***Latent Semantic Analysis***

***Department of Applied Mathematics***

***University of Colorado at Boulder***

***APPM 4720***

***Presented by:***

***Dillon Fancher***

***Motivation/Applications***

In the early years of the Internet, its only purpose was to facilitate the U.S government in attaining a fault-tolerant network of communication through computers. Once the Internet became more accessible to the public with the birth of the World Wide Web in 1989, a new problem developed. Finding relevant information in this massive web was difficult, especially with regard to obtaining specific information. As the quantity of content on the Internet grew, the complexity of traversing the web grew exponentially. This was the beginning of Web Search Engines, who’s future would be unprecedentedly bountiful.

The basic steps for a web search engine are: Web Crawling, Indexing, and Searching. Web Crawling and Searching are two very complicated and very interesting problems on their own but *Latent Semantic Analysis* (LSA) pertains to Indexing. Specifically, LSA shines in indexing large collections of documents and this is where its versatility comes into play. Not only does LSA utilize the keywords of documents, it also examines the ‘Latent’, or contextual, meaning of words relative to the entire corpus of documents as a whole. In essence, analyzing the meaning of a word in this set of documents is not dependent on word overlap. LSA uses linear algebra techniques to construct a vector space known as *Semantic Space* that provides a mechanism to analyze the similarity of any collection of words.

Searching for specific documents or web pages requires a user to specify a string of words for the search engine to query against its indexed web pages. If the indexing process only takes into consideration keywords, the user’s search could be: “car gas prices”, but the computer would not be able to recognize that a web page containing: “automobile, oil, Saudi-Arabia” might be very relevant to the user’s query. Or for example, the search: “coke” might return some rather explicit content when all the user wanted to was see some information on soda. With this regard, one can see the considerable value of *Latent Semantic Analysis* and its ability to contextually analyze the meaning of words.

The extreme growth of technology and the incredibly large amount of data bing produced every second, with regard to social media, poses an interesting question. Where are people talking about similar things? The morphology of language in social media has made this problem even harder to address because unlike 10th grade English class, there are no rules on Twitter, or Facebook. How can a computer learn new acronyms like “*LOL*” and associate them with “*LMAO*”, or “*laughing?”* This can be accomplished by using the context of the word and LSA does exactly that. An experiment was conducted at the University of Colorado, where two students wrote short essays demonstrating their knowledge of scientific topics, which were evaluated by two “expert human researchers” as well as LSA. “The first principal finding that LSA-based measures—which take no account of word order—were as closely related to human judgments as the latter were to each other,” [1].

Having collected over 200 thousand geo-tagged *tweets*  from the U.S.A. this paper addresses the problem: Identify similar tweets accurately, and from there formulate a method to cluster tweets and add in geo-tag for cluster mapping.

***Data: Sampling, Filtering, Cleaning***

To accomplish the goal of determining where people are tweeting about similar topics the sample must be chosen properly. Using the public Twitter Streaming API, which samples the public stream of tweets randomly [2] we are off to a good start. LSA does work for all languages (after tokenization of the symbols), however due to the small percentage of tweets the public Twitter API actually samples some filters were set in place to retrieve a certain subset of tweets:

Tweet Filters:

- Geo-Tagged (200k from U.S.A)

- 300 Mile Radius of Denver, CO. (around 2700)

- English only

When the random sample had been generated, the data was transformed into a format that LSA can use. The data for each tweet was contained in a JSON (JavaScript object notation) document, which provides lots of information unnecessary to this analysis. Using Python, each JSON document was parsed for the geo-tag (latitude and longitude), and the text of the tweet. The text of each tweet was very messy. Using the NLTK (natural language tool kit) module for Python I implemented cleaning scripts that search engines utilize to normalize the text. This consisted of lower casing all letters, removing stop words (words that carry no meaning), deleting words starting with ‘@’, ‘#’, removing punctuation, and tokenizing the text by whitespace. Once these steps had been taken I had a clean set of documents (tweets) that could be analyzed using LSA.

***Mathematical Methods: Latent Semantic Analysis***

The starting point for LSA is constructing a tf-idf (term frequency – inverse document frequency) Matrix :

🡪 for {i = 1, 2, … , m} and {j = 1, 2, … , n}

Where: = ( , ) element of the matrix

= Number of occurrences of term i in document j

= Total # of terms in document j

= Total # of documents in corpus

= Total # of documents containing term-i

Once constructed T is a very sparse, very large matrix. With the random sample of tweets , which means there were 1169 tweets collected within a 300-mile radius of Denver, CO. Now with 2743 unique *tokens* (terms) there is a need to combine terms to create the *Semantic Space* that contains vectors of contextually similar words. In order to get this dimension reduction, first apply the *Singular Value Decomposition* to A.

🡪

Where, are the singular values of A, are the left singular vectors of A, and are the right singular vectors of A. Further are the Eigen-vectors of and are the Eigen-vectors of with being the Eigen-values respectively.

Due to computer accuracy, there are rounding errors and the limitations of machine epsilon prevent The singular values of the tweet data drop off substantially from to = 1.195e-15. Therefore we see there are some linearly dependent columns and we can drop the last 14 singular values and set them equal to 0. Now the condition number of A is the ratio of the largest singular value to the smallest,

🡪 = = 21.245

Now this condition number indicates that A is ill-conditioned and that makes relating documents to one another a very inaccurate system. Namely in order to relate documents to one another, there needs to be a measure of similarity in *Semantic Space* which requires relating document vectors to one another. In order to construct a system to relate documents to one another in a system where the rows are term vectors and the columns are document vectors, we need to calculate:

🡪 , which has condition number 🡪 = = 451.35

Therefore the way the system is now, is very ill-conditioned. Even though the stop words were removed and the text was normalized as much as possible, there is still *noise* in the system creating a large gap between the largest and smallest Eigen-values of

We have a method to reduce this condition number and essentially *de-noise* the system by truncating the SVD of A and then forming a new approximation . To truncate the SVD we delete a certain number of singular values from starting from the smallest, until we get a truncated system that can be recomputed as an approximation A with some acceptable error. That error will be measured using the *explained variance*, which is “how much of the variance in the data is explained by ,” [3]. In this study in order to maintain a good representation of the A, I decided the *explained variance* must be above 97%. With a little bit of testing:

🡪 k = 950 🡪

🡪 k = 875 🡪

🡪 k = 845 🡪

🡪 k = 700 🡪

Thus, k = 950 would enable the explained variance to stay high enough while optimizing the minimization of the condition number of the new model.

🡪 🡪

Now explains 95.21% of the variance in the original data while reducing the rank of the system, effectively:

🡪 = = 451.35

🡪 = = 15.36

TWith this model we are ready to start relating documents to one another. We know that the right singular vectors are the Eigen-vectors of the model and this model relates document vectors in *semantic space* to one another:

🡪

Define L =

🡪 for i , j = 1, 2, … , 1169

Then it is obvious to see that, is in fact the cosine between the vector containing the entries of each Eigenvector of D, and the vector containing the entries of each Eigenvector of D. This corresponds to the cosine between the document vectors in *semantic space* and gives us a percentage of similarity from -1 to 1 because -1cosine . For example:

L(56, 108) = .358

L(1, 267) = -1.73e-18

L(1, 2) = 1.00

L(17,10) = .324

Therefore tweet#42 and tweet#90 are % similar in context to the entire set of documents. Indeed it seems viable for this to be believable as I have pulled the tweets from the CSV file containing the cleaned text:

🡪 Tweet#56: ‘life,coffee,sarcasm,music’

🡪 Tweet#108: ‘sick,driving,need,coffee,music,helping’

🡪 Tweet#1: ‘hollywood,give,little,girls,hope,that,love,love’

🡪Tweet#267: ‘messiah,ronald,reagan,declared,we've,declared’

🡪Tweet#2: ‘hollywood,give,little,girls,hope,that,love,love’

🡪Tweet#17: ‘definately,show,boulder,get,soda’

🡪Tweet#10: ‘yeah,pop,rocks,are,great’

As you can see Tweet#1 and Tweet#2 are the same. That was a control I put in by adding the first tweet twice to make sure I was getting consistent results. As for Tweet#56 and Tweet#108 they are 35.8% similar and it seems to be accurate from a human interpretation obviously from the key words coffee and music being in both. However the contextual power of LSA is demonstrated with Tweet#17 and Tweet#10. They share no words in common yet they have 32.4% similarity according to L(17,10). With further exploration into the data I found multiple Tweets containing both of the words “boulder” and “rocks”, as well as multiple tweets containing both “pop”, and “soda”. So while this is a cool contextual attribute of LSA is also a weakness because LSA cannot capture polysemy. Which is when a word has multiple meanings. Here “pop rocks” are most likely referring to the candy, while the other references of “rocks” with “boulder” most likely pertained to rock climbing, or the rocks in the city of Boulder in Colorado.

We have constructed a well-conditioned, positive semi-definite, weighted matrix that models the connections between documents by a percentage of similarity including itself. We can transform L to aid in clustering the documents by similarity even further by realizing that for i = j, because the cosine between a vector and itself is equal to one always, therefore if we subtract off the diagonal elements of L leaving all of them equal to zero we have a weighted adjacency matrix modeling the documents as if they were part of a complete, undirected, weighted graph.

***Error Analysis***

In the development of this model there were some weaknesses. First Latent Semantic Analysis assumes that have similar meaning will occur in similar places of text. So it would be possible for a document of text to contain lots of contradicting words and poison the corpus of documents being analyzed. Second, the tf-idf matrix is a very sparse and potentially very ill conditioned matrix. When people tweet very similar to almost exactly the same thing the columns of the tf-idf matrix become nearly or completely linearly dependent. This can pose a problem if the proper steps in truncating the matrix with the Singular Value Decomposition are not taken.

Another source of error comes with the limited precision of computers. Lots of the computations such as determining the rank of our tf-idf matrix, A, are based on evaluating the number of non-zero singular values, and the relationships between documents for clustering is measured from 0 to 1. However in computer arithmetic, especially with multiple linear algebra factorizations, 0 almost never registers as a value unless certain tolerances are put into place and the programs manually set a variable to 0. I was able to accomplish this to an extent that I believe the error is minimal, however if the data was changed completely I would need to go back and make sure everything lined up the same way.

The other prominent source for error comes with choice of k for truncating the matrix. This was a very specific problem for the data I was working with, as the geo-tagged only filter limited the number of twitter documents I could reference. If the sample was larger the initial tf-idf matrix A would have had much larger dimensions. Considering that there are on average 50k tweets a second, according to Greg Greenstreet, the VP of engineering at Gnip, recently acquired by Twitter, my sample was extraordinarily small for the total population. This cause my value of k to stay very large as I could not afford very much truncation before the explained variance of the approximation using the truncated system dropped off exponentially. With more documents LSA would have been able to contextualize the meaning of words with greater accuracy, due to a larger reference of places each word pops up, and reduce the rank of the approximation matrix without losing accuracy.

Considering that daunting source of error, I was lucky enough to obtain access to a server on Amazon Web Services that had very impressive computing power. That is one of the reasons the truncated matrix is used, in order to compress the original data into something much easier for a computer to handle and run further computations on. Industry professionals may turn into a truncated SVD with only 300 singular values (as was the value Tim McCandles, Senior Director of Software Development at Oracle suggested was the number his team uses) and save major amounts of computational power later on down the road after indexing. However I was only able to truncate my SVD from 1169 singular values to 950 singular values, before the explained variance dropped well below 97% and I did not want to lose any more accuracy than that.

***Conclusion***

Well the goal of this paper was to develop a tool to aid in the quest of identifying where people were tweeting about similar topics in Colorado. Utilizing the method of Latent Semantic Analysis and using the capabilities of the Singular Value Decomposition, I would say a tool is in place for further analysis of geo-tagging similar *tweeters* in Colorado, and even for any place in the United States with enough time and data. The set of data being analyzed should be larger and better results would follow, but for the limited size of data worked on in this paper I would say that the applications of LSA for relating tweets to one another are accurate and versatile. There are some draw backs to LSA such as its inability to adapt to polysemy, but again with a large enough sample, the contextual representation of words should average out to its actual meaning and placement in relation to other words.

***Bibliography:***

[1]<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.6720&rep=rep1&type=pdf>

[2] <https://dev.twitter.com/docs/api/1/get/statuses/sample>

[3]http://amath.colorado.edu/sites/default/files/2014/03/1352830575/problemset4.pdf