From: AAAI Technical Report SS-96-05. Compilation copyright © 1996, AAAI (www.aaai.org). All rights reserved.

Multi-Media Fusion through Application of Machine Learning and NLP

Chinatsu Aone, Scott William Bennett and Jim Gorlinsky
Systems Research and Applications Corporation (SRA)
4300 Fair Lakes Court
South Building
Fairfax, VA 22033-4232
aonec@sra.com, bennett@sra.com, gorlinsk@sra.com

Abstract

This paper describes work on a system to dynamically cluster and present information from live multimedia news sources. Features are obtained by applying statistical and natural language processing techniques to texts associated with the media (new wire stories or closed-captions). Conceptual clustering is employed to dynamically build a set of hierarchical clusters. Results are presented objectively comparing automated clustering with hand clustering of the same articles.

Introduction

With the ever-increasing amount of information available today in printed and electronic form, it is becoming more and more difficult for people to assimilate information from all the available sources. In this paper, we will discuss a prototype system for the Multi-Media Fusion (MMF) project which enables the automated assimilation of such information. End users of this system would greatly benefit from the speed and ease of access to relevant information. The information collected can consist of text, video, sound, and/or photos. The MMF system can automatically fuse CNN Headline News clips with related news articles using a conceptual clustering algorithm based on Cobweb and high-level features derived by a natural language system. The result is a multimedia document where information, whether text or video, on related events is grouped together. Thus, the user can quickly assimilate information of interest by choosing representative articles or news clips from each group.

System Description

The MMF system assumes an environment where the users need to know regularly and in the most efficient way a summary of what happened and what were "hot" topics around the world. The users do not know *a priori* what topics they are interested in. They want to have access to new information when it becomes available. In addition, they want a wealth of information achieved through multiple sources and accounts. This differs from routing tasks where users have personal profiles selected in advance. Therefore, the MMF system must be different from routing systems which are trained by knowing relevant and non-relevant sets of articles.

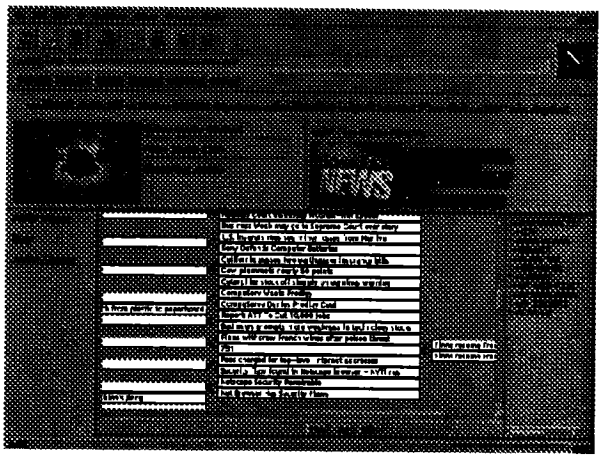

Figure 1: System snapshot

Currently the system uses CNN Headline News and the ClariNet news feed as multiple sources. We decode and utilize the closed-caption text associated with CNN Headline News in order to fuse the video clips with related news articles. The CNN closed-caption text is segmented into different news topics according to the closed-caption format. The video, audio, and captions can be transmitted live while clips are simultaneously being stored and clustered.

The final multimedia document lists clusters of related captions and news articles. The user can select the shortest article or just one of the news video clips as the summary of the clustered information.

Cobweb-based Conceptual Clustering

Since we do not know beforehand the categories into which news clips and articles should be grouped, we cannot use the inductive learning approaches such as C4.5 used recently by (Lewis & Catlett 1994) or FOIL used by (Cohen 1995). Thus, in order to fuse related

CNN news clips and ClariNet articles together into groups of previously unknown "categories," we employed a modified version of the Cobweb conceptual clustering algorithm (Fisher 1987). Cobweb forms a classification tree over input objects using their feature values. It "favors classes that maximize the information that can be predicted from knowledge of class membership." We use this hierarchically organized tree as a representation of fused news.

Our enhancement to the algorithm is motivated by the fact that the original version used a fixed set of features across all examples while it is very difficult for text clustering to determine *a priori* a universal, fixed set of features for any possible corpus. Thus, we represented each document and cluster as hash tables. So, when a specific document instance is added to a cluster, the hash table at the cluster is merged with that of the document, and feature counts are summed.

Training Features

In order to cluster texts by Cobweb, texts must be represented as feature vectors. As an initial step, we decided to use as features keywords picked from the body of a text based upon their worth as determined by the Inverse Document Frequency (IDF) metric (Church & Gale 1995). Two aspects of our use of IDF which are novel and important to this discussion are: how the "words" themselves are chosen, and how the IDF values of those words are calculated and made useful.

It is important to recognize that using white-space delimited tokens in a text as keywords provides significantly less information than is actually available. This is especially true in regard to proper names, such as those of persons, organizations, and places, where the individual tokens of a name may be meaningless individually. To compensate for this difficulty, we use a natural language processing system called NameTag^{TM} developed at SRA. It recognizes names dynamically at high accuracy and speed. It does not depend on a list of finite names, but it can recognize names it has never seen before (like "Kirk Kerkorian") by using contextual clues. This is important because names of people, organizations and places previously unencountered appear quite frequently in real-world texts.

In addition, NameTag can recognized aliases of names such as "IBM" for "International Business Machines" and "President Clinton" for "Bill Clinton" *within* a document. This information is useful, since we can choose one of the possible names of a particular entity to be representative of the entire group of aliases, and in this fashion reduce the chances that aliases will be used as distinct keywords in the same document. This, however, does not solve the problem of aliases appearing in different documents. Work is currently underway on a NameTag name-resolving utility for inter-document name matching, but the results of that work are still pending.

```
>>> OVER A BRIDGE AND  FREEDOM, TWO BUSES
CARRYING 108 U.N. PEACEKEEPERS CROSSED
INTO THE SERBIAN TOWN OF MALI ZVORNIK
ABOUT FIVE HOURS AGO.
```

Figure 2: A CNN closed caption text

To index the closed-caption texts from CNN Headline News, we customized NameTag to handle idiosyncrasies of closed captions. First, we customized NameTag to handle all *upper-case* closed caption texts as shown in Figure 2. Because the standard NameTag assumed ordinary news articles and relied on capitalization information, we added more linguistic information to improve its performance on all upper-case texts. Second, since closed captions are transcriptions of speech by anchor persons or reporters, we accommodated characteristics of such spoken language (e.g. typical closing remarks by reporters like "SEAN CALLEBS, CNN, LOS ANGELES") and eliminated the written-language specific portion of the original NameTag (e.g. "OK" is Oklahoma in a text while it is an answer in a caption.)

The availability of such high-level name features from texts makes our use of IDF values more meaningful, since we have made efforts to both ensure that names are considered as a single unit and that aliases are considered to be the same "word." In addition, we have enhanced the calculation of IDF values on any particular test set of texts by creating a baseline of words and IDF values from five years of text from the Wall Street Journal. Included in this database are all of the multi-words and names which were discovered by NameTag, as well as all of the other non-names from those texts. This provides further assurance that less informative words such as "the" and "of" are not considered good keywords. Further, with such a large database of words available, it is far more likely that full, proper names picked by NameTag will appear as appropriate keywords for any particular article, as these tend to be less frequent than non-proper names. This is good news, as these names tend to be positive indicators of the content of any particular article.

Experiments and Evaluation

Evaluation Method

In order to evalute clustering experiments objectively, we have come up with a method which compares the results of our automatic text clustering to a set of manually clustered texts. The process involves determining the best match for a predetermined cluster in the automatically clustered system, and then scoring that match in terms of how well it encompassed the required texts, and also in terms of how many misplaced articles were also present in that cluster. This gives us measures of both *recall* and *precision* for any particular attempt at clustering, and allows us to identify

Table 1: Experiment results

	Level	Name Weight	Recall	Precision
E-1	5	Unweighted	48.13%	89.11%
E-2	8	Unweighted	48.13%	89.11%
E-3	Unrestricted	Unweighted	29.41%	100%
E-4	5	Weighted	36.36%	85.00%
E-5	8	Weighted	34.22%	94.12%
E-6	Unrestricted	Weighted	29.95%	96.55%

shortcomings in the algorithm and to focus on needed improvements. Since we are dealing with a control set of texts which were clustered by hand, there are no guarantees that the way in which the texts were organized was the best possible way. In fact this is unlikely, as it is very difficult for any two people to organize a large corpus of texts in exactly the same way, much less a machine and a person.

Thus, our test strategy is useful in determining the over-all effect of any changes we make to the clustering process, but it is not exceptionally informative with regards to how we arrive at any particular result. In order to compensate for this, we have developed an interface which allows us to examine the clustering process as it occurs. Developed in JAVA, our interface dynamically displays the contents of the clustering hierarchy and illustrates the processes which occur as the result of each text being added to the hierarchy. This interface has allowed us to begin looking for clues as to what exactly makes a good feature for the text clustering task and why.

Testing and Evaluation

We conducted several clustering experiments with two different parameters using Cobweb and 200 ClariNet texts. First, we varied the number of levels permitted in Cobweb clustering (5, 8 or unrestricted). Then, we weighted names by multiplying their IDF values by their frequency. The results are shown in Table 1. We used the top 10 keywords of each text (measured by IDF values) as training features.

In general, very similar texts were clustered together correctly. For example, two articles "Netscape Security Penetrable" and "Net Browser Has Security Flaws" were clustered together by keyword names like "Netscape," "David Wagner," "Ian Goldberg," "Mike Homer" as well as other keywords like "browser" and "Internet."

In Cobweb, restricting the number of levels results in fewer node-merging operations, and hence a flat tree. In these particular experiments, setting a limit on the number of the levels gives better scores though it sacrifices some precision. It is, however, impractical to set the levels manually because using more or fewer levels may be appropriate for different sets of texts. Thus, we are currently considering using a "reparsing" (Fisher 1995) approach where we do not need to specify the optimal level beforehand but we "massage" the clustering results later to produce a better tree.

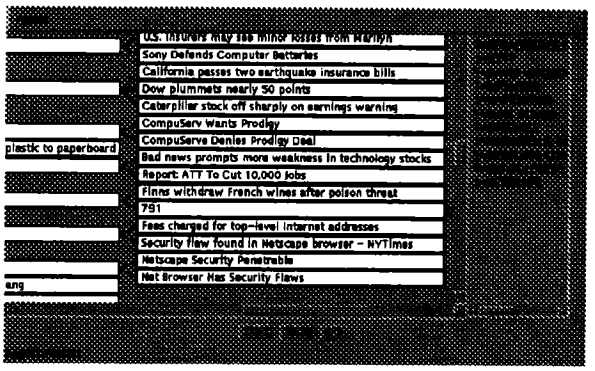

Figure 3: A cluster tree and features

Adding extra weights on the names actually hurts both recall and precision. This is because not all the names are important (e.g., "U.S." and "Clinton" which appears in many texts).

Future Work

The next step is to include the CNN captions in the test set, which were not used in this experiment because we have not manually classified them into a hierarchical tree for evaluation. In calculating IDF values, we plan to incorporate an *inter-text* name resolver, as discussed in Section , to make more accurate assessments of word frequencies. We also plan to improve the way to choose the number of keywords used for each text instead of picking an arbitrary number. For example, we can relativize the number to the word count of the text or use a threshold based on IDF values.

References

Church, K., and Gale, W. 1995. Inverse document frequency (idf): A measure of deviations from poisson. In *Proceedings of the 3rd Workshop on Very Large Corpora.*

Cohen, W. 1995. Text categorization and relational learning. In *Proceedings of the Twelfth International Conference on Machine Learning.*

Fisher, D. H. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2.

Fisher, D. H. 1995. Iterative optimization and simplification of hierarchical clusterings. Technical Report CS-95-01, Department of Computer Science, Vanderbilt University.

Lewis, D., and Catlett, J. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh Annual Conference on Machine Learning.*