Extracting Meeting Topics Using Speech and Documents

Katherine Brainard, Tim Chang, and Kari Lee
CS229 Final Project

1. Overview

The CALO project is an ongoing effort to develop a Cognitive Assistant that Learns and Organizes. Stanford CSLI is working on the meeting assistant section of the CALO project, which involves recording a meeting, dividing the meeting into topics, and summarizing the meeting for later reference. At this point, automated speech recognition has been used to determine the word distributions of the meetings, but the topics are difficult to extract because the speech transcripts still have a high word error. Our goal is to first correlate a set of documents with their associated meetings based on their word distributions and then improve topic extraction for the meetings by using these clusters of related documents and meetings.

2. Background Information

The corpus consists of forty-six meetings held by different sets of CALO researchers over the course of a year regarding their ongoing project. Thus, many of the meetings discuss very closely-related topics, making the task of distinguishing between separate meetings very challenging. Each speaker used a separate microphone, so there is no confusion about which utterance was made by which speaker. Approximately sixteen of the meetings also have associated documents, which can include e-mail threads, published papers, web pages, attendee notes, and PowerPoint presentations. The size of the document sets range from one to twenty-two, and the total number of documents is around eighty.

3. Preprocessing Transcripts and Documents

One of our initial problems was that our word vectors were extremely large. Because our data is composed of documents and speech, there were a significant number of words associated with each transcript we read. Thus, before assembling our word vectors, we explored various hypotheses on how the removal of certain words would impact the effectiveness of our clustering algorithms. These five hypotheses are as follows.

First, we sifted out all punctuation from the text other than hyphens, changed all letters to lower case, changed all contractions to two distinct words, removed all “stop” words (filler words devoid of content, such as “uh”), and removed non-verbal expressions (such as ‘[laugh]’). This filtration reduces the word vector size, and also reduces the differences between the meeting transcripts and their associated documents. Thus, we hypothesized that this filtration will always improve our clustering.

Second, the transcripts from the speech recognizer could be output in a format that either included or excluded the names of each speaker. Presumably, including the names of the speakers would lead to a high similarity between the transcripts themselves and cause them to group with each other rather than with the documents. Thus, we hypothesized that removing the names would increase the effectiveness of clustering.

Third, we decided to test the effect of stripped “rare” words, those that only appear once in any of the documents, out of the documents. We hypothesized that this smoothing would increase the effectiveness of the clustering by making them more similar overall, which in fact it did. We then tailored the algorithms and our processing of the data to more effectively deal with the disparities between meeting transcripts and documents.

Fourth, we tested the effects of removing numerical digits from the documents and transcripts. We hypothesized that even though documents may cluster on these items, they would reveal little about the topics of the documents and meetings, and thus could lead to adverse effects on our clustering coherency. It turns out that removing digits had a negligible effect on clustering.

Finally, we removed all but the top 50 words from each word vector to see if we could isolate only the most important words to cluster on. Unfortunately, this actually hurt performance, as too much information was lost in this blind eradication of words. The more intelligent removal of words from our general filtering was much more effective in reducing our error rates.

4. Algorithms

The central algorithm to our project was K-means. K-means is a simple coordinate descent algorithm that has been extended and applied extensively in text classification **[1,2,3,4]**. It turns out that K-means is both quick and fairly effective for document and speech clustering. Note that we treated the meeting transcripts as documents, so in this paper we occasionally use the term "document" to refer to all documents and transcripts. A plaintext version of each document was created for every file that was not originally in plaintext.

We tried four main variations of K-means. The first variation was the standard algorithm. The number of clusters chosen was 46, which is the same as the number of meetings in the corpus. The clusters were originally initialized by randomly selecting existing documents and setting the centroids of the clusters to be equal to the selected documents. The algorithm was run until the cluster centroids no longer moved within a tolerance of 0.001. This algorithm worked reasonably, but often developed clusters of either transcripts or documents. Thus, we sought an alternative that would encourage the transcripts to group with the documents rather than with each other.

This was achieved with our second variation. We initialized each centroid to an associated transcript rather than a random document, and proceeded to run K-means from there. This method has two advantages. First, the transcripts and documents are now forced to cluster together, so we didn't have to worry about normalizing spoken text to written text. Second, since the clusters were no longer random and the search space was extremely dependant on the initialization, we could now more accurately compare error rates of the algorithm run on different sets of pre-processed data. However, this method also encouraged documents that start off being clustered in the "correct" cluster to move away from their original location, which increased our error rates.

Our third variation of K-means attempted to take this initialization one step further by fixing the meeting transcripts to a particular cluster. The cluster centroids could change in successive iterations due to the assignment of the documents, but the assigned transcript would in some way anchor the cluster to a particular region.

The last variation of K-means only ran the algorithm for one iteration, which forced the documents to cluster solely based on their initial distance to the different transcripts. The idea behind this method is that it would provide a hard comparison of the difference between each transcript and all the documents, which may be more representative of topical relevance than a result achieved by allowing the system to conduct coordinate descent to some optimal value. In this case, each document would be clustered with its closest transcript without having the possibly of moving away from the cluster due to other similar documents not in its meeting. This last method actually worked well for badly-processed data, but our lowest error rates came from running allowing K-means to run in full on properly processed data with the centroids initialized to the meeting transcripts.

In addition to these four variations on K-means, we also tried two different ways of weighting the word vectors that we generated from our preprocessing. First, we weighted the words in each vector by their TFIDF (Term Frequency Inverse-Document Frequency) scores. The TFIDF weight for word *j* in document *i* is $w_{ij} = tf_{ij} * \log\left(\frac{n}{df_j}\right)$ where *tf* = number of times *j* occurs in *i*, *n* = number of documents in the corpus, and *df* = number of documents in which *j* occurs. This reduces the weight of common words in a word vector and increases the relevance of unique words in determining the similarity of two vectors. Overall, TFIDF focuses the clusters on words that are central to the meeting's topic by discounting words that offer no discriminating value between the documents. Second, we normalized the lengths of both the transcripts' vectors and the documents' vectors, which prevented the documents from clustering to the shortest meeting. These weighting mechanisms, combined with appropriate filtering, allowed us to improve dramatically from the baseline performance.

5. Results

We scored the results of the algorithm as follows: for each meeting, we find the cluster which contains the largest proportion of the documents for that meeting. The coherency error is a measure of cluster purity - the percent of documents in the cluster that are not for that meeting. The density error for the meeting is the percent of the documents for the meetings that are not in that cluster. In the example below, meeting 1 has the largest proportion of its documents in Cluster A. Because the total size of Cluster A is 5, the coherency error is 1 - 3/5 = 2/5. Meeting 1 has 4 total documents, so its density error is 1 – 3/4 = 1/4.

	Cluster A	Cluster B
Meeting 1	3	1
Meeting 2	2	5

Looking at both these error rates gives a better view of how the algorithm is performing than looking at either separately. A perfect clustering would have coherency and density scores of 0. An improvement in density error accompanied by a decrease in coherency error tends to reflect fewer, larger clusters containing multiple meetings, while an improvement in coherency accompanied by a decrease in density tends to reflect lots of small clusters with very fragmented meetings. Neither of these is ideal, so using both measures prevents "optimizing" our clusters with respect to one error value without actually improving the quality of the clusters.

Our main results can be seen in the table below:

Filtering	**Top 50**	**TFIDF**	**Fixed meeting transcripts to each cluster**	**Coherency Error**	**Density Error**
NO	NO	NO	NO	.908	.264
YES	NO	NO	NO	.660	.475
YES	YES	NO	NO	.420	.556
YES	YES	NO	YES	.418	.501
YES	YES	YES	YES	.405	.548
YES	NO	YES	YES	.381	.443
YES	NO	YES	NO	.366	.443

The first row of the table is our baseline; the very low density error is a result of everything basically clustering into one cluster, which can be seen in the high coherency error. The last row of the table represents the best score we achieved; while the density error is worse than the baseline, it reflects a much higher degree of clustering by topic, as shown by the significantly reduced coherency error. We can see that filtering and TFIDF improved both the coherency and density error of our clustering. However, it came as a surprise that allowing meeting transcripts travel between clusters actually improved the coherency error and the density error. This was unexpected because we assumed that allowing for moving transcripts would allow the transcripts to cluster together too much. Apparently the combined use of filtering and TFIDF prevented this from being a significant problem and instead, allowing K-means to run in full actually encourage the transcripts to move towards more similar documents rather than each other.

As a whole, the error rates from our table seem to be very high. However, we made a couple assumptions on our metrics that are slight approximations to what we want, so at some point further reduction of these error rates would deviate from our true goal. First, although these documents were assigned by meeting participants to their associated meetings, there may be documents from other topically related meetings that may also be beneficial. Thus, direct correlation between meetings and documents may not always provide us with the best results. Second, these error scores are artificially raised because many of the meetings had the same documents associated with them, and clearly the same document can't cluster with two different meetings. Finally, some documents may be tied only tangentially with a meeting, perhaps with some topic that was only touched upon briefly in the meeting. Thus, these documents, although they should belong to that meeting, could be classified elsewhere because their primary topics may not align with the primary topics from the meeting.

6. Qualitative Topic Extraction

Having seen the results and limitations of our clustering algorithm, we decided we needed additional information to determine if the clusters were actually being grouped by topic correctly. Unfortunately, we could not come up with a great quantitative measure of topic relevance other than the ones we have already trained on. Thus, in order for us to observe the topic cohesion of each cluster, we developed four qualitative metrics to determine the effectiveness of our algorithm. The clusters themselves are word vectors that represent the centroids of the documents and meetings that belong to that cluster. Thus, their word frequency values are used to determine what words are most representative of the documents in that cluster. An example of all four metrics for each cluster in presented in **[B]**. These metrics also are a very crude form of topic extraction that could be used for future extensions.

First, we determined the most similar words in each cluster. Similar here is a misnomer; it actually refers to the similarity in the TFIDF value of the word. We chose the words in each cluster that had the smallest squared distance between all the documents and meetings in that cluster. This was then weighted by the frequency of the word to avoid the fact that words with small frequencies would all tend to cluster together (since they very little variation).

Second, we determined the most different words in a cluster with respect to all other clusters. Again, different here is a misnomer because difference refers to a difference in TFIDF value and not in the actual appearance of a word. We chose these words as the greatest squared distance between the cluster and all other clusters, weighted by the frequency of the word to prevent the selection of low-frequency words that may not be representative of the words in this cluster.

Finally, we determined the most common and least common words in a cluster based on the size of the TFIDF value. Therefore, the least common words can actually be very common by number of occurrences if the document frequency is high.

Looking at the results, we found that the "most different words between clusters" gave us the most qualitative coherency of topics for a given cluster. For example, Cluster#20 has words like "recruiters," "people," "hr," "talent," and "employees." When we looked at the actual content of the meetings and documents associated with this cluster, it turns out the meeting was about hiring a new software developer for the team, with associated documents on best hiring practices. Many of the "different words" for the other clusters were found to have similar cohesion in their topics as well. Thus, from a human heuristic standpoint, it appears that our algorithm works quite well on grouping documents and meetings by topic.

Furthermore, it turns out that our method using associated documents to help us extract topics from the meetings despite speech recognition errors is highly effective. Cluster #37 contains the word “wubhub” among its “Most Different Words” list. The cluster consists of a meeting and some of its related documents. Although part of the meeting is spent discussing the website “Wubhub,” the speech recognizer interpreted the term in various ways such as “what pulp.” Nowhere in the meeting transcript does the term “wubhub” appear, but the word was still able to be extracted in the cluster topic from the related documents.

7. Further work

Further work on this topic could be conducted in several ways. First, having more meeting transcripts with uniquely associated documents would be very helpful, since our data set was fairly small. Additional meetings could be used to identify errors caused by quirks in the data and make sure that our filtering does not over-fit the data. Second, using the probabilities associated with each word (generated by the speech recognition system) could also be helpful; this might reduce errors in a situation where the recognizer picks the wrong word, but the correct word has a very similar score. Another algorithm that we could experiment with is fuzzy clustering, where documents can belong to more than one cluster. This might help the solve problem of having documents assigned to more than one meeting. Finally, we could also try PCA to further reduce the noise in the data by reducing the dimensionality of the word vectors.

References

[1] Dhillon, Inderjit, Jacob Kogan and Charles Nicholas. “Feature Selection and Document Clustering.” CADIP Research Symposium. 2002.

[2] Hotho, A, S Staab, G Stumme. “Wordnet Improves Text Document Clustering.” Proceedings of the SIGIR 2003 Semantic Web Workshop. Maryland, 2003.

[3] Karypis, George, Vipin Kumar and Michael Steinbach. “A Comparison of Document Clustering Techniques.” KKD Workshop on Text Mining. 2000.

[4] Zhao, Y. and G. Karypis. “Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering.” Machine Learning. VOL 55; NUMBER 3, pages 311-331. BOSTON: Kluwer Academic Publishers, June 2004.