An In Depth Look at Two Approaches to Sentiment Classification

Shannon McGrael
Swarthmore College
CS97 — Senior Conference
smcgrae1@swarthmore.edu

Stephen Michael Smith
Swarthmore College
CS 97 — Senior Conference
ssmith1@swarthmore.edu

Abstract

There have been a variety of approaches to the problem of categorizing text based on sentiment. Many of these approaches require a large amount of user input and effort. In this paper we compare two very different manners of categorizing text, both which requires minimal user input. Both techniques parse through reviews in various domains (video games, movies, etc) and attempt to correctly categorize the review as positive or negative. Through this process we will be able to better understand the problem, as well as the strengths and weaknesses of the solutions. From this we hope to create a basis for developing a more robust system.

1. Introduction

As the world enters into the information age, technologies are emerging that allow faster and more complete access to vast bodies of knowledge and text. The growth in quantity and diversity means that it has become increasingly more difficult to locate text that is particularly relevant to a certain topic. In many cases the shear size of the available data can be a hindrance causing one to find less data rather than more. This obstacle has created demand for a classification system capable of sorting through text, and going beyond a regular search engine by taking a user s personal preferences and the sentiment of the text into account as well. There are undoubtedly millions of applications for this type of technology. What has held it back is the low accuracy of previous results and the need for user input that is difficult to amass. What is needed is a robust system that can accurately perform sentient classification with little input from the user.

2. Applications

Peter Turney (Turney, 2002) and Ken Lang (Lang, 1995) speak of different applications and algorithms for addressing the problem of automated review ranking and text categorization. This technology is useful in search engines, news services, information boards, sales and many other applications.

There are various applications for review ranking in search engines. Turney suggests that a user could type in a query such as Akumal travel review and the search engine could then return the result that There are 5,000 hits, of which 80%

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

are thumbs up and 20% are thumbs down (Turney, 2002). There are millions of objects that are reviewed on the web; the problem is compiling it all together. Users could also ask to see only reviews of a certain ranking. This way they could sort through the reviews to get only the data they needed. This could be useful in immeasurable ways. An artist or any user could easily sift through reviews to understand the criticism of a certain work of art. Web designers could quickly find out what features were liked/disliked by their users. Or it could be as simple as consumers quickly being able to benefit from the opinions of others in an effort to get the most of their money.

Ken Lang (Lang, 1995) uses this technology in a netnews-filtering system called NewsWeeder. This system learns a user's preferences through the reviews they give to certain news articles, and through the reviews that other users like them give to news articles. With this information NewsWeeder is able to present users with more of the news articles that are important and relevant to them and less of those that are not.

These systems could also be used in order to determine when a particular piece of text was unusually positive or negative. Many times it is very important that text be impartial and fair, such as judge s comments, text books, and news articles. Systems such as the ones described below could be used to flag text that is unusually emotional, when that type of response is not appropriate. Likewise these systems may even be able to be adapted to learn to detect ideological differences in articles. For instance a system may be able to detect if a newspaper was particularly liberal or conservative or feminist or anti-feminist.

3. Approaches

3.1 Semantic Orientation

Peter Turney utilized the concept of Semantic Orientation (SO) in his approach to this task (Turney, 2002). A phrase has a positive semantic orientation if it is most often seen in conjunction with known positive words such as excellent . A phrase has a negative semantic orientation if it is most often used in conjunction with known negative words such as poor . His approach to this problem centers on the idea that adjectives and adverbs are often present in evaluative (as opposed to factual) sentences (Hatzivassiloglou & Wiebe, 2000; Wiebe, 2000; Wiebe et al., 2001). This means that they often convey the feelings of the writer more so than any other part of speech. The algorithm attempts to estimate semantic orientation of an entire document (a single review) using information about the adjective phrases within that document.

The first step in this approach is to part-of-speech tag the document and extract all bigram phrases that contain adjectives or adverbs. We used the fnTBL1.0 englishPOS tagger for the POS tagging. The bigrams are extracted because the adjectives or adverbs by themselves don t necessarily give all of the information. This is true because adjectives and adverbs are, by definition, modifiers, so it is easy to see that their semantic orientation is influenced by the words that they are modifying. An example that Turney gives is:

> the adjective unpredictable may have a negative orientation in an automotive review, in a phrase such as unpredictable steering , but it could have a positive orientation in a movie review, in a

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

phrase such as unpredictable plot (Turney, 2002)."

So, the modified word acts as context words to accent the sentiment of the adjective. See Turney 2002 for the rules used to extract usable two-word phrases from reviews.

Turney then estimates the semantic orientation of each adjective phrase by making queries to the AltaVista Search Engine (www.altavista.com), ours looked like this:

(<Word1>+AND+<Word2>)+NEAR+poor.
(<Word1>+AND+<Word2>)+NEAR+excellent.

We then recorded the number of hits for each phrase (we ll call them hitsneg and hitspos, respectively). We also recorded the total number of hits if we just search for poor and excellent individually (we ll call them hitspoor and hitsex, respectively). Using this information, we can estimate the SO for a given adjective phrase by:

$SO(phrase) = \log_2((hitspos * hitspoor) / (hitsneg * hitsex))$.

The nature of this calculation is such that if hitsex > hitspoor, then $SO_{phrase} > 0$. Alternatively, if hitspoor > hitsex, then $SO_{phrase} < 0$.

To compute the SO for a given review, we simply take the average of the SO s for all of the adjectives that we extracted from this document. In order to avoid counting phrases that have very little correlation to either poor or excellent , we remove any phrase which gets less than 4 hits for both of the AltaVista queries from this calculation. Also, to avoid a possible division by zero, we added .01 to the number of hits for every individual phrase. Any document with an SO greater than zero is considered to be recommended, and any document with a negative SO is considered to be not-recommended.

3.2 Bag of Words

Lang s method, commonly referred to as the bag of words method, requires separate training and testing data. The idea behind this method is that similar texts will contain similar words. Lang's method does not take word order or syntax into account. The method assumes that most of the information needed to differentiate between texts is in the words themselves. For the purposes of this research we have implemented a system similar to that described by Ken Lang in his paper: NewsWeeder: Learning to Filter Netnews.

In order to implement the bag of words method, the text must first be parsed separated into tokens, each token being a word or punctuation mark. These tokens are then converted into vectors the length of the vocabulary. One main vector is created for the whole corpus that is a list of every unique token in the corpus. Then each separate document or review has its own corresponding vector that contains numerical values indicating the number of times a particular word appeared in that document. A zero for at the location of a certain word would indicate that it did not appear in the document, whereas a 20 would indicate that it frequently appeared in the text.

In order to complete the learning process a representative vector must be created for each separate category. First the document vectors must be changed into normalized vectors, then by using least-squares regression, a numeric value is assigned to each of them. All the vectors in the training corpus pre-determined to be of the same classification are then averaged to get a representative vector for each category.

At this point the training process is completed. To test the system a review must be parsed and turned into a word

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

vector and then given a least-squares regression value in the same manner as the vectors in the training corpus. This value is then compared to each of the representative category vectors. The system guesses that the new document is of the same category of the representative vector closest to it.

4. Corpus Creation

Our corpus consists of 100 reviews taken from the Epinions web site (www.epinions.com). The original idea of this project was to solve the problem of sentiment classification on video game reviews. However, any game that was popular enough to have more than a handful of reviews were generally given a recommended rating. So, we took 50 reviews of the game Tekken Tag Tournament (10 not recommended, 40 recommended), and we took 50 reviews of the movie Gladiator (25 not recommended and 25 recommended) for our corpus data. Since Lang s algorithm requires training data, we split the two halves of the corpus into 40 reviews for training and 10 reviews for testing. The results given in this paper are for those 20 reviews we used for testing, with additional results for Turney s system having been run on the entire corpus.

Table 2. Results for 10 Tekken Tag Tournament reviews

		ACTUAL	
		RECOMMENDED	NOT RECOMMENDED
OUR	RECOMMENDED	2	2
SYSTEM	NOT RECOMMENDED	6	0

5. Results

For the purposes of comparison, we ran both algorithms on the smaller corpus of 20 reviews that we separated from our corpus for testing. The bag of words method uses the other 80 reviews for training purposes. Because Turney s algorithm does not require training data we were able to run it on the entire corpus of 100 reviews.

5.1.1 Turney s Results (Comparable to Lang s Approach)

When run on the test corpus, Turney s approach receives a 40% on the video game reviews, and a 60% on the movie reviews. Tables 2 and 3 show the results we achieved for the test corpus.

Apparently, the reviewers used some pretty negative word-choice to describe their pleasure with the game. This is understandable since it is a fighting game and we may be getting negative feedback from words that the authors mean to use as positive aspects (i.e. prolific gore , massive hits , etc.).

Table 3. Results for 10 Gladiator reviews

		ACTUAL	
		RECOMMENDED	NOT RECOMMENDED
OUR	RECOMMENDED	2	3
SYSTEM	NOT RECOMMENDED	3	2

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

Again, the reviewers seem to have used phrases which in everyday random text would be viewed as negative, but in the context of the entire document, it should be seen as positive. One example bigram from our corpus is disastrous effects . This bigram appears in a section of the review in which the author is describing something that happens in the movie, not how he/she feels about the movie itself. However, our algorithm would incorporate this bigram into a lower Semantic Orientation score than the review most likely deserves.

Table 3a shows a comparison of actual adjective bigrams tested in reviews that Turney s system correctly classified as recommended or not-recommended.

Table 3a. Comparison of adjectives in correctly classified positive and negative reviews.

SAMPLE BIGRAMS from a correctly classified POSITIVE REVIEW (TEKKEN TAG)	SAMPLE BIGRAMS from a correctly classified NEGATIVE REVIEW (GLADIATOR)
Nice night	Many clues
Extra cash	Upper hand
Good amount	Bad guy
Good characters	Pretentious corniness
Main arcade	Embarrassingly easy
Different stages	Inefficient republic
Pretty cool	Natural enemy
Other modes	Not interested
Basic fighters	Moral development
Different buttons	Not entertained
Fairly new	Loving violence
Special combos	Violent scenes
Good fighting	Elaborate distraction
Awesome graphics	Less distasteful
Widely successful	Misogynistic hatred
Joyous sound	Questionable moments

5.1.2 Turney s Extended Results

Since Turney s algorithm does not require training, we decided it might be interesting to see how it runs on the entire corpus of 100 reviews. The final results for this run are as follows:

1. We correctly classified 21 of the 50 Tekken Tag Tournament reviews, giving a precision score of 42%, which is slightly better than the score we received by only running it on a small number of reviews.
2. We correctly classified 26 of the 50 Gladiator reviews, giving a precision score of 52%, which is only slightly better than the baseline score of 50% if one were to choose recommended every time.

Table 4a is a compilation of the results from the 50 Tekken Tag Tournament reviews, and Table 4b is a compilation of the results from the 50 Gladiator reviews.

5.2 Lang s Results

For the following tests the data from Tekken Tag Tournament and Gladiator were always tested separately. The corpus was divided into two categories of reviews, recommended and non-recommended. In order to train the system and create the representative vectors for each category in each domain, 40 of the 50 available reviews were read in and put into token vectors, as previously described. The vectors created from the test documents were then compared to these two representative vectors.

The first round of results were very poor. The results were near 50% for both domains. The data was difficult to manipulate because of it very high

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

dimensionality. Also commonly used words such as and , I , is , etc occur so frequently that they were outweighing other words that would have had much more descriptive power. Because of this the representative vectors for the two separate categories were so similar that it was very difficult to differentiate them.

To improve upon our results we implemented a pruning method that would limit the number of words that the system took into account. After all the training and testing data was read into the system. Each token that appeared more than X number of times in the whole corpus was eliminated from the word vectors. Then these pruned vectors were normalized and there cosine similarity was found. This has the effect of disregarding words that are used frequently throughout the corpus and that have no relevance in determining the category of a particular document. It also reduced the dimensionality of the vectors making them more precise and easier to manipulate.

The bag of word s approach is correct 40% on the movie reviews, and a 90% on the video game reviews. Tables 5 and 6 show the results we achieved for the test corpus in two different formats.

Table 4a. Results from the complete corpus of 50 Tekken Tag Tournament Reviews

		ACTUAL	
		RECOMMENDED	NOT RECOMMENDED
OUR	RECOMMENDED	18	7
SYSTEM	NOT RECOMMENDED	22	3

Table 4b. Results from the complete corpus of 50 Gladiator Reviews

		ACTUAL	
		RECOMMENDED	NOT RECOMMENDED
OUR	RECOMMENDED	9	8
SYSTEM	NOT RECOMMENDED	16	17

Results for 10 Gladiator reviews
Table 5b.

Actual Rating	Our Rating
N	Y
N	N
N	Y
N	N
N	N
Y	N
Y	N
Y	Y
Y	N
Y	N

Table 5a.

		ACTUAL	
		RECOMMENDED	NOT RECOMMENDED
OUR	RECOMMENDED	8	1
SYSTEM	NOT RECOMMENDED	0	1

6. Difficulties

As we attempted to create these systems we came upon many obstacles. Some of these obstacles were unique to our chosen method and others were related to the problem and the way people express themselves.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

Results for 10 Tekken Tag Tournament Reviews
Table 6a.

		ACTUAL	
		RECOMMENDED	NOT RECOMMENDED
OUR	RECOMMENDED	2	3
SYSTEM	NOT RECOMMENDED	3	2

Table 6b.

Actual Rating	Our Rating
N	Y
N	N
Y	Y
Y	Y
Y	Y
Y	Y
Y	Y
Y	Y
Y	Y
Y	Y

The test results can be found in greater detail in appendix A.

One of the largest problems that we both should have expected but that became very apparent as we sorted through our text and results was that people use very different ways of expressing themselves. This problem took a few different forms. What one person may consider to be a negative attribute others would consider a positive. One line of text could be indicative of contrasting emotions depending upon the person, their manner of speech and their likes and dislikes. For example: I had never seen a movie so focused on war, destruction and lifelike, gory violence as Gladiator. While one reviewer may hate the movie for its extensive action scenes and gory details, others may find that to be one of its most exciting aspects.
The other case where the diversity of people is a big obstacle is in the inconsistency of their rankings. Each review was written by a different person with a different concept of what recommended and non-recommended means, where to draw that line, and for what reasons. For example, about half the people that ranked Gladiator as three stars recommended it and the other half did not. But, even more difficult to deal with were the cases where a reviewer would rank the movie as one or two stars, however would still recommend the movie.

6. Pros and Cons

6. 1 Turney Pro s and Con s

In general, this is a very simple algorithm. The system is standalone, and requires no training time or training data. This must be why he called his system “unsupervised,” even though by hard-coding the words “poor” and “excellent” into the estimation algorithm, he automatically gave his system prior knowledge that was helpful in solving the problem.

This system is easily portable to other domains while remaining generalizable within the domain of reviews. Turney showed that he could get much better results with reviews from other sub-domains such as automobile reviews and bank reviews (Turney, 2002). Also, one could replace the words “poor” and “excellent” with “conservative” and “liberal” to create a system which could read a newspaper article and report which of those two categories that article was most heavily associated.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

One positive element of this system that was not immediately obvious is how it handles negation. As the bigrams are extracted, given the rules developed by Turney (Turney, 2002), negation words such as “not” are extracted and used in the computation of that document’s SO. For example, the bigram “not interested” receives a much more negative SO than the bigram “more interested” which appears in a different review.

Turney’s algorithm is easy to implement (low coding overhead), however, this fact also makes it easy to find specific cases that it does not cover. In this way, the generalizability of the system negatively affects its accuracy. The rest of this section will focus on specific problems that we encountered.

While this algorithm takes relatively little time to implement and no time to train, it takes an incredibly long time to issue all of the queries to the Search Engine. In fact, Turney’s experiment on 420 reviews took over 30 hours to run (Turney, 2002)! This has an adverse effect on the scalability of the system.

In a review, it is common for the author to compare the subject that they are reviewing with something else that is well-known in that domain. Adjective phrases that are associated with anything other than the reviewed subject should not have the same impact on the semantic orientation of that review. However, with the current approach, we give equal weight to all adjective bigrams, regardless of the subject of that particular sentence.

It seems intuitively obvious that any system attempting to use the entire World Wide Web as a source of information on a language would have many unforeseeable shortcomings which could be very difficult evaluate. For example, any words remotely related to pornography in a review may have huge weight in the semantic orientation of a review, while very specific technical observations may even be ignored by their relative non-presence on the Web.

Lastly, one reviewer may use language that, in the overall scheme of things, is seen as negative in everyday language. However, if the reviewer is using that language as if it is describing positive events, this algorithm will wrongly classify those phrases as actually being negative. Some examples from our corpus include bigrams such as “incredibly gruesome”, “mighty blows”, and “very painful.” Each of these examples remains ambiguous since it is possible that one person would find these things positive in certain contexts.

6. 2 Bag of Words Pro s and Con s

In the bag of words method there were many problems that were unique to the method because it does not take syntax or word order into account, thus making sense disambiguation impossible. For example if a review were to use the word bad, the system is unable to determine if this word means awful, or refer to an evil person, or poor conditions, or even the slang use of the word that has positive connotations. The system lumps all of these uses of the word bad together.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

Another problem can be seen in the following example taken from the corpus: Why is this analogy in the movie? Because it sounds really cool This is an example of one of the many speeches used in Gladiator to make it characters sound neat. This is a negative review that was wrongly classified as positive. The system picks up on the positive words such as really , cool , and neat. It is unable to see that this reviewer is being sarcastic and does not at all like the analogy used or the other long speeches in Gladiator. The system is unable to pick up on this. Another way a very similar problem occurs is with the word not or other negatives. Because words are not taken in context the system could easily misinterpret phrases such as not good or not bad.

However there are also many Pro s to Lang s approach. It is much faster that Turney s approach and therefore much more likely to be useful to applications such as search engines and navigating the web. It is also very easy to implement and to modify. As long as the training data is correctly set up and the number of categories is set correctly the system can deal with any domain.

7. Improvements

7.1 Ways to improve on Turney s algorithm

1. Do some pre-processing on the corpus to tag Subject-Object pairs. This would allow adjective phrases in certain sentences to be weighted more than others. In this case, adjectives in sentences in which the actual product is the subject would have more weight than adjective phrases in sentences about the author s girlfriend.
2. As Turney himself pointed out, the SO estimator is extremely simple. A much deeper mathematical and statistical analysis of the problem could give us a better estimator, and therefore a better system.
3. Over the next few years, technology is becoming faster, cheaper, and more reliable. It seems possible that in a very short amount of time, we will have the processing power to actually process entire documents for meaning, sentiment, etc., instead of doing a token-by-token discrete analysis.
4. This algorithm doesn t necessarily get to the heart of the problem. It seems as though the semantic orientation of a phrase can be modified for the better by one or more informative subroutines (example, see #1).

7.2 Ways to Improve Results from Lang s Method:

1. Pruning the data

The first step in pruning the data provided much better results. However further experimentation in better methods could further improve the results.

a. TF-IDF Weighting

TF-IDF weighting refers to term frequency/inverse-document frequency weighting. This is an accepted and tested technique. It is based on the idea that the more times a token t appears in a document d, or the *term frequency*, the more likely it is that t is relevant to the topic of d. However, the more times that t occurs throughout all documents, or *document frequency*, the more poorly t discriminates between documents. The TF-IDF weight of a token is computed by multiplying the *term frequency* by the inverse of the *document frequency*.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

b. Minimal Description Length (MDL)
According to Lang the MDL principal provides an information-theoretic framework for balancing the tradeoff between model complexity and training error (Lang). This would involve how to determine the weights of different tokens as well as how to decide which tokens should be left out. MDL seeks to find the optimal balance between simpler models and models that produce smaller error when explaining the observed data (Lang).

2. Better Grouping of data

a. Root Words
Words with the same root are classified as unique tokens. By combining these tokens it might decrease dimensionality and give a more accurate view of sentiment.

b. Punctuation
Uninterrupted punctuation is seen as one word. For example !!!!! is in one token instead of being 5 instances of !. Also !!! is a separate token.

8. Conclusion

During our analysis of each of these approaches, we realized that the two systems differ in the type of review that they will correctly classify. Turney's approach works better on reviews with full sentences written as a concise narrative. These reviews should include little extraneous material. Also, this system will lose any benefit of long strings of adjectives, and all punctuation, since these strings will be parsed into bigrams, and will not be viewed as a single adjective phrase. Lang's approach, conversely, works better on reviews written by the colloquial, informal writer. Sentence structure can be ignored and strings of adjectives do not have a negative impact on the ability of this system to classify a review correctly. However this system loses information from negation words such as "not" and "but".

After analyzing their alternative approaches, it seems as though Ken Lang (Lang, 1995) and Peter Turney (Turney, 2002) developed systems that could be easily combined in parallel to create a more robust system. The output of Turney's system is a ranking, from –1.0 to 1.0, with positive rankings indicating a positive review, and negative rankings indicating a negative review. Lang's approach gives a similarity ranking, in degrees, to pre-trained "positive review" and "negative review" vectors. If both systems give the same rank (positive or negative) to the same review, we can be surer about the correctness of our classification. If they disagree, we can normalize Lang's output by dividing by 360, and compare the magnitude of the rankings (by taking the absolute value) given by each approach, choosing the ranking that is closer to 1.0. This would basically choose the ranking from the system that was "most sure" of its ranking.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

Appendix A

Gladiator Results:

Test 1-5 should have been negative and 6-10 should have been positive

PosAvg = 16.0186606915661
NegAvg = 11.8380473345302

TestReview 1 = Positive
Cosine = 23.2709696439365 Positive Average = 16.0186606915661

TestReview 2 = Negative
Cosine = 13.7287385024832 Negative Average = 11.8380473345302

TestReview 3 = Positive
Cosine = 21.0261787760811 Positive Average = 16.0186606915661

TestReview 4 = Negative
Cosine = 11.5234315064659 Negative Average = 11.8380473345302

TestReview 5 = Negative
Cosine = 12.7149150507662 Negative Average = 11.8380473345302

TestReview 6 = Negative
Cosine = 11.4870277804537 Negative Average = 11.8380473345302

TestReview 7 = Negative
Cosine = 13.3473100260505 Negative Average = 11.8380473345302

TestReview 8 = Positive
Cosine = 14.2941176470588 Positive Average = 16.0186606915661

TestReview 9 = Negative
Cosine = 13.1102578104888 Negative Average = 11.8380473345302

TestReview 10 = Negative
Cosine = 11.2413709596575 Negative Average = 11.8380473345302

Tekken Tag Tournament Results:

Test 1 and 2 should have been negative and the 3-10 should have been positive

NegAvg = 11.8229121304717
PosAvg = 13.1909226170445

TestReview 1 = Positive
Cosine = 14.2503313100448 Positive Average = 13.1909226170445

TestReview 2 = Negative
Cosine = 12.5051490549628 Negative Average = 11.8229121304717

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College

TestReview 3 = Positive
Cosine = 17.0557835789563 Positive Average = 13.1909226170445

TestReview 4 = Positive
Cosine = 15.4480815863922 Positive Average = 13.1909226170445

TestReview 5 = Positive
Cosine = 15.2916255149102 Positive Average = 13.1909226170445

TestReview 6 = Positive
Cosine = 17.4736133305576 Positive Average = 13.1909226170445

TestReview 7 = Positive
Cosine = 18.4138705812283 Positive Average = 13.1909226170445

TestReview 8 = Positive
Cosine = 16.9111365668947 Positive Average = 13.1909226170445

TestReview 9 = Positive
Cosine = 16.9747545634721 Positive Average = 13.1909226170445

TestReview 10 = Positive
Cosine = 13.3273304529849 Positive Average = 13.1909226170445

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 97–108
Computer Science Department, Swarthmore College