Relational RSS Clustering Techniques

Richard Roesler
Stanford University
rroesler@stanford.edu

1. INTRODUCTION

There has been an explosion in the amount of available news and current event information on the Internet. 24-hours news outlets and independent bloggers alike flood the wires with a constant stream of data. Though an increasing number of people rely on the Internet as their primary source of news and current events, it is becoming increasingly difficult for users to find what they are looking for. Real Simple Syndication (RSS) provided a means to bring the data directly to the user through easily parsable XML *feeds*. Of course this only changed how users looked for news, not how much news there was.

RSS aggregators, likes *Google News*, were created to learn a user's preferences and only display RSS stories that a user would like. Google News, in particular, has found a great deal of success applying algorithms from the text clustering community toward RSS data. Users are able to choose a story and the search engine provides a list of related stories that the user (hopefully) finds interesting. The potential applications of RSS clustering, however, go well beyond simple aggregation.

In the future, Semantic-Web technologies may be able to actually extract knowledge from a stream of RSS feeds, i.e. find similarities, detect patterns, and infer things that the best human analysts cannot detect. This type of application is well beyond the scope of this project. Instead, we focus on aspect of such an application: sorting large amounts of RSS data. We present here the results of applying standard clustering techniques to the RSS problem, and an analysis of how well these techniques work.

In Section 2 we describe some of the unique features that separate RSS clustering from other text clustering applications. The methods and results of our application are presented in Section 3, along with an extensive analysis of these results. We finish with some suggestions to improve future applications.

2. PROBLEM SETTING

As a subset of the larger text clustering problem, RSS clustering faces many of the same challenges. The number of documents available to cluster and the high dimensionality of their representation make text clustering an unwieldy problem [Beil, Ester, Xu]. Feature Vectors with tens of millions of elements would not be uncommon in such situations. The ambiguity of human language further complicates things. It is frequently difficult to separate semantic similarity with syntactic similarity. Any RSS clustering application would clearly have to address these problems, along with two further complications introduced by the nature of online news media: a rapidly expanding dataset and real-time search requests.

Major news agencies are able to literally make up-to-the-minute updates to their RSS feeds. Not only do we have to deal with an extremely large dataset, but also with a rapidly expanding dataset. Any RSS application needs to be frequently updated to include newly posted stories. In addition, most RSS clustering applications need to be able to handle real-time user search requests. Having to run an hour-long clustering program will be inadequate in most cases.

3. METHODS

3.1 Data Collection and Storage

In order to make our experiment reflect real-world conditions, we decided to collect our data sample from real RSS feeds from a wide variety of sources. Over the span of three weeks, we accumulated 1,091 RSS articles on topics including sports, politics, science, entertainment, business, and headline events (Appendix I). For each article we stored the headline, the article summary, and the full text of the article in our database.

We relied on the traditional *bag-of-words* approach to modeling our feature vectors. We represent each article as a set of words contained in that article and the frequency that word appeared:

$$d_i = \{f_1 w_1, f_2 w_2, ...\}$$

The difference between any two documents can then be calculated as the ordinary Euclidean distance between those documents. The feature matrix of all documents is then constructed with each row representing a document and each column the frequencies of word j.

3.2 Preprocessing

As is typical in text clustering problem, we needed to perform a number of preprocessing steps on our feature vectors before we could begin. At first this was limited to the removal of all symbols, HTML tags, numbers and unwanted characters. Words like "don't" simply became "dont" and hyphenated words were separated.

Initial clustering attempts turned out poorly, however, as common words ("the", "I", "he", etc.) dominate the less frequently used, but more semantically important words. Thus, we added a preprocessing step to remove these context free *stopwords*. At the same time, an implementation of *Porter's Stemming Algorithm* [4] was to reduce the remaining words to their basic stems. This ensures words like "working" and "work" are clustered as the same word. In the end, each document was represented as a 16,943-word vector.

3.3 K-Means Algorithm

[5] showed that *k-means* clustering far outperforms hierarchical techniques in the realm of text processing. While some improvements have been suggested to *k-means*, such as *bisection k-means* [1], we decided to use the ordinary *k-means* on order to make use of efficient clustering software. In order to try a variety of parameters, clustering was performed with cluster sizes of *k=6 - 12, 25, 60* and *80*, each time being run to convergence at least 20 times.

We tested three different distance measures for the clustering algorithm: ordinary Euclidean distance, Pearson Correlation, and Spearman's Correlation. Through experimentation we found that the Euclidean distance provided the most logical clusterings and was used for all our testing. Although not tested here, [1] also suggests using the cosine angle to determine the difference between stories.

3.4 Evaluating Accuracy

The difficulty in characterizing the accuracy of text clustering applications is that the idea of "right" and "wrong" is ambiguous. Using a set of data that did not belong to some well-known database, like the *Reuter's Corpus*, meant we had no standard with which our results could be compared. In the face of this, we define the concept of cluster coherence to qualitatively speak about solution quality.

We define *Term Coherence* as a measure of how tightly a specific term (or group of terms) is clustered, regardless of context. For example, with *k=12*, over 82% of the articles containing the name “Obama” appeared in one cluster. We say that this cluster has high term coherence. For news stories, this corresponds to how well a cluster represents a theme or genre.

Story Coherence refers to how well a cluster represents a specific story. With *k=60* we found that the name “Obama” was split into 9 of different clusters, but each cluster represented a different series of events. One cluster, for example, contained 91% of the stories involving President Obama’s policy on Afghanistan, while another contained 90% of the stories about President Obama’s visit to Japan. These clusters have high story coherence at the expense of term coherence.

We represent this visually in Figures 1a and 1b. Each color represents the number of times “Obama” appeared in a cluster. We see that for *k=12* that the vast majority of occurrences appeared in a single cluster, representing a high term coherence. Likewise, for k=60, we see a larger number of clusters, but looking at the headlines of each would reveal each cluster to be more topically correlated.

In order to get a more quantitative view of accuracy we will use two more measures. The first is ratio of the maximum cluster size to the mean cluster size. Solutions which are able to distribute data into a number of medium sized clusters is more desirable than one which creates many tiny clusters and a few extremely large clusters. The second is the norm of the Mean-Squared Error for any story in a cluster. The further away a story is from the “average story” the less likely it is to fit in that cluster.

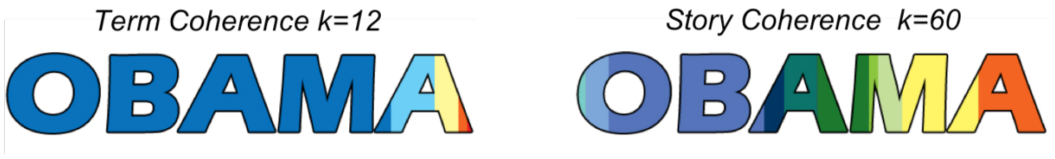

Figure 1a and 1b: A smaller number of clusters results in higher term coherence, while larger numbers of clusters yields higher story coherence.

4. RESULTS

After running the k-means algorithm at a variety of clusters sizes we found that *k=12* and *k=60* generated the results with highest coherence. From the 12 clusters we identified ten as belonging to one of the categories: politics, crime, sports/games, and entertainment. Of those, we noticed that three clusters corresponded strongly to a specific story. One cluster had 80% of the stories about the Fort Hood shooting, another had 100% of the stories about Hurricane Ida, and the third had 85% of the stories about the musician Rihanna and her domestic abuse case. That is to say, 12 clusters give us a solution with high term coherence, but low story coherence. When we increase the number of clusters to 60, we get a very different result. We found 28 of the 60 clusters to have discernable categories, with 25 of those representing specific stories. In other words, 60 clusters yield a solution with high story coherence, but lower term coherence.

From this we can say that the number of clusters roughly corresponds to the “resolution” of our solution. Fewer clusters represent higher-level groupings with high term coherence, while more clusters result in a more story-centric view of events. This fact could be leveraged in future applications by first breaking the problem into larger themes and then breaking each theme into individual stories, through reapplication of the clustering process. An interesting result was also noticed in our above results. Stories which had highly repeated, very unique terms, like “Hurricane Ida” could be correctly clustered even with small numbers of clusters. So it may not always be necessary to cluster with large *k* to get results about a specific story.

Unfortunately, both cases above suffer huge polydispersity in the cluster size. In fact, for *k=60* the largest cluster is 440 elements. That is 24 times the average cluster size! The reason for this is simple to understand. Most stories will fall within some average range of our vector space. Features vectors that lie far outside the average are easier to separate than those within. Thus, as *k-means* runs, each cluster centroid is pulled toward a group of stories on the extremities. All the remaining stories in the center end up grouped into one large cluster.

The obvious next step was to improve our results by normalizing all our vector data, and thus reducing the effect of magnitude on our clusters. Again, we ran *k-means* with *k=12* and *k=60*, this time on our new normalized data. Immediately we saw that cluster sizes were more consistent. In fact, the largest cluster size for *k = 60* was only 3.68! We found only 20 of the 60 clusters had discernable themes, a decrease of 13.3%. What is interesting, however, is that the clusters without strong correlations still were better, on average, than those in the non-normalized set. What we saw is that clusters contain two or three different unrelated stories, but each story was very strongly grouped.

In general, errors in clustering solutions can be fixed if they can be detected. For the relatively small amount of data we used this could be performed by hand, but for any realistic problem this needs to be done automatically. Calculating the Mean-Squared Error for each story proved to be a good way to detect how poorly a story fit in a cluster. If a story had an MSE significantly greater than the others in the cluster, we found with good probability that this story did not belong in the cluster. Iteratively removing the stories with high MSE's provides a simple method for finding cluster quality.

5. CONCLUSIONS

There are a number of promising avenues that should improve results significantly. Particularly research focusing on comparing semantic meaning, instead of just word frequencies has given good results. [3] has used the *WordNet Ontology* to find synonym and hyponym relationships between documents. Applied to RSS clustering, however, this may not provide the most significant difference. We found that most articles prefer using the same common words, instead of using synonyms. What may prove more useful is the work by [2] into leveraging the taxonomic structure of Wikipedia. Wikipedia has the advantage of being frequently update with current events and people, which are likely to show up in RSS feeds and links between articles typically denote logical links between elements. Thus, articles about "pork barrel spending" could be related to government finance and the 2008 US elections, instead of with stories on the meat industry.

One problem that still needs to be addressed is what t do when new articles are added. Our tests with only 1,100 articles took up to an hour an a half for large values of *k*. Being useful in an internet setting would require near real-time results, which cannot be achieved by re-clustering every time a new article in introduced. Instead, we could cluster a large set of data once and then calculate the probability a new story fits into each of the clusters. If we treat our cluster assignments as "correct labels" for a training set, this could be done with any supervised learning technique.

We have seen that even using the most basic techniques good reasonably good results could be achieved. Defining a set of quantitative and qualitative measures to speak about accuracy allowed us to find a number of places where improvements could be easily made. As work proceeds in the field of text clustering we can expect even greater improvements in both accuracy and speed.

RESOURCES:

[1] Beil, F., Ester, M., and Xu, X. 2002. Frequent term-based text clustering. In *Proc. of the Eighth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining* (Edmonton, Alberta, Canada, July 23 - 26, 2002). KDD '02. ACM, New York, NY, 436-442.

[2] Hu, J., Fang, L., Cao, Y., Zeng, H., Li, H., Yang, Q., and Chen, Z. 2008. Enhancing text clustering by leveraging Wikipedia semantics. In *Proceedings of the 31st Annual international ACM SIGIR Conference on Research and Development in information Retrieval* (Singapore, Singapore, July 20 - 24, 2008). SIGIR '08. ACM, New York, NY, 179-186.

[3] Andreas Hotho, Steffen Staab, Gerd Stumme. 2003. "Ontologies Improve Text Document Clustering," icdm, pp.541, Third IEEE International Conference on Data Mining (ICDM'03).

[4] Porter M 1980. "An Algorithm of Suffix Stripping". Cambridge England.

[5] Steinbach M., Karypis G., Kumar V. 2000. "A Comparison of Document Clustering Techniques", *Proc. TextMining Workshop, KDD.*

ACKNOWLEDGEMENTS

Cluster 3.0 by Michael Eisen of Stanford University.
http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/software.htm

Stop Word list from the Journal of Machine Learning, MIT CSAIL:
http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

APPENDIX I: RSS SOURCES

MTV Entertainment and Music News (http://www.mtv.com/rss/news/news_full.html)
Reuters Political News (http://feeds.reuters.com/reuters/politicalNews)
Reuters Science News (http://feeds.reuters.com/reuters/scienceNews)
Reuters Technology News (http://feeds.reuters.com/reuters/technologyNews)
BBC Business News (http://newsrss.bbc.co.uk/rss/newsonline_world_edition/business/rss.xml)
ESPN Sports News (http://sports.espn.go.com/espn/rss/news)
CNN Top Stories (http://rss.cnn.com/rss/cnn_topstories.rss)
CNN Entertainment News (http://rss.cnn.com/rss/cnn_showbiz.rss)
BBC Headline News (http://newsrss.bbc.co.uk/rss/newsonline_world_edition/front_page/rss.xml)