A Cluster-Based Approach to Tracking, Detection and Segmentation of Broadcast News

David Eichmann[1], Miguel Ruiz[1], Padmini Srinivasan[1],
Nick Street[2], Chris Culy[3], Filippo Menczer[2]

[1]School of Library and Information Science / [2]Dept. of Management Science / [3]Dept. of Linguistics
University of Iowa
Iowa City, IA 52242

ABSTRACT

We present results of the University of Iowa topic tracking and detection as well as story segmentation efforts. Topic tracking is performed for the "boundaries given" case. The DET curves for all the runs are consistently smooth and concave suggesting no sudden changes in expectation required from the user. The effect of reducing the training size of relevant stories is examined. The detection runs are performed using a "pipeline" model to utilize the advantage of the deferral period. Performance is strongly influenced by the fact that roughly 2000 to 3000 declared topic clusters are generated during the detection runs. Performance is analyzed with respect to changing the cluster threshold. In segmentation, an agglomerative clustering strategy is adopted. The decision to declare a boundary depends upon both lexical similarity of neighboring segments as well as the pause duration. The algorithmic complexity of the method is O(k log k) where k is the number of pause delimited sentences in the file. The tracking, detection and segmentation modules provide a sound framework for future extension and experimentation.

1. INTRODUCTION

The Information Retrieval Group at the University of Iowa participated in all three tracks of the Topic Detection and Tracking effort. Our system architecture is an extension and enhancement of that which we have built for TREC [2], implemented in Java and supporting a variety of task-specific configurations through observer/observable design patterns [3]. This year's submission will function as a baseline for our subsequent work in topic detection and tracking - we have kept architectural and algorithmic details as clean as possible to allow for evaluation of individual enhancements.

2. TRACKING

The tracking system begins by generating two sets of clusters from the training data: one set from the on-topic training stories and the other from the off-topic training stories. A membership similarity threshold (α) controls cluster formation and extension. In order to cut down on the time and space for computations, we introduced a second threshold (β). An off-topic training story must be within β similarity of at least one positive cluster in order to be clustered during this training phase. Any off-topic stories failing this criteria are discarded as insufficiently similar.

Similarity between documents and clusters is measured using a straight-forward vector cosine measure:

$$sim(d, c) = \frac{\sum_{i=1}^{max(N_d, N_c)} W_{id} \cdot W_{ic}}{\sqrt{\sum_{i=1}^{N_d} W_{id}^2 \cdot \sum_{j=1}^{N_c} W_{jc}^2}}$$

where $TF(W_i)$ is the (current) TF×IDF weight for term W_i in a given document or cluster. Term frequencies are built up incrementally as a given run progresses and cluster term weights are adjusted every ten input files. This approach is therefore somewhat inaccurate in the initial phases of a run, but quickly reaches a point of reasonable stability with respect to term frequencies and has the added benefit of requiring no fore-knowledge of the vocabulary. Weights are incrementally updated every 10 input files. All vocabulary is stemmed using Porter's algorithm [4] and filtered through a stoplist. We prune document term vectors to the 100 most weighty terms and cluster vectors to the 200 most weighty terms. This proves to have no significant effect on the accuracy of our results, but a significant effect on both memory requirements and execution time, the latter due to a corresponding reduction in the cost of dot product calculations.

During testing, a new story must first qualify for consideration by having a maximal on-topic similarity that exceeds its maximal off-topic similarity. Any story failing this criteria is declared a non-match for the topic with a confidence equal to its maximal on-topic similarity. If the maximal match is on-topic a second level criteria is applied. If the maximal on-topic similarity is above α, the story is declared relevant, at or below α, the story is declared non-relevant, in each case
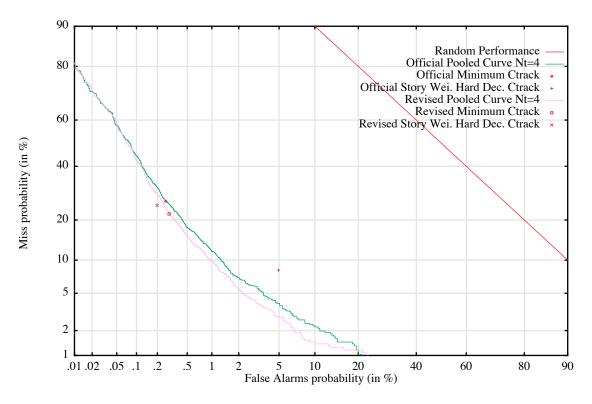
Figure 1: ASR Tracking Results, Nt = 4, α = .25, β = .20

Table 1: Official Tracking Results, α = 0.25, β = 0.20

	Story Weighted			Topic Weighted		
	P(Miss)	P(Fa)	C_{track}	P(miss)	P(Fa)	C_{track}
Official asr, Nt=4	.0821	.0493	.0500	.1460	.0425	.0446
Official man_ccap, Nt=4	.2335	.0018	.0064	.2531	.0018	.0068
Corrected asr, Nt=4	.2476	.0020	.0069	.2639	.0020	.0072

with a confidence equal to its maximal similarity. The clusters generated with the training data remain unchanged throughout the test phase.

2.1. Analysis of Results

Our initial shakedown runs involved only α thresholds in effect at a very low setting (0.1). We then post-processed the result files to toggle yes/no decisions at a variety of α values, with optimal results in the range 0.20 - 0.25. We then tested the following parameter combinations using the development data:

1. α = 0.25 and β = 0.20;

2. α = 0.20 and β = 0.15.

Unfortunately, due to human error the results became interchanged and we selected the second set of parameters instead

of the first. An additional programming error was introduced during the asr runs that overrode α in a key declaration to 0.1. Our official results were therefore less than stellar… Table 1 shows both the official results and the results obtained with the declaration error removed. Figure 1 show the original and corrected DET curves for ASR, Nt = 4.

We have subsequently completed a full set of tracking runs for the 'boundaries given' case, with results shown in Table 2. Tthe corrected results lead us to believe that even simple algorithmic approaches can perform well when customized for the specific task. The DET curves for all the runs are consistently smooth and concave suggesting no sudden changes in expectation required from the user.

As expected, reducing the amount of relevant information used, from 4, to 2 to 1 relevant document worsens the C_{track}. However comparison of the DET curves is interesting. For all three types of sources the highest P(Miss) value on the curve drops while the lowest P(Fa) value rises as fewer rele-

Table 2: Tracking Results

Source & # Training Examples	Run	Story Weighted			Topic Weighted		
		P(Miss)	P(Fa)	C_{track}	P(Miss)	P(Fa)	C_{track}
ASR 4	$\alpha = .25, \beta = .20$	.2476	.0020	.0069	.2639	.0020	.0072
	$\alpha = .20, \beta = .15$	.1758	.0040	.0075	.1765	.0037	.0071
ASR 2	$\alpha = .25, \beta = .20$	.3040	.0014	.0074	.3814	.0013	.0089
	$\alpha = .20, \beta = .15$	.2154	.0031	.0073	.2441	.0027	.0076
ASR 1	$\alpha = .25, \beta = .20$	.4176	.0010	.0093	.4618	.0009	.0101
	$\alpha = .20, \beta = .15$	.2586	.0021	.0072	.3967	.0020	.0099
CCAP 4	$\alpha = .25, \beta = .20$	.2474	.0018	.0067	.2637	.0018	.0070
	$\alpha = .20, \beta = .15$	.1618	.0034	.0065	.1844	.0031	.0067
CCAP 2	$\alpha = .25, \beta = .20$	.3133	.0013	.0075	.3542	.0012	.0083
	$\alpha = .20, \beta = .15$	.1669	.0029	.0061	.2288	.0025	.0070
CCAP 1	$\alpha = .25, \beta = .20$	.3785	.0009	.0084	.4319	.0008	.0094
	$\alpha = .20, \beta = .15$	.2225	.0018	.0062	.3222	.0018	.0082
FDCH 4	$\alpha = .25, \beta = .20$	.2248	.0018	.0063	.2483	.0018	.0067
	$\alpha = .20, \beta = .15$	.1606	.0034	.0065	.1859	.0031	.0068
FDCH 2	$\alpha = .25, \beta = .20$	.3036	.0014	.0074	.3449	.0013	.0082
	$\alpha = .20, \beta = .15$	.1672	.0029	.0062	.2316	.0025	.0070
FDCH 1	$\alpha = .25, \beta = .20$	.3759	.0009	.0084	.4296	.0008	.0094
	$\alpha = .20, \beta = .15$	.2226	.0018	.0063	.3231	.0018	.0082

Table 3: Pipeline Effects on Detection Performance

03/98 ASR		Story Weighted			Topic Weighted		
Deferral	α	P(Miss)	P(Fa)	C_{det}	P(Miss)	P(Fa)	C_{det}
10	.15	.9502	.0013	.0203	.3664	.0013	.0086
1 w/ retro	.15	.9546	.0012	.0203	.4268	.0012	.0097
1 w/o retro	.15	.9585	.0018	.0209	.5624	.0018	.0130

Table 4: Detection Development Results

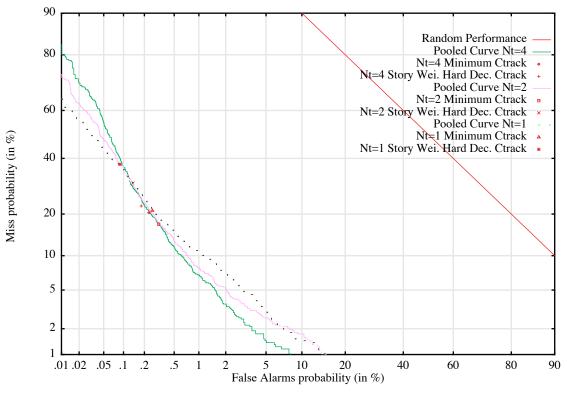| asr		Story Weighted			Topic Weighted		
Deferral	α	P(Miss)	P(Fa)	C_{det}	P(Miss)	P(Fa)	C_{det}
1	.10	.9742	.0017	.0212	.5255	.0017	.0123
	.15	.9612	.0009	.0201	.4181	.0008	.0092
	.20	.9678	.0005	.0198	.4974	.0005	.0104
	.25	.9770	.0004	.0199	.6450	.0004	.0133
10	.10	.9735	.0018	.0212	.4706	.0018	.0112
	.15	.9632	.0007	.0200	.3944	.0007	.0086
	.20	.9637	.0005	.0198	.4954	.0005	.0104
	.25	.9730	.0003	.0197	.5987	.0003	.0122
100	.10	.9740	.0018	.0212	.4478	.0017	.0106
	.15	.9590	.0007	.0199	.3528	.0007	.0078
	.20	.9681	.0003	.0197	.4065	.0003	.0084
	.25	.9733	.0002	.0197	.4751	.0002	.0097

Figure 2: FDCH Tracking Results, α = .25, β = .20

vant documents are used. For example, Figure 2 shows FDCH runs with 4, 2 and 1 training examples, α = .25 and β = .20. The highest P(Miss) score on the curve drops from 0.8 to 0.65, while the lowest P(Fa) score rises from 0.08 to 0.15 as one moves from 4 relevant documents to 1.

2.2. Future Directions

There are a number of ways in which we could improve upon these results. For instance we could refine the initial criteria by considering the magnitude of the difference between the new story's similarity with the closest relevant cluster and the closest non-relevant cluster. This is likely to reduce our false alarm rate. Other refinements that modify the training clusters with "high confidence" topic stories are also possible. These are likely to impact both the misses and false alarms.

3. DETECTION

This task takes the same general approach as that done for tracking. The stories in each input file are clustered using a specified membership threshold (α). We then 'pipeline' these cluster sets for the specified deferral period and then base decisions first on whether a given cluster is sufficiently close to a previously declared topic cluster based upon an inter-cluster threshold (currently also α). If it is we merge the new cluster with the declared topic cluster. Otherwise we look

forward in the pipeline to see if any future cluster is sufficiently close (same threshold α) as to warrant declaring the current cluster as a new topic cluster. Clusters failing both tests and containing a single story were discarded as noise for our official runs. Non-singleton clusters are declared as a new topic.

A deferral period of 1 is handled as a special case by retaining the 10 most recent file cluster sets for use in the second stage decision making process. This leads to a lag in the identification of new topics, but avoids the discarding of stories with a low appearance frequency. Story vectors and hence cluster vectors are generated after excluding stopwords and stemming the rest. Terms weights are computed using TF×IDF scores after normalizing for length of story. Term weights are incrementally updated every ten files.

3.1. Developmental Runs

Our initial approach to detection did not include the pipeline concept, focussing instead on tuning α over a range of 0.10 - 0.25 in increments of 0.05 for the specified deferral periods. As shown in Table 3 for one month of ASR data, providing a retrospective pipeline for a deferral period of 1 has significant effect upon performance.

Table 4 shows the scoring for our development runs. In all asr cases, α = 0.15 generates the minimal C_{det} value, which we subsequently used for our official runs, shown in Table 5.
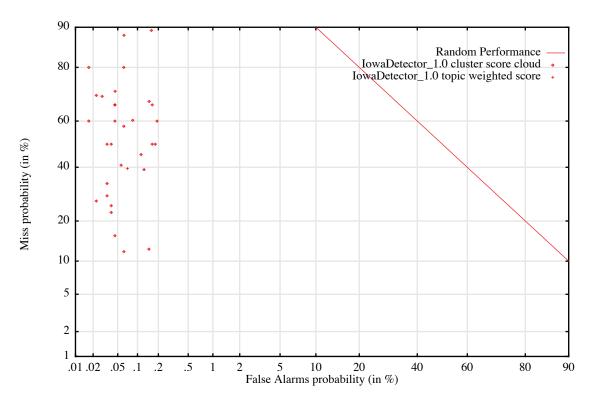
Figure 3: ASR Detection Development Results, deferral = 10, α = .15

Table 5: Official Detection Runs

Source	Deferral	Story Weighted			Topic Weighted		
		P(Miss)	P(Fa)	C_{det}	P(Miss)	P(Fa)	C_{det}
nwt+asr	1	.4957	.0012	.0111	.4207	.0012	.0096
	10	.6051	.0009	.0130	.4323	.0009	.0095
	100	.5540	.0013	.0123	.3593	.0013	.0084
nwt+man_ccap	1	.5621	.0009	.0121	.4421	.0009	.0097
	10	.4138	.0010	.0093	.3776	.0010	.0085

3.2. Analysis of results

Comparing cluster score clouds across runs, we have found that raising the threshold does improve our false alarm rate proportionately. Unfortunately, we are not seeing a corresponding improvement in P(Miss). Rather than the entire cloud shifting down, we are finding that the cloud is instead elongating, with some topics improving well and others hardly at all. Figure 3 illustrates this effect for α = 0.15. We suspect that this is due in a great extent to the fact that we are generating roughly 2000 - 3000 declared topic clusters during a given run, and that the documents relevant to a given topic are splitting across two or more clusters. The evaluation scheme then chooses only one of these for scoring. We are currently examining intercluster similarities to see if some form of cluster fusion could reduce the number of declared clusters.

4. SEGMENTATION

Text segmentation was performed with an agglomerative clustering approach. Clusters were built iteratively from the word level up, combining neighboring clusters as long as sufficiently similar neighboring clusters appeared in the deferral window. The result is a very fast and flexible algorithm that will be extended in several natural ways to increase its accuracy. We note that in this section, the word "cluster" refers to a logical construct consisting of a block of consecutive words and pauses.
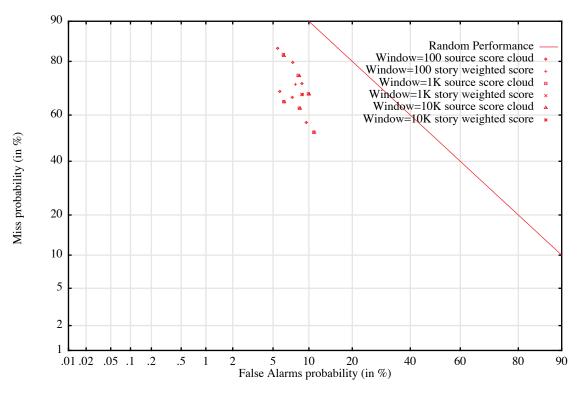
Figure 4: Segmentation Results

The algorithm begins as follows. Source text is read until the deferral window is filled. Initially each sentence (note: for ASR text, "sentence" refers to a group of words between pauses) is considered to be a cluster. A similarity score (described below) is then computed for all pairs of neighboring clusters. If the most similar pair of neighbors meets a minimum similarity threshold, the two clusters are combined to form a new cluster, which is then compared to its neighbors. The process repeats until no pair of neighbors meets the similarity threshold, or until all the sentences in the window have been combined into one cluster.

A general step of the algorithm proceeds similarly. The left end of the deferral window is placed at the first inter-cluster gap; this is the earliest potential segment gap. The window is again filled with new words, and the clustering algorithm is performed until no further combinations are possible. If the leftmost cluster in the deferral window was combined with the cluster on its left, that means that the potential gap was in fact just part of a larger segment. If not, then the potential gap was in fact a segment break, and the new segment is declared. This step is then repeated until the end of the file is reached.

We use a max-heap [1, 5] to access the cluster similarity scores. This means that the most similar cluster pair can always be found in logarithmic time. The algorithmic complexity of the clustering method is therefore O(k log k), where k is the number of sentences in the file. In practice we

found that the clustering runs could be performed in under an hour on a top-end Linux PC.

A simple combination criterion of similarity was used for the TDT2 runs, depending on the duration of the inter-cluster pause and a lexical similarity score. If the pause duration was smaller than a given time threshold (0.5 seconds in the test runs) the similarity score was set to the maximum, insuring that the clusters would be combined. If the duration was greater than a second threshold (4 seconds), the score was set to the minimum, ensuring that a gap would be declared. These two rules were applied irrespective of lexical similarity. Otherwise, the pause duration was considered to be of no value, and similarity was computed as a dot product of TF×IDF weighted cluster representation vectors. Stemming and stop words were not employed. Thus, only three operational parameters controlled the performance of the system. Figure 4 shows the combined results for all window sizes. Our false alarm rate is reasonable compared to other systems, but we have a rather high miss rate.

With the framework in place, we now turn our attention to fine-tuning the algorithm with some natural extensions, of which we mention three. First, the values of the three operational parameters will be optimized. We are currently implementing a stepwise gradient descent method for learning the optimal values within the current combination objective. Second, the criterion will be extended to incorporate more information, such as the number of words in a cluster and the presence of stop words. Finally, the algorithm itself can be

made more general by examining more than neighboring pairs of clusters. For instance, it is possible that a cluster could match its neighbor poorly, but match its neighbor's neighbor very well, indicating that all 3 clusters belong to the same story. We will experiment with varying the width of this search.

REFERENCES

1. Crane, C.A., "Linear lists and priority queues as balanced binary trees," Tech. Rep. STAN-CS-72-259, Stanford University, Stanford, CA, 1972.

2. Eichmann, D., M. E. Ruiz and P. Srinivasan, "Cluster-Based Filtering for Adaptive and Batch Tasks," *Seventh Conference on Text Retrieval,* NIST, Washington, D.C., November 11 - 13, 1998.

3. Gamma, E., R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison Wesley, 1995.

4. Porter, M. F., "An Algorithm for Suffix Stripping," *Program,* v. 14, no. 3, 1980, p. 130-137.

5. Weiss, M.A., *Data Structures and Algorithm Analysis (Second Edition)*, Addison-Wesley, 1997.