283		

Table 44: Algorithm 2 Results - S_{12} = {atomic, nuclear}

GoogleHack(10,15,4,1,40)	Google Sets (Small Set)	Human Subject Experiments
nuclear weapons , 29.28 , 35	Particle	Bohr
energy , 31.06 , 117	Condensed Matter	bomb
bomb , 33.61 , 48	High Energy	bond
weapons , 34.25 , 45	Astrophysics	charge
nuclear weapon , 34.92 , 16	Astronomy	chemistry
nuclear power , 35.09 , 49	Undecided	Chernobyl
nuclear capabilities , 36.89 , 6	Biomedical	compound
atomic bomb , 38.57 , 28	Optics	covalent
atomic energy , 37.54 , 15	Mathematical	Destruction
nuclear arsenal , 39.48 , 6	environment	detonation
robert oppenheimer , 39.50 , 11	chemistry	diffusion
nuclear science , 39.72 , 10		Einstein
		electron
		element
		energy
		Explosion
		facility
		Generator
		Hiroshima
		holocaust
		interference
		ion
		light
		mass
		Max Planc
		molecule
		mushroom cloud
		Nagasaki
		neutron
		nucleus

Table 45: Algorithm 2 Results - S_{12} = {atomic, nuclear} Continued

GoogleHack(10,15,4,1,40)	Google Sets (Small Set)	Human Subject Experiments
		number
		orbit
		orbital
		particle
		periodic
		photoelectric
		physics
		polarization
		Power
		power plant
		quantum
		radiation
		ratio
		shell
		spectrum
		speed
		submarine
		table
		testing
		theory
		waste
		wave
		weapon
		weight

Table 46: Algorithm 2 Results - S_{13} = {passport, tickets}

GoogleHack(10,10,0,1,40)	GoogleHack(10,10,3,1,40)	Google Sets (Small Set)	Human Subject Experiments
travel , 23.18 , 84	travel , 23.18 , 84	credit cards	visa airplane
ticket , 24.96 , 57	ticket , 24.96 , 57	Wallet	travel
passports , 27.69 , 51	passports , 27.69 , 51	itinerary	visa
details , 31.76 , 55	concert tickets , 31.00 , 14	Money	airplane
buy , 32.25 , 35	details , 31.76 , 55	Phone Numbers	air
concerts , 32.79 , 31	buy , 32.25 , 35	Reservations	aeroplane
nba , 33.20 , 29	las vegas , 32.57 , 8	Vouchers	flight
services , 34.10 , 32	concerts , 32.79 , 31	travelers checks	security
college , 34.15 , 34	nba , 33.20 , 29	Vaccination Certificate	charge
nfl , 34.90 , 27	macys passport , 33.56 , 31	Travel Insurance	aerodrome
racing , 35.50 , 34	services , 34.10 , 32	traveler’s checks	hostess
includes , 35.82 , 29	college , 34.15 , 34	Medication	stewart
sports , 36.28 , 150	nfl , 34.90 , 27		money
day , 36.29 , 98	gift certificates , 35.35 , 11		dollars
service , 37.07 , 163	racing , 35.50 , 34		age
city , 37.21 , 24	sports , 36.28 , 150		information
football , 37.56 , 63	day , 36.29 , 98		photograph

Table 47: Algorithm 2 Results - S_{13} = {passport, tickets} Continued

GoogleHack(10,10,0,1,40)	GoogleHack(10,10,3,1,40)	Google Sets (Small Set)	Human Subject Experiments
	auto racing , 36.72 , 15		signature
	nba basketball , 36.98 , 12		flight
	service , 37.07 , 163		travel
	city , 37.21 , 24		immigration
	football , 37.56 , 63		customs
	horse racing , 38.05 , 15		forms
			layovers
			baggage
			carry ons
			stewards
			Visa
			Aeroplane
			Luggage
			Airport
			Foreign
			Vacation
			Packing
			Holidays

Table 48: Algorithm 2 Results - Precision, Recall and F-measure

Input Set	Google Sets (Small Set)			Human Subject Experiments		
	P	R	F-Measure	P	R	F-measure
S_1 = {kenmore, maytag} GoogleHack(10,10,4,1,30)	4/5	4/13	0.43	NA	NA	NA
S_2 = {toyota, ford} GoogleHack(10,10,0,1,30)	0	0	0	2/3	2/13	0.24
S_3 = {toyota, ford, nissan} GoogleHack(10,10,0,1,30)	6/11	6/12	0.51	NA	NA	NA
S_4 = {january, february, may} GoogleHack(10,15,4,1,30)	9/9	9/9	1	NA	NA	NA
S_5 = {george bush, bill clinton, ronald reagan} GoogleHack(10,10,5,1,30)	1/6	1/11	0.106	NA	NA	NA
S_6 = {red, yellow} GoogleHack(10,10,4,1,30)	2/4	2/11	0.265	2/4	2/14	0.218
S_7 = {bank, money} GoogleHack(10,10,4,1,30)	0	0	0	4/7	4/25	0.24
S_8 = {buddhism, islam} GoogleHack(10,10,4,1,30)	1/23	1/3	.06	NA	NA	NA
S_9 = {artificial intelligence, machine learning} GoogleHack(10,15,5,1,30)	7/21	7/48	0.19	NA	NA	NA
S_{10} = {versace, armani} GoogleHack(10,15,5,1,40)	7/18	7/9	0.5	NA	NA	NA
S_{11} = {sunny, cloudy} GoogleHack(10,15,4,1,30)	4/8	4/12	0.39	NA	NA	NA
S_{12} = {atomic,nuclear} GoogleHack(10,15,4,1,40)	0	0	0	2/12	2/61	0.05
S_{13} = {passport, tickets} GoogleHack(10,10,3,1,40)	0	0	0	1/17	1/33	0.0375

4.5 Experimental Results of Algorithm 3

Unlike the Algorithm 1 and Algorithm 2 experimental results discussion sections, this particular section focuses on one particular feature of Algorithm 3. The only difference between Algorithm 2 and Algorithm 3 is the function that tries to identify if a particular bi-gram is a bad collocation. To do this, 5 input sets from the Algorithm 2 experiments were chosen, as the results for these input sets contained bi-grams. Specifically, some of the bi-grams that were included in the set of related words were in fact bad collocations.

Take for example the input set $S_1 = \{\text{sunny, cloudy}\}$. The actual set of related words predicted by Algorithm 3 are given in Table 49. Two particularly noisy terms *forecast text* and *bulletin fpcn* that were part of the set of related words predicted by Algorithm 2 (See Table 43 of Algorithm 2 Experimental Results Section), do not appear in the set predicted by Algorithm 3. All the input parameters were the same except for the word frequency cutoff, which does not effect the bi-grams that are considered. Table 50 shows all the bi-grams that were considered by Algorithm 3 before calculating their respective Log Likelihood scores. The first column in the table represents the bi-gram that was to be verified, the second column of Table 50 gives the number of hits returned by Google for that particular bi-gram, the third column gives the expected value for the given bi-gram, and finally, the fourth column gives the log likelihood score. Note that if for a particular bi-gram the observed value is 0, that is, if the number of hits for a particular bi-gram is 0, then the log likelihood score is also 0. Also note that if the observed value for a particular bi-gram is less than the expected value, the log likelihood score is assumed to be 0. In both these cases, the bi-grams are discarded from the results before the relatedness measure is calculated.

According to this implementation of the log likelihood score, only 3 bi-grams were considered as not being a bad collocation. They are *partly cloudy, canada word mark,* and *severe thunderstorm*. Note that the number of hits returned for the bi-gram *bulletin fpcn* is 0. Also note that the observed value for *forecast text* is much less than the expected value 1698201.05. Hence, this bi-gram was also discarded. Overall, out of other discarded bi-grams the only two questionable discards were *meteorlogical service* and *weather service*.

The next example set is $S_2 = \{\text{kenmore, maytag}\}$. The set of words predicted by Algorithm 3 are given in Table 51. The bi-grams and their respective log likelihood scores are given in Table 52. The results here seem to be a little bit suspect. A possibly valid big-gram such as magic chef was not considered because the observed value for *magic chef* which was 35100, was less than it's expected value of 39025.57. The

Table 49: Algorithm 3 Results - S_1 = {sunny, cloudy}

GoogleHack(10,30,4,1,30)
partly cloudy , 26.06
showers , 28.80
cloud , 29.30

difference between the two numbers is considerably small, especially with respect to the World Wide Web. Hence, this particular bi-gram could be considered as a borderline collocations.

The third set to be considered is S_3 = {artificial intelligence}. The set of words predicted by Algorithm 3 are given in Table 55. Table 56 shows the bi-grams and their respective log likelihood scores. The bi-grams that were identified as not being bad collocations are actually very good collocations. However, some valid collocations have been identified by Algorithm 3 as bad collocations. In addition, some of the bi-grams that were discarded by Algorithm 3 through the log likelihood score were part of the large set of related words predicted by Google Sets for S_3 = {artificial intelligence}. For example, collocations such as *ai magazine, expert systems* and *logic programming* all failed the test where the observed value is less than the expected value.

The next set to be considered is S_4 = {atomic, nuclear}. Algorithm 3 performed fairly well in identifying bi-grams, however, once again, some of the discarded bi-grams are possible collocations where the observed value is less than the expected value. For example, *energy commission, full story* and *nuclear power*.

The final set to be considered is S_5 = {bank, money}. The bi-gram results for this particular input set is quite good. Two suspect collocations *msn money* and *paper money* were discarded as bad collocations. The bi-gram *credit cards* was identified as being a good collocation.

In conclusion, the log likelihood function does a decent job with identifying bad collocations, however, it needs some fine tuning since it also discards some valid and good collocations.

Table 50: Algorithm 3 Results - S_1 = {sunny, cloudy} Bi-grams

Bi-gram	Observed (“word1 word2”)	Expected (“word1 word2”)	Log Liklihood Score
partly cloudy	808000	14400	5475034.57
canada wordmark	146000	16507.93	571649.16
severe thunderstorm	46500	5045.92	128838.18
additional weather	25500	4642857.14	0
bulletin fpcn	0	0	0
canada weather	112000	3869047.61	0
cloudy periods	10100	26847.79	0
domain graphical	559	231289.24	0
environment canada	258000	6769488.53	0
fog patches	5100	12246.56	0
forecast text	2160	1698201.05	0
graphical standard	719	622529.1	0
meteorological service	64400	506666.66	0
northwest km	3	80317.46	0
office related	39000	34268077.60	0
percent chance	188000	587178.13	0
scheduled forecast	2690	211753.08	0
showers early	8360	123597.88	0
sunny periods	9370	37108.99	0
thunderstorms ending	981	9529.98	0
weather office	33400	9564373.89	0
wind northwest	2590	165523.8	0

Table 51: Algorithm 3 Results - S_2 = {kenmore, maytag}

GoogleHack(10,30,4,1,30)
whirlpool , 17.31
frigidaire , 20.55
ge , 23.40
kitchenaid , 23.70
ebuild , 35.57
dryers , 36.40
refrigerators , 37.27
sq ft , 37.47
appliance , 38.43

Table 52: Algorithm 3 Results - S_2 = {kenmore, maytag} Bi-grams

Bi-gram	Observed (“word1 word2”)	Expected (“word1 word2”)	Log Liklihood Score
sq ft	643000	21509.34	3403351.20
ft sq	3830	21461.7283950617	0
magic chef	35100	39025.57	0

Table 53: Algorithm 3 Results - S_3 = {artificial intelligence}

GoogleHack(10,15,5,1,30)	Google Sets (Large Set) - Not Complete Set
neural networks , 20.87	Neural Networks
neural , 21.63	Robotics
pattern recognition , 22.89	Knowledge Representation
genetic algorithms , 24.30	Natural Language Processing
reasoning , 24.40	Pattern Recognition
intelligent , 25.67	Machine Vision
knowledge , 25.86	Programming Languages
logic , 26.16	Data Mining
data , 26.17	Genetic Programming
applications , 26.53	Vision
artificial intelligence , 27.36	People
computer , 27.78	Publications
ai , 29.19	Philosophy
	Qualitative Physics
	Expert Systems
	Genetic Algorithms
	Expert system
	Logic Programming
	AI Magazine
	Neural Nets

Table 54: Algorithm 3 Results - S_3 = {artificial intelligence, machine learning} Bi-grams

Bi-gram	Observed ("word1 word2")	Expected ("word1 word2")	Log Liklihood Score
neural networks	617000	144620.81	954551.64
morgan kaufmann	138000	5067.61	692428.92
pattern recognition	419000	248456.79	102193.35
genetic algorithms	129000	75014.81	32818
grammatical inference	4590	1474.92	4214.81
based learning	861000	13804761.9	0
computer science	12700000	27947089.94	0
ai magazine	99500	340178.13	0
ai programming	8050	197317.46	0
based reasoning	46300	676825.39	0
case based	150000	12690476.19	0
data mining	1160000	1424162.25	0
expert systems	165000	3705114.63	0
intelligence machine	3650	587160.49	0
knowledge acquisition	86100	915590.82	0
knowledge discovery	142000	1722398.58	0
learning methods	136000	3369453.26	0
learning research	87900	22384479.71	0
learning resources	811000	22620105.82	0
learning systems	121000	11899118.165	0
list journals	5250	7086419.75	0
logic programming	126000	403984.12	0
mail comp	771	941111.11	0
natural language	1000000	3786948.85	0
pc ai	2480	204899.470	0
reinforcement learning	39700	91677.77	0
research groups	912000	17466666.66	0

Table 55: Algorithm 3 Results - S_4 = {atomic, nuclear}

GoogleHack(10,15,4,1,30)
nuclear weapons , 29.41
atomic energy , 37.73
atomic bomb , 38.81
robert oppenheimer , 39.53

Table 56: Algorithm 3 Results - S_4 = {atomic, nuclear} Bi-grams

Bi-gram	Observed (“word1 word2”)	Expected (“word1 word2”)	Log Liklihood Score
united states	80900000	20221340.38	231153374.00
los alamos	896000	40034.03	5433736.97
nuclear weapons	1380000	178539.68	3572696.81
atomic bomb	223000	34772.48	469023.03
atomic energy	597000	353333.33	152832.92
north korea	2280000	1787301.58	144903.63
robert oppenheimer	24600	9244.44	18857.95
energy commission	130000	1570811.28	0
full story	4260000	9922398.58	0
interim committee	30200	370589.06	0
nuclear power	1120000	1361269.84	0

Table 57: S_5 = {bank, money}

GoogleHack(10,15,5,1,30)
banking , 21.15
credit , 21.69
financial , 25.10
credit cards , 25.92
account , 31.42
note , 34.70

Table 58: Algorithm 3 Results - S_6 = {bank, money} Bi-grams

Bi-gram	Observed (“word1 word2”)	Expected (“word1 word2”)	Log Liklihood Score
credit cards	4050000	1583068.78	3142845.08
msn money	165000	482065.25	0
paper money	152000	3086155.20	0

4.6 Experimental Results of Sentiment Classification Algorithm

A total of 20 reviews consisting of movie reviews and automobile reviews were classified using the Sentiment Classification Algorithm. The reviews were picked from a web site that allows users to enter reviews such as $http://www.automobilereviews.com/$. The reviews were classified using different positive and negative connotation pairs such as {{*excellent, bad* } { *great , poor* }}, {{*excellent, poor* } { *good , bad* }}, and {{*excellent, poor*} {*good, bad*} {*great, mediocre*}} as input.

Overall, the Sentiment Classification Algorithm performed well on Negative reviews, it correctly predicted all 10 negative reviews as being negative. However, it performed rather poorly on positively oriented reviews. The algorithm was only able to classify 5 out of the 10 positive reviews as being positive. However, a couple of the reviews were almost neutral. A review is considered barely negative if the overall Semantic Orientation (SO) score is between 0 and -1.

Tables 61 - 68 show the various phrases extracted form the reviews, and their respective SO scores. The overall classification is given at the bottom of each table.

Table 61 shows some of the phrases extracted from a negatively classified Movie Review. The phrases are sorted in ascending order of the SO of the phrase. It can be seen from the table that the most negative phrase is *criminally cruel*, which is a good prediction, since the phrase has a sinister and extremely negative feel to it. Other phrases such as *serial killer, lonely martha, technical sloppiness* and *tasteless things* also seem to have a negative meaning that is correctly identified by the Sentiment Classification Algorithm. However, certain terms like *real person* and *well balanced* which could be positive phrases are given negative SO values. In addition, for some particular reason, the phrase *dumb giveaway* has been classified as a positive phrase. The overall classification is negative, which is correct.

The next table, Table 62 shows the Semantic Orientation scores for a particular movie review over different positive and negative connotations. The second column of Table 62 represents the SO for each phrase with the connotations {{*excellent, bad* } { *great , poor* }} and the third column represents the SO of phrases with the connotations {{*excellent, poor* } { *good , bad* }}. Overall, the SO score in Column 3 is slightly more negative than the second column of SO scores, except for the phrases *desperate need* and *urban legends*. One more interesting point worthy of noting is that the 4 phrases *amy mayfield, best known, high points,* and *thematic successor* which have a positive SO score in the first column have negative scores in the second

column of SO scores. This shows that with the variation of the positive and negative connotations, the SO score varies. This review was also correctly classified as being a negative movie review.

Next, Table 65 gives the SO orientation of phrases over a positive movie review. Note that not all the phrases that were extracted from the review are given. This particular example is every interesting in that it shows that the use of multiple pairs of positive and negative connotations can change the overall SO of a review. The second column of Table 65 represents the SO of this review with the single pair of connotations {*excellent, poor* }. The phrases in this table are sorted on descending order based on the second column. The resulting SO scores calculated by Sentiment Classification seem to be mixed. The SO of the review based on the single pair of connotations resulted in the review being incorrectly classified as negative. Terms such as *perfect fit, believable individuals* and *good balance* were given a positive SO score. However, the phrase *perfect fit* was given a 0 SO score in the third column. Also note that the phrases *believable individuals* and *good balance* which were correctly identified as being negative in the second column, have been incorrectly identified as being negative in the multiple connotation pair experiment. However, the term *amusing odyssey* received a very high score from the multiple connotation pair experiment. The overall classification by the Sentiment Classification Algorithm with the single pair of connotations is negative. This is however, an incorrect classification. The multiple connotation pairs experiment correctly predicted the review as being positive. Hence, there results from using single pairs of connotations and multiple pairs of connotations can be very unpredictable at times.

The next review classified by the algorithm is a positive automobile review. The phrases are given in Table 66. The pairs of connotations used are {*excellent, poor*} and {*good, bad*}. The algorithm performed well here, and it correctly classified the review as being positively oriented. Some of the positive phrases include *proportioned qualities, renowned Quattro, aerodynamic masterpiece, overhead cam, audio controls* and *technical enhancement*. Except for the phrase *audio controls,* all the other positively classified phrases seem to signify a positive meaning. The phrase *renowned Quattro* was assigned a positive SO of 1.24. This is a very good result since Audi is known for it's Quattro mechanism. the phrase *wise investment* was given a negative SO score of -0.49, which is incorrect. Other than that, the phrase *serious contender* was also given a negative SO score, which is also a questionable prediction. However, the phrase can be used as a positive phrase or as a negative phrase The phrase *serious contender* as in, "the Audio is a serious contender to the BMW", should be considered as a positive thing. However, the same phrase in the sentence "the Audi is a

Table 59: Sentiment Classification Algorithm Results - Accuracy

Google-Hack Classification	Actual Classification		
	Pos	Neg	
Pos	5	0	5
Neg	5	10	15
Total # of Reviews	10	10	

serious contender for the most unreliable car title", is definitely not a positive use of the phrase. Hence, this shows that there exists a certain amount of ambiguity in using 2-word phrases as a means of predicting the SO of a paragraph of text.

Table 60: Phrases from a Negative Movie Review - Sorted Ascending on SO

Phrase	SO(phrase)
{{excellent, poor }, { great , bad }}	
criminally cruel	-3.14
lonely martha	-2.08
french kiss	-1.85
criminally idiotic	-1.75
serial killer	-1.57
perfect normalcy	-1.45
asinine ending	-1.43
fake persona	-1.39
as drastic	-1.26
total loony	-1.09
acceptable development	-1.08
technical sloppiness	-1.08
dive past	-0.91
tasteless things	-0.91
initial impetus	-0.89
new film	-0.87
hot affair	-0.86
way underdeveloped	-0.84
real person	-0.78
first section	-0.71
well balanced	-0.3
incomparable joan	-0.09
dumb giveaway	0.46
Predicted Semantic Orientation	Negative

Table 61: Phrases from a Negative Movie Review - Sorted Ascending on SO

Phrase	SO(phrase)	SO(phrase)
	{{excellent, bad } { great , poor }}	{{excellent, poor }, { great , bad }}
poor successor	-1.93	-2
startling dearth	-1.58	-1.62
instead wallowing	-1.57	-1.84
critical revulsion	-1.36	-1.52
galling fact	-1.32	-1.46
lucrative hitchcock	-1.28	12.14
huge stretches	-1.22	-1.43
third rate	-1.18	-1.36
somebody worthy	-1.14	-1.28
just thrown	-1.11	-1.25
completely defused	-1.09	-1.22
downright embarrassing	-1	-1.2
other plot	-0.84	-1.03
actually spent	-0.84	-1.06
amy protests	-0.82	-1.03
potential suspects	-0.77	-0.98
teen slasher	-0.76	-0.99
desperate need	-0.75	-0.7
thoroughly unengaging	-0.73	-0.86
urban myths	-0.73	-1.03
just used	-0.73	-0.78
most notable	-0.65	-1.06
first movie	-0.63	-0.62
final cut	-0.62	-0.89
urban legends	-0.62	-0.48
own mystery	-0.55	-0.84
brief history	-0.25	-0.81
Predicted Semantic Orientation	Negative	Negative

Table 62: Phrases from a Negative Movie Review - Sorted Ascending on SO

Phrase	SO(phrase)
	{{excellent, bad } { great , poor }}
old song	-1.12
more scary	-1.1
prior conviction	-1.08
new meaning	-1.08
evil creature	-1.06
really stupid	-1.04
little horror	-0.93
conspicuously absent	-0.91
not know	-0.88
gruesome conclusion	-0.88
old actor	-0.87
satanic film	-0.81
horrifying jagged	-0.8
desolate countryside	-0.79
supernatural abilities	-0.66
scary movies	-0.6
justin long	-0.54
bizarre feeding	0.37
Predicted Semantic Orientation	Negative

Table 63: Phrases from a Negative Movie Review - Sorted Ascending on SO

Phrase	SO(phrase)
{{excellent, bad } { great , poor }}	
bad action	-1.97
macabre stories	-1.5
human savages	-1.35
ridiculous dialogue	-1.27
ethical arguments	-1.15
dirty apes	-1.09
completely absent	-0.69
unimaginative narrative	-0.64
smart chimps	-0.52
predominant themes	-0.33
vicious general	0.27
classic novel	-0.23
Predicted Semantic Orientation	Negative

Table 64: Phrases from a Positive Movie Review - Sorted Descending on Column 2

Phrase	SO(phrase)	SO(phrase)
	{excellent, poor}	{{excellent, poor} {good, bad} {great, mediocre}}
perfect fit	0.35	0
believable individuals	0.1	-0.57
good balance	0.05	-0.29
critically acclaimed	-0.09	-0.39
amusing odyssey	-0.09	7.79
creative process	-0.23	-0.56
dead dog	-0.39	-0.52
top notch	-0.41	-0.75
aptly named	-0.53	-0.53
pleasurable journey	-0.58	-0.61
well acted	-0.59	-1.34
clearly illustrated	-0.59	-0.83
screwball elements	-0.61	-0.96
right note	-0.71	-1.06
slow moving	-0.75	-0.94
overall plot	-0.84	-1.74
personal struggles	-0.91	-1.63
screwball comedy	-0.92	-1.36
melodramatic realism	-0.93	-1.14
stereotypical hollywood	-0.96	-1.95
new love	-1	-0.84
lone exception	-1.02	-1.85
sometimes harrowing	-1.04	-1.52
final fate	-1.06	-1.7
smart movie	-1.39	-2.07
hilarious performance	-1.48	-2.55
fake gun	-1.68	-2.08
Predicted Semantic Orientation	Negative	Positive

Table 65: Phrases from a Positive Automobile Review - Sorted Descending on SO

Phrase	SO(phrase)
	{{excellent, poor }, { good , bad }}
proportioned qualities	12.64
renowned Quattro	1.24
aerodynamic masterpiece	0.78
overhead cam	0.71
audio controls	0.7
interior Audi	0.48
solar sunroof	0.34
other luxury	0.23
technical enhancement	0.21
attractive features	-0.25
somewhat customized	-0.34
finally available	-0.38
remote transmitter	-0.4
standard A6	-0.4
new versions	-0.49
wise investment	-0.49
skeptical buyer	-0.52
serious contender	-0.7
Predicted Semantic Orientation	Positive

Table 66: Phrases from a Positive Automobile Review - Sorted Descending on SO

Phrase	SO(phrase)
	{{excellent, poor }, { good , bad }}
remote entry	1.11
Standard equipment	0.87
rear defroster	0.56
good dampening	0.43
antilock brakes	0.41
driver comfortable	0.35
reasonable price	0.2
optional leather	0.11
Interior space	0.08
likable car	-0.02
compact sedan	-0.11
nice range	-0.23
very good	-0.32
Resale value	-0.45
long trips	-0.55
small car	-0.68
American cars	-0.75
adequate pickup	-0.75
entire car	-0.81
not exceptional	-0.82
not sporty	-0.86
far smoother	-1.01
rough road	-1.02
especially generous	-1.04
not bad	-1.41
Predicted Semantic Orientation	Barely Negative

Table 67: Phrases from a Positive Automobile Review - Sorted Descending on SO

Phrase	SO(phrase)
	{{excellent, poor }, { good , bad }}
Superb handling	0.19
automotive pleasure	1.62
automatic climate	1.1
noteworthy selection	0.99
dual zone	0.94
automatic transmission	0.74
Rear passengers	0.63
precise responsiveness	0.15
optional side	0.1
free clutch	0.07
well equipped	0.06
satisfying package	-0.18
asthetic beauty	-0.21
selective car	-0.33
block variable	-0.38
Other features	-0.44
not overlooked	-0.58
joyful experience	-0.61
accessible power	-0.65
overall competance	-0.82
real love	-0.98
unusually elevated	-1.09
different rpm	-1.2
Predicted Semantic Orientation	Barely Negative

5 Related Work

5.1 Finding Sets of Related Words

The purpose of this thesis research is to automatically find sets of related words by using the WWW as a source of information. Methods have been developed to identify sets of related words, however, almost all these methods are based on extracting sets of words from a static corpora of text. In the next two sections we discuss two methods, where the first method is used to identify sets of related words from a static corpora of text, and the second methods is used to automatically construct a hypernym-labeled noun hierarchy.

5.1.1 CBC (Clustering By Committee [7, 8])

The algorithm known as Clustering By Committee (CBC) tries to ”cluster” words with similar meanings together to create sets of related words [7, 8]. Take for example the word ”engine”. The word engine has several possible meanings. For example it could refer to a tool like Google which is a “search engine” or it can refer to an automobile which is ”run by” an engine. The algorithm will identify sets of words associated with each of the different senses of engine, for example ”turbine, electric motor” versus ”software, search”. This way, the CBC algorithm tries to discover concepts from text.

The CBC algorithm works by creating a centroid cluster by averaging the feature vectors of a subset of cluster members. Take for example a cluster for U.S state names. Some of its features or rather the contexts in which a U.S state name can appear can be capital, campaign in, governor of, senator for etc. The CBC algorithm works by representing each word by a feature vector which corresponds to a context in which a particular word can be used. Take the example ”threaten with _” as a context, if the word ”handgun” occurred in this context, ”threaten with” is a feature of handgun. The features are assigned values by using the PMI measure.

The algorithm has three phases. Phase 1 calculates each elements top k similar elements. Phase 2 constructs a collection of tight clusters. The elements of each cluster form a committee. The CBC algorithm tries to form as many committees as possible, ensuring that each new cluster is not very similar to any of the existing clusters. The final Phase assigns each element to its most similar cluster. A parser known as Minipar was used to parse about 1 GB of newspaper text. The contexts are then collected along with their frequency

counts. PMI values can then be assigned to these sets.

There are a number of clustering algorithms other than CBC that have been used to cluster concepts. K-means, Chameleon and Bisecting K-means are examples of other clustering algorithms. However, the CBC algorithm out performs these algorithms when applied to find concepts.

5.1.2 Automatic Construction of a hypernym-labeled noun hierarchy [3]

The general idea behind the algorithm proposed in [3] to create a hypernym-labeled noun hierarchy lexicon is based on extracting conjunctions of noun phrases and appositives from any corpus. The goal here is to create a method that can automatically create a noun hierarchy regardless of the domain of the corpus on which it is run. In this paper, the corpus used is the Wall Street Journal corpus. As an example of a noun phrase conjunction, consider the phrase ”executive vice president and treasurer” [3] . Here, the phrases ”executive vice president” and ”treasurer” could be considered as being semantically related. As an example of an appositive, consider the phrase ”Northwest, an airline”. Here Northwest and airline can be regarded as semantically related. Hence, for each noun, a vector is created by identifying all other nouns that appeared with it as either a conjunction or as an appositive. In addition, the vector for each noun also maintains the number of times a particular noun occurred as a conjunction or appositive the extracted noun. Finally, to identify if two nouns are similar, the cosine between their respective vectors are calculated.

Once the vector for each noun has been created, a bottom-up clustering approach is used to create a single tree structure containing a hierarchy of nouns [3]. The first step here is to treat each noun as an individual node in the tree. Next, the two most similar nouns are grouped together with a common parent. This process is repeated until a single common parent has been achieved.

The next step in the algorithm is to assign hypernyms to each node in the tree. This process is also automatically implemented by extracting hypernyms by looking for patterns such as ”X, Y, and other Zs, where Z can be considered as a hypernym of X and Y. The process of creating a vector for each noun is repeated, however, in this case, for each noun, the list of hypernyms instead of nouns that appear with it are the dimensions of the vector. Once these vectors have been created, to assign a hypernym to a node, first a common hypernym vector is created by adding together the vectors of all its child nodes. Once this common hypernym vector has been created for a node, the top 3 most common hypernyms in the vector are assigned as hypernyms for

the node. This process results in a hypernym-labeled noun hierarchy.

5.2 Sentiment Classification

The area of sentiment classification has received a lot of attention in recent times. Initially, a lot of work was focused on manually identifying words that convey sentiment. That is, a word can either have a positive sentiment, a negative sentiment or no sentiment at all. For example, the word ”gleeful” has a positive sentiment, and the word ”sadness” has a negative sentiment. Hence, both these words would be regarded as affect words. Lexicons such as the Lasswell Value Dictionary and the General Inquirer Dictionary have been created in order to identify a list of affect words and their dimensions [5].

However, the focus has now shifted into automatically identifying words that carry a sentiment in order to automatically create a comprehensive lexicon of affect words. In addition to automatically creating a lexicon of affect words, there has been a fair amount of research focused on automatically classifying a particular word or text as having a positive or negative sentiment. In this chapter we focus on two papers that concentrate on using the World Wide Web as a source of information to identify a lexicon of affect words, and to identify the sentiment or semantic orientation of words.

5.2.1 ”Thumbs up or Thumbs down? Semantic Orientation Applied to Unsupervised Classification of reviews” [10]

The Point wise Mutual Information - Information Retrieval (PMI - IR) algorithm uses the Semantic Orientation (SO) of phrases to decide if a review has a positive or negative recommendation [10]. These reviews can be automobile reviews, movie reviews, bank reviews and or even travel destination reviews.

The algorithm works in three steps. First, a part-of-speech tagger is used to identify words in the review as adjectives, nouns, verbs or adverbs. Next, the algorithm calculates the semantic orientation of each of the adjectives, nouns, verbs and adverbs extracted from the review by forming phrases out of adjacent words. The algorithm then queries a search engine with the combination of positive and negative words with each of these phrases to find the PMI measure, and thereby the Semantic Orientation. The final step is to calculate the average of the semantic orientation calculated thus far, and assign a value to the review as ”recommended” or ”not recommended”.

The base for the PMI-IR algorithm is the Pointwise Mutual Information (PMI) measure between two words. PMI can be defined as the ratio between the probability that word1 and word2 co-occur (p(word1 word2)) and the probability that word1 and word2 occur if word1 and word2 are statistically independent (p(word1) p(word2)) .

The equation below represents PMI,

$$PMI(word1, word2) = \log \frac{(p(word1\ word2))}{(p(word1)p(word2))} \tag{9}$$

From the equation above it can be said that PMI measures the degree of statistical dependence between two words. In the case of the PMI-IR algorithm for reviews, the Semantic Orientation is calculated by finding the difference between PMI (phrase, ”excellent”) and PMI(phrase, ”poor”).

Here, the positive inference of a word is represented by ”excellent”. That is, it can be said that a word has a ”good association” when it is mentioned with excellent in the same sentence. The ”NEAR” operator from the search engine AltaVista serves as the tool to find the probabilities. For example, take the query ”Hawaii” NEAR ”excellent”. Hawaii is assumed as a travel destination. We find out if ”Hawaii” is a recommended travel destination by looking at the number of the web pages in AltaVista that mention ”Hawaii” and ”excellent” in the same sentence or a sequence of 10 words. The same reasoning applies for ”Hawaii” and ”poor”. Hence, it can be seen that if ”Hawaii” is mentioned with ”poor” more often than with ”excellent”, the semantic orientation will be negative.

Hence the Semantic Orientation for a phrase using the PMI-IR algorithm can be represented by the following equation,

$$SO(\textit{phrase}) = \log_2 \frac{hits(\textit{phrase}\ NEAR\ \text{“}excellent''))\ hits(\text{“}poor'')}{hits(\textit{phrase}\ NEAR\ \text{“}poor'')\ hits(\text{“}excellent'')} \tag{10}$$

A positive average of the calculated semantic orientation recommends the subject of the review, and a negative average does not recommend the subject of the review.

5.2.2 Validating the Coverage of Lexical Resources for Affect Analysis [5]

A lexicon of affect words denoted by hand was created by (Subasic and Huettner 2000) [5]. Each entry in the lexicon was described by the following 5 fields: a lemmatized word form, a part-of-speech tag, an

affect class, a weight of centrality to that class, and a weight for the intensity of that word in that class. For example, the entry for the word ”gleeful” in the lexicon contained the following:

”gleeful” adj happiness 0.7 0.6
”gleeful” adj excitement 0.3 0.6

This entry basically says that the word ”gleeful” is an adjective that belongs to the two affect classes happiness and excitement, and it has a giher centrality or relatedness to the affect class ”happiness”. The last value 0.6 in both entries signifies the intersity of the word ”gleeful”. In order to validate the comprehensiveness of this lexicon, Grefennn et al proposed the following method of automatically obtaining a list of affect words. The idea here was certain phrases such as ”appears extremely __” can introduce a positive or negatively charged word. For example, the sentence ”appears extremely good” introduces the positive word ”good” and the phrase ”appears extremely bad” introduces the negative word ”bad”. Hence, 105 such patterns were identified by first creating a list of 21 words such as appear, feel, are is and then identifying 5 words that commonly follow the 21 words such as almost, extremely, so, too, very to create the input patterns. Once these patterns were created, queries were issued to the WWW, specifically to the site www.alltheweb.com, and 4000 snippets of text containing each pattern were extracted.

Once the snippets were extracted, a list of the words following each pattern and the frequency of occurrence counts of each of these words were established. Around 15,000 unique inflected words were extracted by these experiments. From this list of words, around 4746 words that at least had a frequency count of 3 were manually tagged by the authors as being an affect word or not. Of these 4746 words, 2988 words were found to be affect words. This list of affect words were then used as a gold standard, specifically, the Clairvoyance Gold Standard.

Next, instead of manually classifying each of these words as being an affect word or not, the authors used PMI-IR algorithm to classify the sentiment of each word. In this case, the authors used the approach described in Turney and Littman (2003) where, multiple pairs of positive and negative words were used to calculate the Semantic Orientation of each word returned earlier by extracting information returned by www.alltheweb.com.

6 Conclusions

The overall goal of this thesis research was to find sets of related words by using the World Wide Web as a source of information. As a result, we developed 3 Algorithms that try to predict sets of related words.

The first algorithm was a simple baseline approach that depended on pattern matching techniques and frequency counts of words to identify sets of related words.This algorithm performed reasonably well, and was able to identify words that are related in meaning to the input set. However, since we depended on raw frequency counts to rank the words from the most relevant word to the most irrelevant word, certain irrelevant and noisy words were being pushed to the top. For example, in the set of words returned for the input set *red* and *yellow*, the terms *enterprise, software* and *solutions* were regarded as the most relevant, and the terms *black* and *blue* were regarded as the most irrelevant. One additional problem with Algorithm 1 was that it did not allow bigrams (2-word phrases) as input, or as output. As bigrams were not accepted as input, to find the set of related words for words such as *George Bush* and *Bill Clinton*, the set had to be broken into *Bush* and *Clinton*. In addition, terms such as *White House* were split into separate words such as *White* and *house* in the set of related words that were returned.

Algorithm 2 developed with the goal of addressing the issues raised above, and was basically an improvement of Algorithm 1. Algorithm 2 allowed more than two words as input, and it also allowed bigrams as output. In addition, we extended the Jiang and Conrath similarity measure such that it can be applied to hit counts retrieved from the WWW [6]. This approach performed very well, and the amount of noise seen in Algorithm 1 was reduced. In addition, the most relevant terms were often ranked higher than the irrelevant terms. Once again, consider the example input set *red* and *yellow*. Algorithm 2 returned the words *black* and *blue* as the most relevant words to *red* and *yellow*. The other terms such as *enterprise, software* and *solutions* that were returned as the most relevant by Algorithm 1 were automatically discarded since those terms did not meet the relevance cutoff score.

The addition of bigrams in the sets of related words introduced a new problem. Most of the bigrams returned by Algorithm 2 were good collocations. However, occasionally the algorithm also returned really bad collocations as results. For example, for the search terms *sunny* and *cloudy*, the bigram *forecast text* was returned as a resulting related term. Obviously, this is not a good collocation. This problem lead us to the development of Algorithm 3. Algorithm 3 was simply an extension of Algorithm 2, where it included a

function that tries to identify bad collocations. We used the Log Likelihood score as a means of identifying bad collocations.The results from Algorithm 3 were quite good, and bigrams such as *forecast text* that were returned by Algorithm 2, were automatically discarded by Algorithm 3 as being a bad collocation.

Overall, the algorithms perform well for input sets that contain brand names such as *toyota* and *ford* in comparison to more general input sets such as *red* and *yellow*. This can easily be explained by the fact that a search engine such as Google is commercialized. The latter set returns web sites such as *www.redlobster.com* and *www.redbull.com*, which are both commercial entities.

The final addition to this thesis was the extension of the PMI-IR algorithm proposed in (Turney 2003). The PMI-IR algorithm was extended to include multiple pairs of positive and negative words, and Google was used as the search engine, instead of AltaVista.

In conclusion, the methods proposed in this thesis research show that the World Wide Web can be used to process natura language with good accuracy.

7 Future Work

The three Algorithms developed in this thesis research that try to predict sets of related words by using the WWW as a source of information show that the WWW can be used for processing text and extracting valuable information. However, there are still many improvements that can be made to the Algorithms.

Some of the possible refinements and additions are outlined below:

7.1 Proximity Based Ranking

It would be interesting to see the result of adding a proximity score to the words in the set of related words. The basic idea would be to weight the relatedness score of a word based on how close it appears to the search terms in the parsed text. The set of words can also be restricted words that occur within a window size of n words.

7.2 Web Page Rank Based Ranking

Since Google does a very nice job of finding the most relevant information to the search terms, it would be interesting to see how the results of the set of related words would vary, if we based a particular word's ranking on the web pages that it appears on. For example, if we search for *Toyota* and if the word *Honda* appeared on the first, second and fourth web page, it would be assigned an overall ranking of (1+2+4)/3. The lower the score the higher the ranking for the word. This ranking can then be used to weight the overall relatedness value.

7.3 Web Page Parser

Consider the use of a different and better web page content parser. Currently we used a freely available PERL package HTML::TokeParser. This package performs fairly well in removing HTML tags etc, however, there are occasions in which Java script variable names filter through the parser and appear in the set of related words. These kind of words are difficult to discard since they occur with high co-occurrence frequencies.

7.4 Restrict the links that are traversed

Another interesting improvement that is worth considering is restricting the links that are traversed. That is, when Google Hack retrieves the content of a particular web page, it also traverses and retrieves the content of the web pages that are linked to by that web page. Though this allows us to retrieve more information, it can also lead us to a non-related web page. For example, a web page result for the search term “Toyota” can also link to a bank that does financing of loans. Consider another example, where a particular search terms retrieves a web page that sells a lot of products. Such a web page can link to many other web pages that sell different products. For example, if a query for a product such as an *I-pod* retrieves a commercial web site such as Amazon, it can easily lead us to many other products that are currently daily deals on Amazon. This can lead to irrelevant information. Hence, resulting in irrelevant words being included in the set of related words. Therefore, it would be interesting to develop some sort of technique to distinguish between relevant and non-relevant links.

7.5 Integrate Multiple Search Engines

One addition refinement to consider for Google Hack would the use of multiple search engines. Yahoo Incorporated has released a Yahoo API, which like the Google API, allows programs to interact with their database and retrieve web pages . Hence, by using the Google API and the Yahoo API, the resulting sets can be compared over slightly different sources of information. One additional advantage to using the Yahoo API is the query limit by Yahoo is 5 times more than Google. Google restricts the number of queries per API key to a 1000 queries a day . Whereas, Yahoo allows 5000 queries per day per API key .

7.6 Restrict Web Page Domain

The final suggestion for improvement would the ability to restrict Google’s search to a web site. For example, if we restrict Google’s search to sites that contain more academic text rather than commercialized web pages, the results for certain non-specific input sets might improve.

The improvements suggested in this section are very feasible, and we hope to continue to try and improve the quality of the sets of related words that are returned by Google Hack.

References

[1] Praveen Aggarwal and Vaidyanathan Rajiv. Mirror, mirror, on the web: A lexical-semantic analysis of brand positioning. *Advances in Consumer Research*, 32.

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[3] S. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text, 1999.

[4] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1994.

[5] Gregory Grefenstette, Yan Qu, David A. Evans, and James G. Shanahan. Validating the coverage of lexical resources for affect analysis. 2003.

[6] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the Tenth International Conference on Research on Computational Linguistics (ROCLING X)*, Taiwan, 1997.

[7] Dekang Lin and Patrick Pantel. Concept discovery from text. In *Proceedings of the Conference on Computational Linguistics*, pages 577 – 583, Taipei, Taiwan, 2002.

[8] Dekang Lin and Patrick Pantel. Discovering word senses from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD-02)*, pages 613 – 619, Edmonton, Canada, 2002.

[9] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[10] Peter D. Turney. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics*, pages 417 – 424, Philadelphia, Pennsylvania, 2002.

[11] Peter D. Turney and M.L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21(4):315–346, 2003.