CS229 Final Project: Clustering News Feeds with Flock

Ari Steinberg - ari.steinberg@stanford.edu

December 16, 2005

1 Introduction

1.1 The problem - information overload

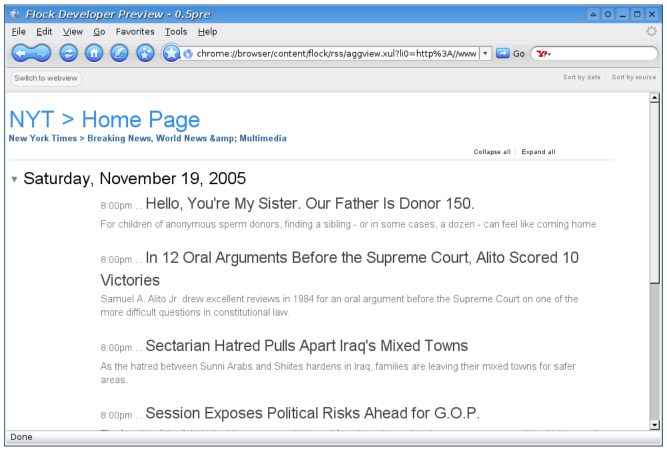
The rise of blogging and RSS (Really Simple Syndication) have created a more personalized way of reading the news, with a much richer diversity of information and perspectives. Unfortunately, though, the rapid growth of these technologies has created a new problem - how to handle the vast amount of information now available. Unfortunately, while RSS aggregators have helped to bring all of the information to one place, they do not solve the problem of filtering through this huge amount of information. This problem tends to get worse as the reader begins to follow more and more blogs, and it is now common for blog readers to spend several hours every day going through all these stories. Many bloggers have discussed this problem; Google's blog search tool shows 24,808 results for "information overload" [1].

1.2 Previous approaches

While most aggregation sites have thus far ignored this problem, there have been some attempts to address it. For example, SearchFox[2] attempts to assign a score to each new article based on the words it contains and how they compare to the reader's past reading habits, theoretically allowing the reader to ignore low-scoring articles that the filter predicts will not be interesting. While this works in some situations, accurately modeling the user's interest level would be extremely difficult, especially given the many outside influences that cause readers to change their habits over time. Expecting such a system to work perfectly would be naive, and if it does not work perfectly the user risks missing stories he may be interested in that have been assigned a low score. This approach also has the risk of "losing serendipity": a user will only see articles similar to the past articles he liked and will never see any new topics that might be interesting to him. It is also undesirable in that the user needs to train the filter.

Another, more promising approach to this problem is to display articles clustered according to topic. While this will not reduce the number of articles displayed, it allows the user to quickly filter entire sets of articles on topics that are not interesting to him without having to make any impossible predictions about what topics the user likes. This approach has been successfully applied on many news sites[3] as well as a popular new blog-reading site called Memeorandum[4]. However, these implementations all have a fixed set of sources from which they display articles and do not allow the user to specify the sources he is interested in monitoring, thus losing the benefit of personalization RSS has brought us.

[1] `http://blogsearch.google.com/blogsearch?q=information+overload`
[2] `http://rss.searchfox.com/`
[3] e.g. `http://news.google.com/`
[4] `http://www.memeorandum.com/`


Flock Developer Preview - 0.5pre
File Edit View Go Favorites Tools Help
chrome://browser/content/flock/rss/aggview.xul?li0=http%3A//www
Go
Switch to webview
Sort by date
Sort by source
Collapse all
Expand all
Done

NYT > Home Page

New York Times > Breaking News, World News &amp; Multimedia

Saturday, November 19, 2005

8:00pm ... Hello, You're My Sister. Our Father Is Donor 150.

For children of anonymous sperm donors, finding a sibling - or in some cases, a dozen - can feel like coming home.

8:00pm ... In 12 Oral Arguments Before the Supreme Court, Alito Scored 10 Victories

Samuel A. Alito Jr. drew excellent reviews in 1984 for an oral argument before the Supreme Court on one of the more difficult questions in constitutional law.

8:00pm ... Sectarian Hatred Pulls Apart Iraq's Mixed Towns

As the hatred between Sunni Arabs and Shiites hardens in Iraq, families are leaving their mixed towns for safer areas.

8:00pm ... Session Exposes Political Risks Ahead for G.O.P.

Figure 1: Flock's aggregator view

1.3 Our approach

We approach this problem with a combination of clustering and aggregation. Unlike existing news clustering sites, we allow the user to select which feeds are to be clustered. The clustering will enable the user to quickly filter out topics he is not interested in, and by restricting our clustered view to the set of feeds the user has specified, the user is guaranteed to be shown all of the articles from the sources he is interested in. However, restricting our clustering to only a small set of sources presents some additional challenges that we will have to address.

2 Implementation

2.1 Flock

Rather than create a new RSS aggregator from scratch, we implement our feed clusterer on top of an existing, open source aggregator - the one built into the web browser Flock[5]. Flock is a desirable base platform as it is built upon the Mozilla platform and thus has been designed to enable a great deal of extensibility, while unlike Firefox it ships by default with an aggregation view. Figure 1 shows Flock with its current aggregation view. Because Flock is capable of displaying any collection of feeds in one aggregation view, we would ideally like to do the clustering "on the fly" (since we do not know what view the user will look at ahead of time). Thus, speed is important, so we will often comment on how various algorithmic decisions impact this in our analysis below.

2.2 Clustering algorithm

We implement several clustering algorithms including K-Means, Expectation-Maximization (EM), and several variants of Hierarchical Agglomerative Clustering (HAC). Our implementation of EM uses a Mixture of Gaussians model.

HAC (described in [2]) starts with each article being placed into its own cluster and greedily chooses the closest pair of clusters to merge one at a time. To select the closest pair of clusters, there are numerous options, including the base similarity metric (for which we implement Matching, Dice, Jaccard, Overlap, Cosine, Information Radius, and L1 Norm, all as described in [2]) as well as how to update the similarities when merging the clusters (we implement the single-link and complete-link variations).

Ignoring for the time being the differences in results (which will be discussed in section 3), HAC has some advantages that make it more desirable for our task than the more traditional EM and K-Means approaches. Most

[5]`http://www.flock.com/`

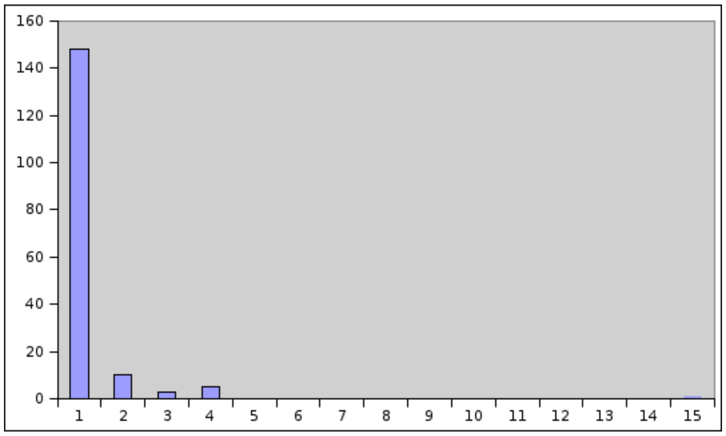


Figure 2: The distribution of cluster sizes for a human-labeled set of 212 blog posts and news articles spanning various topics and 2 days. There is one cluster with 15 posts representing a particularly hot topic at the time (Yahoo's acquisition of del.icio.us), while the rest of the clusters all contain under 5 posts. There were a total of 167 clusters.

importantly, HAC does not require a preset number of clusters - since it is hierarchical, it will continue to merge articles until they have all been connected into one giant cluster. Of course, putting everything in one cluster is not at all useful, but the agglomeration can be stopped at any point along the way when either the desired number of clusters is reached or when all of the remaining pairs of clusters have low similarity values. In order to make comparisons between the many similarity metrics as well as to EM and K-Means easier, we "cheat" by hard-coding the correct number of clusters beforehand, but with HAC it would be easy to tune the similarity cut-off appropriately. This should if anything help EM and K-Means far more than HAC, since selecting the appropriate number of clusters is a much harder process for EM and K-Means.

It is also important to note that unlike typical clustering problems where the goal is to make a relatively small number of large clusters with lots of documents in each cluster, in our problem we expect to see a large number of clusters (since in general there tend to be a huge number of completely different topics with articles written about them) with the majority of clusters having only one document each. Only the top stories of the day should have more than a couple documents in their clusters. A human-labeled clustering of 212 blog posts and news articles resulted in 167 different clusters, as shown in more detail in figure 2.

As a result of these properties of our problem, HAC, traditionally assumed to be the slower algorithm, tends to perform significantly faster than EM and K-Means. While it is possible to implement HAC with runtime quadratic in the number of documents, even our simplistic $O(n^2 f + (n-k)n^2)$ (where n is the number of documents, k is the number of clusters and f is the number of features per document) implementation is significantly faster than our EM and K-Means implementations, which are $O(knf)$ per iteration. This is because in most applications k is small enough to be ignored whereas for us k and f are both comparable to n, and because the constant factors tend to heavily favor HAC; the other algorithms must do many random restarts to select the appropriate number of clusters. We also implement a variant of K-Means in which we precompute the similarities of the documents and avoid ever explicitly computing the centroids (using a trick described in [3]). This helps to speed up K-Means, but HAC is still noticeably faster.

2.3 Feature selection

Feature selection is another important area to explore for this problem. The most basic approach to this problem is to use all of the text in the feed (after stripping out HTML and splitting words on punctuation and spaces) and to use a standard binomial or multinomial bag-of-words model. However, we improve upon this using stop words and also experiment with stemming, using bigrams, and different ways of weighting certain features. These weighting factors can include IDF as well as boosting[6] the weight of words depending on where they occur (such as if they are in the title of a post instead of the body). Finally, while we generally strip out HTML, it can be useful to include any URL's linked to in the feed, since often blog posts will include links to the document they are commenting on, which can be a dead give-away that they are addressing the same story.

One problem is that web sites have a great amount of variety in the amount of information presented in the feed itself; while some sites present the entire contents of their articles in their RSS feed, others provide only a short snippet. Some provide feeds containing full HTML including links and images, while others provide a feed with only plain text and all of the links removed; and worse some contain embedded text ads. CNN's feed often uses "Read full story for latest details" as the entire body text of an article! Thus, just using the content provided in the feed may miss out on some of the details that we would like to capture, so it could be useful to crawl the contents of the site linked to in the post. We choose not to do this, however, since it would significantly slow things down and presents additional problems in terms of the variety of content on the linked pages; it is hard enough to cluster small well-formed XML snippets let alone full documents full of navigational content and often with mal-formed HTML.

3 Results and Discussion

As usual with clustering, evaluation is a tricky problem. Even given a hand-labeled clustering treated as correct, evaluating how close a given clustering is to the correct clustering is nontrivial. We implemented the Rand index and the Jaccard index, both of which are based upon the percentage of $O(n^2)$ pairwise comparisons of the clustered articles that are correct (as described in [1]). Because we have so many clusters with only one article, the Rand index gives highly inflated numbers across the board (almost always above 97% in our tests), so we show results for the more interesting Jaccard index, which ignores pairs that are correctly separated. [1] also finds Jaccard to be superior.

We hand-clustered two separate sets of feeds - one a collection of 212 posts from approximately 50 blogs and news feeds on a huge array of subjects merged into 167 clusters (the one displayed in figure 2); the other a set of 65 posts from 4 mainstream news sites[7]. The first set was used as a development set while the second was held out and used purely to test the final version of the clusterer. To validate that our labeling makes sense, we had two judges label the second set of articles. The Jaccard index between the two judges was fairly high (0.941), indicating that, as we expected, there really is a "correct" clustering that a human can identify.

Table 1 shows the results of running HAC and K-Means on the set of 212 blog posts and news articles described earlier. The Dice and Jaccard metrics perform identically - not too surprising given that they are both based primarily on intersection. Notice that in general the cosine and information radius similarity metrics tend to do best, and in particular they do best using bigrams and boosting. Also, in general, bigrams tend to do better with boosting, perhaps because the boosted title words help to reduce the effect of some noisy bigrams. We expect that significant further improvements can be made by tweaking the boosting scaling factors - we chose not to do this to avoid overfitting our data.

The table also makes it clear that K-Means, in addition to running slower, gives far worse results on this task than HAC. This is not entirely surprising, since K-Means is more typically used in a setting where the number of clusters is nowhere near as close to the number of documents. EM ran even slower than K-Means and thus we were not able to obtain complete results. However, preliminary EM results indicate that it too performed far worse than HAC, both in terms of speed and accuracy (one test on the smaller second set of news articles, for example, had a Rand index of 0.757 and Jaccard index of 0.006, compared to 0.989 and 0.439 for HAC using the same features).

Unfortunately, one drawback of using the Jaccard index for evaluation is that it is hard to intuitively understand how good or bad a performance of 0.593 actually is. Anecdotally, the results look reasonable enough to suggest that

[6] Note: when using the word "boost" in this paper I am referring to "upweighting" and not the machine learning strategy used in e.g. AdaBoost.

[7] The news sites used were the front page RSS feeds from The New York Times, CNN, Yahoo News, and the BBC.

	binomial	multinomial no boosting					multinomial boosting				
	multi-	single	complete-link				single	complete-link			
	link		normal	stem	IDF	bigrams		normal	stem	IDF	bigrams
Matching	0.109	0.085	0.234	0.237	0.168	0.271	0.109	0.308	0.375	0.251	0.376
Dice	0.146	0.383	0.219	0.293	0.266	0.213	0.331	0.379	0.301	0.267	0.371
Jaccard	0.146	0.383	0.219	0.293	0.266	0.213	0.331	0.379	0.301	0.267	0.371
Overlap	0.129	0.197	0.222	0.242	0.246	0.407	0.306	0.469	0.506	0.378	0.335
Cosine	0.141	0.266	0.586	0.560	0.413	0.426	0.281	0.424	0.427	0.358	0.582
L1	0.102	0.372	0.457	0.445	0.237	0.176	0.330	0.503	0.503	0.259	0.298
InfRad	0.147	0.349	0.578	0.356	0.316	0.410	0.316	0.509	0.343	0.256	0.593
KMeans	0.084	-	0.123	0.131	0.115	0.128	-	0.118	0.131	0.163	0.119
FastKM	0.095	-	0.173	0.187	0.177	0.166	-	0.191	0.171	0.185	0.178

Table 1: Performance of HAC and K-Means using a variety of different settings. "Single" refers to single-link clustering (only relevant to HAC). "Boosting" refers to upweighting URLs by a factor of 5 and title words by a factor of 3 (these numbers were chosen intuitively and without testing; further optimization could be done here). K-Means is repeated 3 times with 10 iterations each and with the best scores for each of the 3 runs averaged. FastKM is a tweaked K-Means implementation. EM ran too slowly on this data set to get results.

0.593 should at least be somewhat helpful to a reader hoping to cluster his feeds, but more thorough evaluation should be done here.

To test the generalizability of our algorithm, we evaluated the best set of algorithms and features from the earlier results on our testing set of 65 news articles. The Jaccard score here when using complete-link HAC with information radius as a similarity metric, bigrams, and boosted weights for titles and URLs was 0.439. However, it turns out that 4 of the CNN articles contain the body text "Read full story for latest details", even though they are completely unrelated. We believe this problem can be solved in the future by storing a per-feed document frequency count, and using it to downweight terms. For now, though, we try eliminating these 4 stories from the test set, resulting in a Jaccard index of 0.581 - very close to the result observed on our development set and showing that our work generalizes reasonably well.

References

[1] L. Denoeud, H. Garreta, and A. Guénoche. Comparison of distance indices between partitions. In *Applied Stochastic Models and Data Analysis*, 2005.

[2] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*, chapter 8 - Lexical Acquisition and 14 - Clustering, pages 299 – 304, 500 – 507. The MIT Press, Cambridge, Massachusetts, 1999.

[3] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.