View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Ursinus College

Ursinus College
Digital Commons @ Ursinus College

Computer Science Summer Fellows | Student Research

7-22-2016

Latent Semantic Indexing in the Discovery of Cyber-bullying in Online Text

Jacob L. Bigelow
Ursinus College, jabigelow@ursinus.edu

Follow this and additional works at: https://digitalcommons.ursinus.edu/comp_sum

Part of the Communication Technology and New Media Commons, Computational Linguistics Commons, Databases and Information Systems Commons, Social Media Commons, and the Theory and Algorithms Commons

Click here to let us know how access to this document benefits you.

Recommended Citation

Bigelow, Jacob L., "Latent Semantic Indexing in the Discovery of Cyber-bullying in Online Text" (2016). *Computer Science Summer Fellows*. 2.
https://digitalcommons.ursinus.edu/comp_sum/2

This Paper is brought to you for free and open access by the Student Research at Digital Commons @ Ursinus College. It has been accepted for inclusion in Computer Science Summer Fellows by an authorized administrator of Digital Commons @ Ursinus College. For more information, please contact aprock@ursinus.edu.

Jacob Bigelow
Latent Semantic Indexing in the Discovery of Cyber-bullying in Online Text
Mathematics & Computer Science
Mentor: April Kontostathis

Abstract

The rise in the use of social media and particularly the rise of adolescent use has led to a new means of bullying. Cyber-bullying has proven consequential to youth internet users causing a need for a response. In order to effectively stop this problem we need a verified method of detecting cyber-bullying in online text, we aim to find that method. For this project we look at thirteen thousand labeled posts from Formspring and create a bank of words used in the posts. First the posts are cleaned up by taking out punctuation, normalizing emoticons, and removing high and low frequency words. Due to the nature of online text many of the words are misspelled either purposefully or unintentionally so a spell check software is used to check the vocabulary, ensuring spelling variations are accounted for. Using this word bank we create a term by document matrix with each post being its own document. By implementing Latent Semantic Indexing (LSI) a query can be placed to the matrix for posts that could have cyber-bullying content. Then the algorithm is trained by adjusting our methods to clean posts and revising spelling corrections for particular repetitive words. With an established approach to pruning the word bank we test our LSI algorithm on other data sets.

1. Introduction

Cyber-bullying has become a serious issue for young internet users with the growth in the usage of smartphones and social media sites. According to a survey done by the Cyberbullying Research Center more than 80 percent of teens use a cell phone, making them highly susceptible to cyber-bullying (1). Cyber-bullying is harassment done through the use of technology and could be anything from nasty messages to posting hurtful messages on social networking sites. Anytime someone sets out aiming to hurt someone else online would be considered cyber-bullying. Another survey found that one in three young people have even experienced cyber threats while using the internet (1). With the young people around the world experiencing this problem, it is urgent we find a solution. Before we devise a means of stopping this dilemma, there must be a supported technique of detecting the presence of cyber-bullying. Once we have a certified way of measuring cyber-bullying it can be applied to existing technology to help reduce cyber-bullying. The field is relatively new but several methods have already been explored and researched. Some of the existing work involves using a bag-of-words technique to look for a high use of certain curse or identified bully words. We will try a different approach, using Latent Semantic Indexing (LSI) to determine whether or not a message contains bullying. Latent Semantic Indexing is a process that helps show higher order term co-occurrence amongst the words used in a group of texts. The idea is that since LSI is said to bring out the 'latent semantics' between documents, we can search for context rather than just for matching terms.

For our project we look at the online forum Formspring and use each post as its own document. Before we can perform LSI we need to generate a dictionary of words used in the messages. We break down each post and clean it up, looking for spelling errors and normalizing common online conversation jargon. Then a term-by-document matrix is produced that we can run Singular Value Decomposition (SVD) on. This gives us the matrices that allow us to perform LSI. We test varying values to truncate the matrix and calculate precision for two different sized datasets. This method of evaluating for cyber-bullying brings a different approach to the table and brings new information on the 'term relationship' amongst the posts (3). Little research has been done in the pruning of online text term lists, but the tactics used here could prove useful to other researchers in the field.

2. Related Work

Latent Semantic Indexing has been said to bring to light the 'latent semantics' amid a group of texts. Although it was named LSI because of this, that's not exactly what is happening. This algorithm is based off traditional vector space retrieval. This uses the whole term-by-document matrix and a query vector to determine the relevance of each document to said query vector. The query vector is multiplied by the term-by-document matrix which gives a weighted vector that should represent how similar each document is to the query vector. LSI does the same task but instead of just looking at the similarities between texts it also looks at the relationships between the terms, and does so by looking at a more complex matrix. However, even though the matrix has sacrificed simplicity it has become considerably less sparse and gained much more meaning for all of the entries. It has the ability to do this by utilizing a mathematical approach, Singular Value Decomposition. The SVD algorithm breaks down the term-by-document matrix (A) into three smaller matrices, a term-by-dimension matrix (T), a singular value matrix (S), and

a document-by-dimension matrix (D). Let r be the rank of matrix A, and that is the number of dimensions to be used. When we multiply the matrices back together we get the original matrix A, as you can see in equation 1 (3).

(1) $A=TSD^T$

Now that we have the three matrices produced by the SVD we can truncate the matrix by restricting our dimension, r to k. This can be accomplished by simply removing columns k+1 to r from T, rows and columns k+1 to r from S, and rows k+1 to r from D^T. By doing this it is thought to remove noise from the term-by-document matrix lessening the computational time and space needed, and compacting the information contained in the matrix. Now that we have our truncated matrices we can send in another query vector q. We query by multiplying the vector q by the reduced term-by-dimension matrix T_k. Then compare the result to the truncated document-by-dimension matrix D^T but first we have to scale it, using the singular values matrix S. Similarly to Traditional Vector Space Retrieval we find a similarity ranking for each document compared to our query vector. Equation 2 shows how we get these rankings.

(2) $A_k = T_k S_k D_k^T$

This then allows to find all of the weights for each of the documents compared to the query vector with equation 3.

(3) $w = q A_k$

For each set of texts with LSI running on it, a value for k must be found and fit to each dataset specifically. Typically a dimension from 100-300 best suits the data but finding the perfect value for k remains a mystery (3).

3. Methodology

We previously had 13,159 posts from Formspring labeled by Amazon Mechanical Turk (MTurk) for cyber-bullying presence. MTurk is an Amazon web service that allows individuals or businesses to request for paid assistance on a task, only humans can complete. This is helpful for us to label large datasets so that we may know what posts are cyber-bullying and which ones are not. This service is crucial to our research since we have not yet attained a reliable method of detecting online bullying. Each post was evaluated by three people on a simple yes or no for cyber-bullying content. For this project we will use each person's vote as metric for how potent the bullying is, zero being no one thought it was present to three, everyone thought bullying occurred. MTurk gives the labeled data in a XML file which stands for Extensible Markup Language and determines the format the document is, making it easily readable for a human. Due to the nature of online text a large amount of words are misspelled or only used as word for online speak. For example emoticons, which are a depiction of facial expressions created using keyboard characters, are not words but are used to convey emotion. Therefore, emoticons are potentially critical in deciphering the intentions of the sender. We start with scanning each post into our machine and then clean out the XML garbage and normalize frequently used emoticons. This is possible through a simple search and replace, looking for any weird XML code that

accidently ended up in our data. Again using the same method we look for emoticons and set them equal to the written equivalent of that particular emoticon. Our project normalized the majority of emoticons found in the throughout the messages. The faces we included are regular smiley faces (:), :], =D), frowny faces (:(, :[ ), faces that are winking (;)), heart symbols (<3), and faces with a tongue out(:p, :P), as well as any combination of the faces. Then all the punctuation is removed and the post is broken down into single words using spaces as delimiters. Any word one character in length is ignored, this is a typical practice when pruning term lists. Now that the post has been cleaned up we can push it through our projects customized spellcheck.

Once we have each individual word we check to see if it falls on a list of misspelled words (such as hahah or wht) or acronyms (like idk or lol), commonly used in internet speak. If the word does fall into this category it will either be ignored or converted to the more common spelling of the word, depending on the type of misspelling. If the word does not fall into either category it's sent through Language Tools software which is an open source spell checking software. The software allows you to add words to ignore and choose different rules to implement based on how the user wants the dictionary to resolve misspelled words. Of course this approach causes some words to be incorrectly changed but due to the high volume of misspellings from the online conversations. Through trial and error we try to adjust the dictionary to best suit our set of words. By changing the ignored words and turning off a few grammatical rules we find our term list at a desirable length with most of the words conveying meaning. Finally we want to remove all words with a very high frequency and all words with a very low frequency because these words typically don't discriminate what kind of messages they are in. Words like, the or, and don't hold very much information on the context of that post. We get rid of words with very low frequency because if it only appears a few times there probably won't be enough resources for the algorithm to develop good 'term relationship' information. Hence, we remove words on the order of 10^0 for low frequency and 10^3 for high frequency words (4).

Once we have a solid set of words that make up our dictionary we can construct a term-by-document Matrix. Our dictionary makes up one dimension of the matrix and each post is on the other dimension. We mark how many times each word is used in every post and thus we have our matrix. We can send this matrix to a high level mathematical programming language (we used R) that will help perform some of linear algebra necessary to perform LSI. Our computational software can calculate SVD on the matrix giving us our T, S, and D^T matrices. We then truncate the matrices looking for an ideal k, to give us the most accurate term relationships present. First we need to create query vector to send into the algorithm, this is accomplished by simply creating a post which is obviously cyber-bullying. Then this post is sent through the same methods to clean it and spell check it against our personal spell checker. Now that we have our query vector (v), we can multiply the truncated term-by-document matrix found in equation 2. This results in our result vector (w), found in equation 3, with weights for the similarities between the query vector and each post in the dataset. To look for the best k we start k = 50 and increment by 25 going up to 350 trying to find the optimal precision. We use three batch sizes of n=10,20,30 where n is the top matching results from vector w. We measure precision to be the number of votes in favor of cyber-bullying for top matching results divided by total number of votes. For example if n = 10, and 4 of the posts are labeled cyber-bullying then the precision

would be 40% because each post has three votes (3*4)/(3*10) = 12/30 = 40%. Two sized datasets are used, one large set of 5000 posts, and one small set of 500 posts. It should be noted that the term list is created specifically for each set of posts. We find the optimal k for this particular set of parameters.

4. Results

Pruning the term list to an appropriate size is a crucial part of effectively running LSI. Too many words and the matrix elements will lose meaning and too few can give too much emphasis to a term's importance. Without cleaning the posts at all the count starts at close to 40,000 words, but after our process of cleaning and accounting for misspellings it is reduced by over 85 percent to 5,716. Initially when creating my term list it was decided to leave words that were written in all caps as such. Thinking words in all caps may have a particularly strong relationship to cyber-bullying. However, I found that it only seemed to be confusing the algorithm, not understanding the significance of the uppercase word. We changed it to making everything lowercase in order to make more words fall under the same term. I hypothesize that all caps still could hold special significance but due to the low frequency they occur at in my dataset they becoming irrelevant. The figures in table 1 support this theory because it is clear that only a very small percentage of the words in all caps even made it to the final term list.

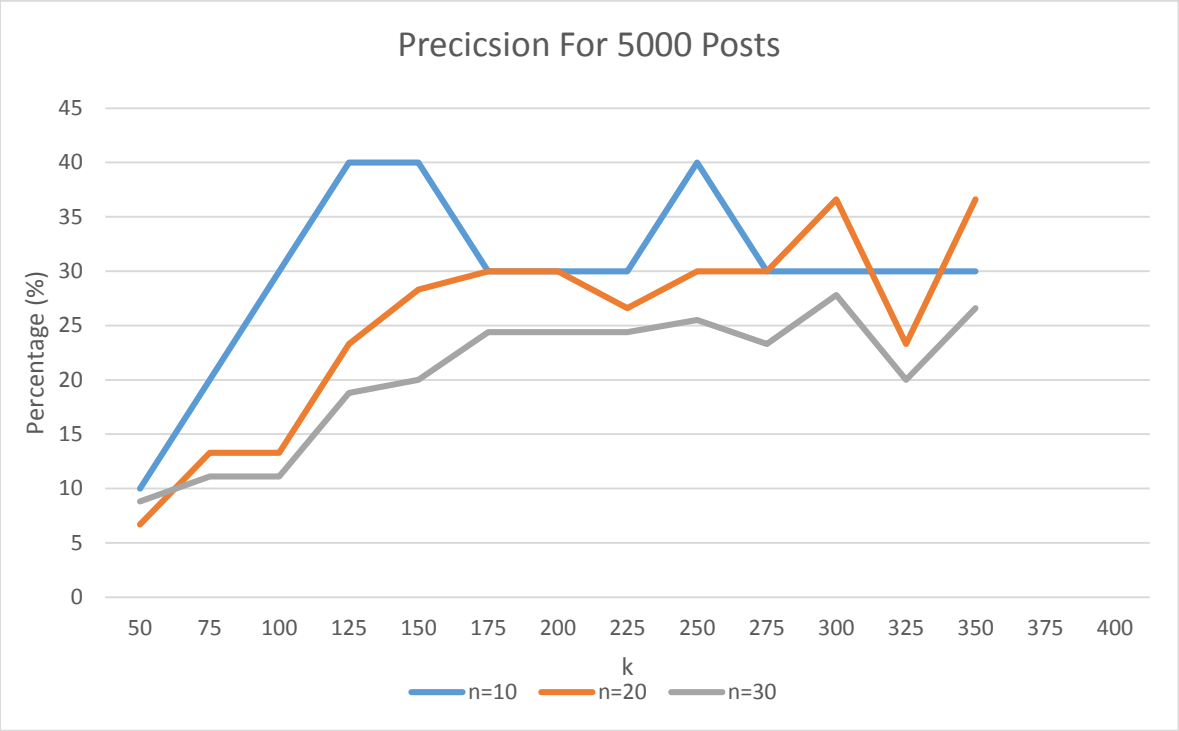
Table 1

Posts	Words Before Case Change	Words After Case Change	Percentage of words remaining
500	538	536	99.6%
1000	876	852	97.3%
2000	1479	1443	97.6%
3000	2043	1989	97.4%
5000	2843	2744	96.5%
10000	4955	4722	95.3%

Due to heavy time consumption needed to generate the matrix and perform SVD we choose two of these sets of posts to analyze, 5000 posts and 500, both after all terms were converted to lowercase.

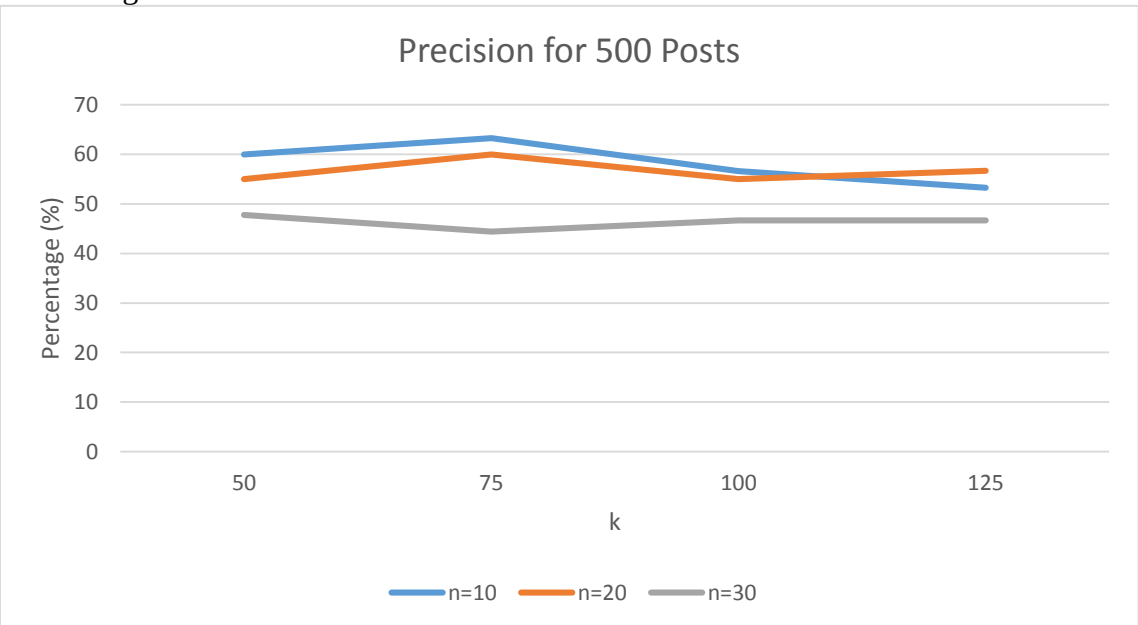
Looking at figure 1 we see the dataset using 5000 posts we test incrementing values of k starting at 50 and going to 300, incrementing by 25. We find that for n = 10 the precision stays pretty level jumping between 30 and 40 percent and continues to do so even up to 1000. For the n=20 we see the precession rise relatively quickly and then levels off and slowly decline. There is a similar trend for n=30 but doesn't rise quite as much as n=20.

Figure 1



With the dataset of 500 posts we got significantly better results with precision ranging up to 63.3 percent. Even for n=30 we get close to 50 percent precision as you can see in figure 2. It is also encouraging to see n=20 and n=10 so closely bound, this shows we are detecting cyber-bullying at a high rate.

Figure 2



5. Discussion

Considering the frequency of bullying that occurs throughout the entire dataset, which is only a small fraction of the whole set, the numbers we are getting for precision are very promising. Even finding 30 percent precision means if we were to have a human checker about 1 in 3 posts they receive will contain cyber-bullying content. This could be incredibly helpful in the fight against stopping this epidemic. Little tuning has been done to the algorithm so it is reasonable to think the program could attain ever better results. We can adjust the size of our dataset, the k values we use, and the batch size to try and improve. Trying different query vectors could also prove quite useful in optimizing the algorithm. It may also be useful to revisit the cleaning and spell checking of the posts to see if we can't catch even more of the internet lingo or emoticons. The work done in this paper shows LSI could be a vital tool in the prevention of cyber-bullying, catching the mean comments before they are even seen. We have found excellent results for exploring the use of using Latent Semantic Indexing to identify posts with cyber-bullying content. With some more tests of different parameters we could pave the way for a new path of stopping online bullies.

6. NSF Acknowledgement

This material is based upon work supported in part by the National Science Foundation under Grant Nos. 0916152 and 1421896. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

7. Bibliography

[1] "Cyber Bullying Statistics." Bullying Statistics. N.p., 07 July 2015. Web. 22 July 2016.

[2] Kontostathis, April, and William M. Pottenger. "A Framework for Understanding Latent Semantic Indexing (LSI) Performance." Information Processing & Management 42.1 (2006): 56-73. Web.

[3] Kontostathis, April. "Essential Dimensions of Latent Semantic Indexing (LSI)." SpringerReference (n.d.): n. pag. Web.

[4] Madsen, R.e., S. Sigurdsson, L.k. Hansen, and J. Larsen. "Pruning the Vocabulary for Better Context Recognition." Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.(2004): n. pag. Web.