CMU Report on TDT-2: Segmentation, Detection and Tracking

Jaime Carbonell, Yiming Yang, John Lafferty,
Ralf D. Brown, Tom Pierce, and Xin Liu

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
{jgc,yiming,lafferty,ralf,tomp,xliu}@cs.cmu.edu

ABSTRACT

This paper reports the results achieved by Carnegie Mellon University on the Topic Detection and Tracking Project's second-year evaluation for the segmentation, detection, and tracking tasks. Additional post-evaluation improvements are also presented.

1. INTRODUCTION

Carnegie Mellon University (CMU) participated on all three "official" TDT-2 evaluations: 1) automated segmentation of ASR text, 2) detection with a 1-file deferral period on segmented ASR text, and 3) tracking on segmented ASR text with 4 training stories per topic (N_t=4). CMU also reported results for both detection and tracking on unsegmented text by combining automated segmentation with detection and with one of the tracking methods (d-trees). CMU's results for segmentation were the best reported in the official evaluation, and average for detection and tracking. We report on each task and method briefly in this paper, as well as on post-evaluation results.

2. AUTOMATED SEGMENTATION

The story segmentation system that was developed for the TDT-2 evaluation was built closely along the lines of the framework presented by Beeferman *et al* [1]. As described in this paper, the basic approach is simple: we train a statistical model that uses both "topicality" features to detect shifts in the subject of the story, and "cue-word" features that allow us to model the style of language that is characteristic of the beginning and end of a story. The "topicality" features are constructed using an adaptive trigger-based language model trained according to the minimum divergence criterion with respect to a smoothed trigram default model [2]. This model was trained on 10 million words of CNN data from the years 1992--1994, and is identical to the one used in CMU's TDT-1 system. The "cue-word" features query the presence of individual words in the neighborhood of a candidate boundary. The statistical modeling paradigm incrementally adds features to a log-linear model, using a maximum entropy framework that is similar to stage-wise versions of additive logistic regression [3].

The candidate set for feature selection made use of all of the fields in the TDT-2 ASR-text files except for the confidence score. Thus, the features incorporated information about the individual words, the speaker cluster id, silence duration, and the document source. Story boundaries were only hypothesized where there was a silence annotation in the ASR-text files. The models were trained on all of the January-April development set (train and devtest), except for approximately 100,000 words of text that were held aside for testing and threshold setting.

Judging from the features that were induced, the speaker cluster feature appears not to have been informative. One significant new aspect of the segmentation system was that individual models were trained for the different news sources: one for each of the ABC, CNN, PRI, and VOA sources. While VOA consists of two subsources in the development data (and three in the evaluation data), a single VOA model was trained. The use of source-dependent models led to a 1-2% absolute improvement in C_{seg} on the training set. A less significant change from the TDT-1 system was that the cue-word vocabulary was reduced, in order to decrease the number of candidate features and thus speed up the process of feature selection. A preliminary error analysis suggests that this may have hurt performance somewhat, since several important words, particularly for the VOA sources, were missing. Approximately 200 features were induced for each of the sources; cross-validation was carried out on the smaller sources (ABC, PRI, VOA) to determine the best stopping point, but no smoothing was used.

There were a few things that had been useful for TDT-1 data and other manually transcribed texts that did not prove to be effective on the TDT-2 ASR-text data. For example, in previous work a significant reduction in miss rate resulted from growing a decision tree on the same data, and then linearly interpolating the tree and the exponential model. However, the performance of decision trees was significantly lower on the TDT-2 data, and so they were not used. We also experimented with boosted decision trees, but these too performed significantly worse than the exponential models. In particular, we used the Real AdaBoost algorithm [4] applied to 3-level and 1-level decision trees as "weak learners," but these performed just slightly better than a single tree, with a roughly 3% higher absolute C_{seg} than the maximum entropy method. Further work needs to be done to better understand these negative results.

In spite of the fact that the additional error reduction methods proved ineffective, our results using the standard TDT error metric for segmentation was the lowest, 0.1463 in the official evaluation on the previously unseen stream of test stories.

3. DETECTION

The CMU topic detection detection system for TDT-2 was largely based on previous retrospective and online detection systems used in the TDT-1 pilot study [5, 6], since the new definition of the detection task encompasses the the earlier versions of the tasks. In particular we used the vector space model (VSM) to represent documents as weighted unigram models. We also used temporally-sensitive versions of our incremental and hierarchical group-average (GAC) clustering algorithms to detect new topics within the 3 deferral periods.

The operation of the system is fairly straightforward. First, the inverse-document frequencies for all terms are initialized using a training corpus. As new stories "arrive" in the deferral window, they are used to update the IDF statistics. Then, the entire deferral window is clustered using GAC with fractionation [7], an efficient greedy agglomerative clustering algorithm. This method divides the incoming data stream into sequential buckets, clustering by cosine similarity within each bucket, and then merging clusters among buckets. Additional shuffling and reclustering among stories in adjacent buckets improves performance. A fortunate property of fractionation lies in the way it achieves it's efficiency- by dividing the data, and performing a partial clustering, and repeating; by using the time of stories as a criteria for these divisions we can prefer associating stories that occur nearby in time. Preferring temporally proximate stories in this way provided a substantial performance increase in earlier TDT work.

The TDT task requires the identification of new topics prior to the expiration of the deferral window. In other words, decisions as to whether individual stories belong to existing topics or create the start of a new one must be made without waiting for new data past the deferral window. These decisions are made by cluster-to-cluster similarity comparisons between each cluster in the deferral window and past topic clusters prior to the deferral period. Specifically:

$$\mathrm{sim}(C_1, C_2) = \cos(\mathrm{centroid}(C_1), \mathrm{centroid}(C_2))$$

$$\mathrm{NEW}(C_1) \text{ iff } \max_i[\mathrm{sim}(C_1, C_i)] < \mathrm{THRESHOLD}$$

If the maximum similarity is above a threshold (optimized on the training corpus), then the cluster is deemed to be an extension of the previous topic. Otherwise, it is deemed to be the onset of a new topic. In either case (the extended old topic or the beginning new topic) becomes part of the past as the deferral window is moved forward in time.

As an improvement to the similarity function we implemented a form of temporal decay for the cluster-to-cluster similarity function. At first, we implemented the date of the earliest story as a timestamp, but this led to a high miss rate (0.3526 story weighted, to be specific). Changing the time stamp to the most recent story (to penalize less topics that continue to be reported) significantly improved our results, dropping our miss rate to 0.0765 while increasing false alarm by a much smaller proportion (0.0013 vs 0.0004).

We found that the use of incrementally-updated IDF statistics allowed for larger events to be followed through their evolution when later reports contain evolving content; essentially allowing topic drift. This becomes important in larger events, when knowledge about the event, the terminology used to describe it, and related developments can change substantially from initial reports as the event matures.

Our methods appear to degrade gracefully under imperfect transcription and boundary conditions. Machine recognized speech tended to increase the number of miss and false-alarm documents minimally, although word-error rates (WER) approached 25%. Initial tests with automatically determined story boundaries incurred more degradation, but still less than anticipated. Further tests are required to quantify these effects more reliably, but the results reported in Tables 1 and 2 below provide good evidence of robust performance.

Source Condition	Segmented?	Deferral Period	Story Weighted	Topic Weighted
ASR	yes	10 files	0.0075	0.0057
ASR	yes	1 file	0.0092	0.0060
Transcript	yes	10 files	0.0068	0.0049
Transcript	yes	1 file	0.0089	0.0057
ASR	NO	10 files	0.0107	0.0076

Table 1: Detection Results

Source Condition	Segmented?	Deferral Period	Story Weighted	Topic Weighted
ASR	yes	10 files	0.0028	0.0042

Table 2: Post-Submission Improved Detection Results

We expect that performance would be improved by treating incoming stories differently based on source, as will be discussed in Section 6. Also, editorial style articles, regardless of source, often might diverge enough to be considered distinct from a group of ‘‘just-the-facts’’ reporting on the same topic. Both of these behaviors have been observed, and hopefully can be addressed in future revisions.

4. TRACKING

We used two methods for topic tracking, k-nearest neighbors (kNN) and decision-tree induction. kNN is an instance-based classification method which has been previously applied (by Yang et al [8]) to text categorization, with generally good performance.

A requirement in official TDT evaluations is that each event be tracked independently, without any knowledge about other events. According to this constraint, we adapted our conventional M-ary classification kNN developed for text categorization in general, and trained a kNN classier per topic. Essentially, the system converts an input story into a vector, compares it to the training stories, and selects the k nearest neighbors based on the cosine similarity between the input story and the training stories. The confidence score for a YES prediction on the input story is computed by summing the similarity scores for the positive and negative stories respectively in the k-neighborhood, and checking the difference between the two sums is above a threshold (typically 0, unless we want to introduce a bias towards avoiding false negatives or avoiding false positives). The scores for all test events are also recorded to computer Decision-Error Tradeoff (DET) curves.

Since positive instances are sparse (for most topics) in the training set, it is difficult to achieve greater recall (fewer misses) without significantly sacrificing precision (more false alarms). One solution to this problem is to discount the influence of negative examples by sampling a small portion in the k-neighborhood, and ignoring the remaining examples. This idea leads to a modified version of kNN; for distinction, we refer to the original version as kNN-a, and the modified version as kNN-b. In the modified version, we take k1 (less than k) nearest positive examples (P(x,k1)) and k2 (less than k) nearest negative examples (N(x,k2)) from the k-neighborhood, and average the similarity scores of the two subsets respectively. The confidence score for a YES prediction on the input story is computed as an interpolated value between these two quantities. For the purpose of the official evaluation, however, we did not use different k1 and k2 parameters.

The Decision Tree (DTree) tracking method also uses binary classifiers. DTree uses not only the words as features, but also a number of meta-features such as presence of a word near the beginning of a story, multiple occurrences of a word, highly-collocated bigrams, and M-of-N thresholds. Further, the decision-tree induction can be tuned through a variety of inductive-bias parameters such as the maximum number of negative training instances, when to stop growing the tree, and which meta-features to consider. A time-windowing capability is available when performing the actual tracking using the induced decision trees, but it was ineffective in this evaluation because the data covered only two months of time -- in TDT-1 we found that two months was an appropriate size for the temporal window.

To create an M-of-N threshold meta-feature, DTree simply selects the N top-ranked features (including non-threshold meta-features) and then splits the collection of stories for the decision node by whether or not they contain at least M of those N features, rather than whether or not they contain the single top-ranked feature.

Bigram meta-features are selected based on the mutual conditional probability of the two words. From each possible pair of adjacent words, select only those where the occurrence probability of *each* word given the other is above some pre-selected threshold. This list of candidate bigrams is then further limited to a specified number of bigram features by selecting the most likely from among the candidates.

Because of the small number of positive training instances, the information gain which is the primary means of ordering features in decision-tree induction had to be augmented with additional measures that serve as tie-breakers. If two (meta-)features have the same information gain, then

- bigrams will be given priority over single words,
- words occurring near the beginning of the story will be given priority over words occurring anywhere,
- simple words are given priority over multiple-occurrence meta-features, and
- the feature with higher TF*IDF score is given priority.

Should none of the above tie-breakers prefer one word over another, the final resort is to prefer the longest words and then to simply alphabetically sort words of equal length.

The DTree program has been optimized for speed when tracking multiple events (especially when testing multiple values of N_t) on the same story collection. The entire collection is read into memory and indexed before tracking begins, trading off increased memory use (depending on parameters, 3 to 4.5 times the size of the text without SGML markup) against reduced run-time. The evaluation set was processed at N_t=4 for all 21 evaluation events in just 4 minutes 31 seconds, using 140 megabytes of memory, and for N_t=1,2,3,4 in 8 minutes 25 seconds.

System	Source Condition	Boundaries Given?	Story Weighted	Topic Weighted
kNN	ASR	yes	0.0076	0.0077
kNN	Cl.Caption	yes	0.0072	0.0072
DTree	ASR	yes	0.0096	0.0085
DTree	Cl.Caption	yes	0.0100	0.0090
DTree	ASR	NO	0.0116	0.0109

Table 3: Official Tracking Results, N_t=4

System	Source Condition	Boundaries Given?	Story Weighted	Topic Weighted
DTree	ASR	yes	0.0085	0.0079
DTree	Cl.Caption	yes	0.0089	0.0080
kNN	ASR	NO	0.0105	0.0092
DTree	ASR	NO	0.0106	0.0105

Table 4: Post-Submission Tracking Results, N_t=4

For the default evaluation, DTree used 300 negative training stories, 4-of-12 threshold features, the multiple-occurrence and near-beginning (within 125 words) meta-features, and stopped growing the trees whenever a node had at least 56% positive instances. The parameters for the closed-captioning text and FDCH manual transcriptions were similar, using 200 negative training stories in each case, and stopping at 30% (32%) positive instances for a node. All of these parameters were set via cross-validation on the development set.

We found that on the TDT-2 collection, the following were advantageous, improving C_{Track} for DTree:

- preferring words near the beginning of the story
- threshold meta-features (greatly improve robustness)

The following did not improve performance:

- adaptive time-windowing (insufficient temporal extent in the collection)
- bigram features (increased miss rate more than it reduced false alarms)
- sets of words common to the positive training stories (often increase **both** miss and false alarm rates)

And the following varied, sometimes helping and sometimes hurting performance:

- stemming the words of the story
- distinguishing between single and multiple occurrences of a word

Planned enhancements to DTree are 1) per-source decision-tree training and induction parameters and 2) unsupervised updates triggered by a second, high-precision decision tree. Currently, all parameters such as M/N for threshold features and the cutoff for "near the beginning of the story" are set globally, but DTree would benefit from having these parameters set individually for each source. This will be discussed in more detail in Section 6.

The tracking results for kNN and DTree were as shown in Table 3. These results indicate that kNN is somewhat superior to DTrees for this tracking task, and also that ASR with 25% word-error degrades performance only moderately (about 10% as in detection). Moreover, combining automated segmentation with tracking also degrades performance gracefully, by about the same percentage as for detection. Most of the increase in C_{Track} is due to increased false alarms (roughly double), many of which result when an incorrect segmentation places key words into one of the stories adjacent to the actual on-topic story, causing DTree to decide that the adjacent story is also on-topic. Since kNN is less affected by individual words, it suffers less degradation in the topic-weighted error metric from the segmentation errors, as shown in Table 4 for a post-evaluation run.

After submitting our results for tracking with DTree, we discovered that an error in the code used to find optimum parameter settings resulted in suboptimal parameters. On correcting the error and re-tuning for the training set, we achieved the improved results shown in Table 4 using 225 negative stories, 4-of-12 threshold features, the near-beginning meta-feature, and stopping expansion at 50% positive instances.

5. CONCLUSIONS

Carnegie Mellon's automated segmentation system had the best performance metric at 0.1463, and in post-submission collaboration with IBM researchers was shown to perform even better when using an IBM-developed post-processor. At this level of accuracy, the automatic segmentation allows detection and tracking to be performed with a degradation of about 20% to 30% in the error metric, primarily as a result of increased false alarms. The increase in false alarms appears to be due to segmentation errors causing key content words to be assigned to neighboring stories.

While our submitted detection results were substantially poorer than the best system's results, primarily due to a very high miss rate, a minor post-submission change to the algorithm allowed our detection system to nearly match the best reported topic-weighted results with an error metric of 0.0042 and perform even better for story weighting with an error metric of 0.0028 (in both cases, the error metric resulting from *never* detecting a new topic is 0.0200).

For tracking, both of our systems had similar performance on ASR data, with kNN yielding an error metric of 0.0077 and DTree yielding 0.0079 (as for detection, the error metric from *never* declaring a story to be on-topic is 0.0200). Surprisingly, while kNN performs about 10% better on transcribed data, DTree actually yields a marginally higher error metric.

6. FUTURE WORK

As mentioned previously, we intend to add the source of a story as another item contributing to the decisions for both detection and tracking. The different sources of TDT-2 stories often differ dramatically in character -- length, detail, vocabulary, style, etc. -- and thus a single globally-tuned set of parameters is unlikely to be optimal for **any** of the sources. As a particular example, DTree performed best when preferring the first 125 words of a story, but that is close to the *average* length of a CNN story, meaning that this preference has little effect when the source of a story is CNN. For CNN, 60 words may be more a appropriate setting for the "near beginning of story" meta-feature.

The small number of positive training instances poses a significant problem for per-source training in the tracking task. We intend to overcome this problem by ignoring the source of the positive training stories, in effect duplicating them across each news source. While this clearly leads to degraded training data for any given source, we expect the improvement from training individually on each source to outweigh the degradation caused by not having positive training instances from some of the sources.

An area of concern in tracking which we have not yet addressed is topic drift. Our current tracking systems simply train an event model once and for all from the training stories and then continue to use the same model, even though the event evolves over time. We intend to add some form of adaptation to the tracker in order to better model the event. Using a second copy of the tracker tuned for high precision will allow those stories in which the system has the greatest confidence to be added to the training set (without supervision), letting the tracker train on additional instances as additional news stories are received.

7. REFERENCES

1. D. Beeferman, A. Berger, and J. Lafferty, ''Statistical Models for Text Segmentation'', *Machine Learning*, Vol. 34, 1999, pp. 1-34.

2. D. Beeferman, A. Berger, and J. Lafferty, ''A Model of Lexical Attraction and Repulsion'', *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics.*, 1997.

3. T. Hastie and R. Tibshirani, *Generalized additive models,* Chapman and Hall, 1990.

4. R. Schapire and Y. Singer, ''Improved Boosting Algorithms using Confidence-Rated Predictions'', *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998.

5. James Allan and Jaime Carbonell and George Doddington and Jonathan Yamron and Yiming Yang, ''{Topic Detection and Tracking Pilot Study: Final Report}'', *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

6. Yiming Yang, Thomas Pierce, and Jaime Carbonell, ''A study on retrospective and on-line event detection'', *Proceedings of the 21th Annual Int ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998, (SIGIR'98)

7. D.R. Cutting, D.R. Karger, J.O. Pedersen, and J.W. Tukey, ''Scatter/Gather: a Cluster-basd Approach to Browsing Large Document Collections'', *Proceedings of the 15th Annual Intl. ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992, (SIGIR'92)

8. Yiming Yang, ''An evaluation of statistical approaches to text categorization'', *Journal of Information Retrieval*, 1999, (to appear)