Document Clustering for Distributed Fulltext Search

Jinyang Li {JINYANG,RTM}@PDOS.LCS.MIT.EDU
Robert Morris
MIT LCS, 200 Technology Square, Cambridge MA, 02139 USA

1. Introduction

Recent research efforts in peer-to-peer (P2P) systems concentrate on providing a "distributed hash table"-like primitive in the P2P system (Stoica et al., 2001). However, to make P2P systems useful, we need to build a keyword search engine to index the entire document collection in the distributed system. Doing keyword search in a distributed environment poses new challenges for traditional information retrieval techniques. Traditional search engines are inherently centralized; web pages are collected by crawlers and put in a centralized repository. Subsequently, a global index of the entire document collection is built and used for answering keyword queries. There are lots of successful search engines that index the Web, e.g. Google, Altavista etc.

One easiest solution to search in a P2P system is to build a separate centralized search engine like Napster. However, due to the large amount of resources required to index a huge amount of data, this straightforward solution requires too much start-up cost from a single party. Furthermore, it is prone to legal attacks as in the case of Napster's central song title search engine. In comparison, searching in Gnutella is done in a completely distributed fashion where queries are simply flooded to the network. However, this flooding technique is very costly and it has been noted that most of the traffic in the Gnutella network is the flooded-queries.

In Gnutella network, documents are simply distributed among participating node as they are published and downloaded. Therefore, we have to resort to flooding during the query phase as we have no clue where the matching documents might reside. This paper proposes intelligent ways of partitioning documents among different machines to facilitate key word search in a distributed environment. We use a classic machine learning algorithm, Probablistic Latent Semantic Analysis (PLSA) to partition documents according to their "topics" and evaluated its performance using real web site query logs and search results from Google.

2. Topic based Document Partitioning

Ideally, we would like to partition the entire document collection according to different "topics" present. Thereafter, a query like "peer-2-peer systems" will be sent to the partition that is about"computer science". This poses two technical challenges: Firstly, how to partition documents based on "topics"? Secondly, how to classify a query to a particular "topic"? We use classic machine learning techniques to address both problems.

Unlike Cora (McCallum et al., 2000) and other similar systems, we decide to use "clustering" algorithms (unsupervised text classification) to group documents into a small number of clusters to completely avoid human efforts to identify topics and pre-classify training set documents. Traditional "clustering" algorithm groups documents according to their word (term) "similarities". HyPursuit (Weiss et al., 1996) is an indexing system that uses clustering to group similar documents together. These "clustering" algorithms do not explicitly model the "topic" of each document and we can only assume that documents on the same "topic" have similar word occurrences and hence are more likely to be grouped into the same cluster.

The more recent Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 2001) is one generative clustering technique that explicitly models document topics. It models each document as generated from a number of "hidden topics" (i.e. topics whose identity or signature we do not know). Each "hidden topic" has its signature defined as the conditional probabilities of word occurances in that topic class. Classic model fitting algorithm like Expectation Maximization(EM) is then used to fit the model given a set of documents. As a result of the model fitting, we obtain the conditional probabilities of word occurances in each topic class. These probabilities can be used as a topic class' signature to classify previously unseen documents and actual queries into different topic classes.

Because of PLSA's high computation overhead, we can only afford to train with a small subset of all web pages for model fitting. In our experiment, we selected the $10,000$ (0.56% of all crawled pages) top ranked web pages according to the well-known Google PageRankalgorithm and ran PLSA algorithm on them. Our rationale is that "authoritative" web pages are more likely to be representative of their topic classes and hence we are likely to get better results by fitting our topic model with top ranked web pages. These web pages are passed through a standard word stopper and a Porter's stemmer before being trained. The rest of the web pages are classified by comparing each web page's term vector with that of each cluster's term vector. A standard cosine measure (i.e. inner product of the two term vectors) is used to measure the similarities between the two term vectors and the web page is assigned to the cluster whose term vector is most similar to itself.

In our distributed search system, there is a designated node (root node) that contains the "signatures" of all document clusters. All documents belong to a cluster reside on a single node where a standard local inverted index is also maintained. When a publishing node wants to insert a document into the system, the cluster "signatures" of all clusters are retrieved from the root node and the document is classified and sent to the appropriate cluster node. All queries are issued to the root node which then returns the conditional probabilities that the query belongs to each cluster using Bayes' rule.[1]. The query is subsequently forwarded to a series of clusters with decreasing conditional probabilities until the user is satisfied with the query results. The root node should be replicated for better performance and this could be done easily as there is little space required compared with what is needed for a global search engine.

3. Evaluations

We obtained a $web.mit.edu$'s search engine's query log in October 2001. The log contains $81,006$ queries of which $40,414$ queries are distinct. We assume that Google runs the "perfect" search engine that we want to compare ourselves against. We obtained the top 100 Google results for each query by replaying the query log to Google with $site : mit.edu$ constraint. We also crawled $mit.edu$ over the same week period to collect over $1,800,000$ non-image webpages. Of all the Google result URLs we gathered, more than 99% of the web pages have been crawled.

Figure 1 shows the average fraction of result URL (out of the top 50 Google results for each query) retrieved as a function of the total number of clusters contacted. Ideally, we would like to contact just one cluster and still be able to retrieve 100% of all the matching results. In reality, we are able to retrieve slightly over 60% of matching result URLs by contacting 5 clusters out of 20 on average. This preliminary system performance is quite encouraging, however, it is far from satisfactory. In particular, we notice that the queries often fail to be assigned to the biggest matching cluster that contains most of the results.

[1]For queries with multiple terms, we use the naive Bayes assumption that given a cluster, all terms in the query are independent of each other

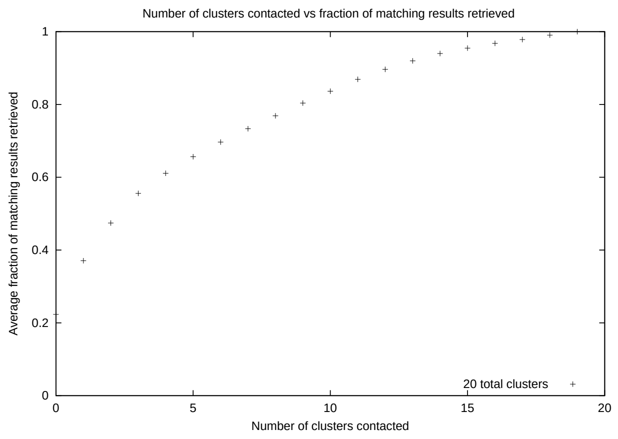

Figure 1. Percentage of matching top 50 Google results retrieved as a function of the number of clusters contacted

4. Conclusions and Future Work

Search in a distributed environment is one important aspect of Oxygen's vision of distributed and pervasive computing. Document clustering based on "topics" holds good promise for efficient distributed fulltext search. We presented the preliminary results of our distributed clustering based search with real web query logs and compared its performance with Google. Future work includes more extensive performance evaluation with hundreds or thousands of clusters. We also plan to introduce a hierarchical organization of clusters and mechanisms to automaticly reconfigure the clusters based on perceived system performance.

References

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning* (pp. 177–196).

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval Journal*.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM*.

Weiss, R., Velez, B., Sheldon, M., Namprempre, C., Szilagyi, P., Duda, A., & Gifford, D. (1996). HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering. *ACM Hypertext*.