Ares Research Semester Synopsis Spring 2006

Clement Pang

Introduction

The semester's research focuses on the use of various information retrieval algorithms to solve the problem of duplicate article detection and irrelevant article detection. Slight adjustments to the system were also made to allow for such algorithms to be used in conjunction with news scrapers.

A full-text OKAPI lookup algorithm was implemented in the existing Web Services infrastructure which checks for similar articles within the same publication. This is necessary since sources like the Associated Press releases new stories as soon as any information regarding a developing story emerges, many of which are corrections or additions to revise previously released material. As such, the Ares news repository would have a large collection of articles pertaining to a particular story which can potentially generate a host of interpreted events if they are not treated as referring to the same incident internally. Since the Ares news repository allows revisions to a particular article through its versioning capabilities, the goal is to group such stories together as one article and retains the latest submission as the most recent revision.

Several machine learning algorithms were explored to detect and tag irrelevant articles in the system. As news scrapers generally collect a large collection of articles from a specific news feed, they would inadvertently include stories which we have no interest in. Examples of such stories include sports articles, economic reviews and local news. Major work was done this semester using the Reuters dataset and tagging certain categories as relevant and leaving the rest as irrelevant. Naïve Bayes, Transformed Weighted Complement Naïve Bayes, K-Means Clustering and OKAPI BM-25 were explored to see their performance in separating the relevant articles from the irrelevant ones. The idea was to add a filter layer to news insertion mechanism to ensure that irrelevant articles are tagged as they enter the system and expensive analysis are deferred on them.

Duplicate Article Detection

The goal of duplicate article detection is to lookup the most similar articles with respect to an incoming article in order to determine if an incoming article looks similar enough to an existing one to be considered as a revision.

Full-text searching has always been a feature in the Ares system and thanks to the additional hardware and storage purchased in Fall 2005, rapid retrieval of articles can be done within a pool of close of 1 million full-length articles. The intuition of using Full-text searching to solve the problem of duplicate articles stems from the fact that the problem itself is no different than doing a full-text search on the database, albeit using a much bigger query vector.

The system is implemented in such a way that any incoming article would go through a simple duplicate filter and, if passes, be fed into the database engine as a full-text query to find similar article matches. The simple filter is a naïve approach to spot obvious duplicates with the same exact headline, date of publication and author. Initially, this worked remarkably well and tagged obvious duplicates in the system, mostly introduced by news scrappers being ran on the same day on multiple sources. However, there also exist a large collection of duplicate articles with slightly different headlines as the focus of the story shifts as it progresses and additional contributors are added as the story develops, making the identification of them harder than it might seem at first.

To maximize performance, a series of preprocessing are done to the query to improve retrieval. Due to restrictions in the full-text engine of MSSQL 2005, the maximum number of terms in a full-text query is limited to approximately 2000 (as it is internally expanded into a SQL query). The system first restricts the length of the article and removes stop words which would otherwise limit the number of input terms. The resultant word vector is formulated into a SQL query and fed into the database engine. The relevancy score returned by the full-text engine is used as a measure of how similar the incoming article is with respect to articles already in the system. A threshold is used to determine whether the incoming article is a duplicate of the article with the highest relevancy score and a simple algorithm is used to rule out cases when the similarity score passes the threshold but is not a single unique match, i.e. there are other articles that seem to match the incoming article just as well and thus the system should not assign the incoming article as a duplicate to a particular article but as a new entry instead. This applies to articles which happen to be similar to a lot of articles and simply picking the most similar article and assigning it as a duplicate can lead to potentially losing articles. The method used is to simply rank the articles returned from the full-text engine and see if the highest ranked article is receives a score that is at least a certain factor higher than the second highest match. In experimental trials where humans are used to judge the validity of the filter, this has shown to reduce false positives.

In a set of 659 articles, a human is asked to verify whether the output of the filter is correct by displaying the incoming article and the article that the filter thinks is most similar to. The results are analyzed dynamically to find the best threshold and factor to use as the “distinctiveness” factor as explained in the previous paragraph. With a threshold factor of 800 and a distinctiveness factor of 0.5, i.e. the second highest match must be at least half the relevancy score than the first; the duplicates that the filter selects are all correct while 5% of the duplicates in the pool are classified as new stories. The filter should try its best not to misclassify an article as a duplicate and not the other way around as articles will be lost if there are false positives. False negatives are analyzed further with NL methods in the Ares Event Data System and will probably result in having the same exact event signature as an event gathered from its duplicate and disregarded. Thus, false negatives can only do so much as to increase processing load of event extraction whereas false positives can lead to losing of event data. Thus the threshold and "distinctiveness" factor is chosen to yield as few false positives as possible.

Relevant Article Detector

After applying the OKAPI full-text algorithm to solve the problem of duplicate article detection, the same method is explored to see whether it can be used to detect irrelevant articles. The intuition here is to give the full-text engine a set of irrelevant articles and a set of relevant articles (from the Reuters set) and see if the filter can distinguish between the two. The method is somewhat ad-hoc as a full-text engine is not designed to perform binomial text classification and thus the training set of articles is stored as two tables with full-text searching enabled and an unclassified instance a search query performed against these two tables. The top results returned from the two queries are summed and compared with each other. Whichever is higher indicates that the unclassified instance is more "similar" to a particular class.

In a set of 1000 articles, the filter achieves a precision and recall of 92.3%. Experiments were further performed using a much larger data set with precision and recall around the 90% region.

A host of other algorithms were explored to see if we can further improve classification performance. Naïve Bayes, N-grams, Transformed Weighted Complete Naïve Bayes and K-Means Clustering were implemented with varying degrees of success but none were as robust and computationally feasible as OKAPI. A lot of time was put into clustering since although it was computationally expensive during the clustering process, the resultant clusters can be compared very quickly to an unclassified instance in order to generate a predicted class. The general idea of the algorithm is use K-Means and cosine angle differences between articles with the addition of separating clusters when they are not "pure", i.e. using the binary nature of relevant and irrelevant articles to separate clusters when they seem to contain too many of both. This is different from traditional clustering algorithms as the end goal is to perform a binomial text classification on unknown instances by matching them to known clusters of relevant and irrelevant articles and having all of them "vote" to give a probabilistic score of whether the unknown class is relevant or not – an improvement on simply relying on the categories of Reuters or a simple numerical comparison of relevancy scores in OKAPI in order to make the call. All of the algorithms mentioned above were implemented as series of database queries and mutations. As such, they were made possible to be used in large databases without a constant worry on memory limits and storage complications with respect to model generation and usage.

Future Work

As a project for COMP540, Benedict further explored OKAPI and was successful in marginally improving the performance of OKAPI in close-cut cases. However, when a fresh set of humanly classified articles from a different publication is used to test the performance of the filters, even OKAPI failed to achieve a reasonable precision. An explanation for why this is the case could be a bias in the training data (the Reuters set) and also the fact that the classification problem itself is inherently difficult where any novel data can easily trip off the algorithm into misclassifying it. In the coming semester, algorithms that reduce empirical risk by finding the optimal hyperplane between two classes such as Support Vector Machines will be explored and the feature space expanded to include features that are computed in additional to word counts, such as the number of countries mentioned, their locality to each other, the number of important political figures mentioned and their locality – knowledge that is not otherwise known to the computer.