A HIDDEN MARKOV MODEL APPROACH TO TEXT SEGMENTATION AND EVENT TRACKING

J.P. Yamron, I. Carp, L. Gillick, S. Lowe, and P. van Mulbregt

Dragon Systems, Inc.
320 Nevada Street
Newton, MA 02160

ABSTRACT

Continuing progress in the automatic transcription of broadcast speech via speech recognition has raised the possibility of applying information retrieval techniques to the resulting (errorful) text. For these techniques to be easily applicable, it is highly desirable that the transcripts be segmented into stories. This paper introduces a general methodology based on HMMs and on classical language modeling techniques for automatically inferring story boundaries and for retrieving stories relating to a specific event. In this preliminary work, we report some highly promising results on accurate text. Future work will apply these techniques to errorful transcripts.

1. INTRODUCTION

Over the last few years Dragon, like a number of other research sites, has been developing a speech recognition system capable of automatically transcribing broadcast speech [7]. With the recent advances in this technology, a new source is becoming available for information mining, in the form of a continuous stream of errorful, unsegmented text. Applying standard information processing techniques to this data, such as filtering or routing, requires that this text first be segmented into topically homogeneous blocks [1, 2, 4, 5, 6]. Unlike newswire, typical automatically transcribed audio data contains no information (other than pauses in the audio) about how the stream should be divided up.

Our approach to the problem of segmentation is to treat a story as an instance of some underlying topic, and to model an unbroken text stream as an unlabeled sequence of these topics. In this model, finding story boundaries is equivalent to finding topic transitions. At a certain level of abstraction, identifying topics in a text stream is similar to recognizing speech in an acoustic stream. Each topic block in a text stream is analogous to a phoneme in speech recognition, and each word or sentence (depending on the granularity of the segmentation) is analogous to an "acoustic frame." Identifying the sequence of topics in an unbroken transcript therefore corresponds to recognizing phonemes in a continuous speech stream. Just as in speech recognition, this situation is subject to analysis using classic hidden Markov modeling (HMM) techniques, in which the hidden states are topics and the observations are words or sentences.

Given a segmentation of the text stream, it becomes possible to apply other kinds of information processing. Here we will consider the problem of *event tracking*, a variation of the filtering task in information retrieval, in which the system is given a few examples of stories on a particular event of interest and is asked to automatically find other examples in the text stream. While, in the longer term, it is our intention to address the twin problems of text segmentation and event tracking using errorful transcripts generated by automatic speech recognition, in the present study we focus our attention on accurate texts only.

The experiments described here on segmentation and event tracking were carried out using the Topic Detection and Tracking (TDT) Pilot Study Corpus, and evaluated following the procedures set out in the TDT Evaluation Plan. Both are available through the Linguistic Data Consortium (LDC) at the University of Pennsylvania.

2. THE SEGMENTER

Suppose that there are k topics $T^{(1)}$, $T^{(2)}$, ..., $T^{(k)}$. There is a language model associated with each topic $T^{(i)}$, $1 \leq i \leq k$, using which one can calculate the probability of any sequence of words. In addition, there are transition probabilities among the topics, including a probability for each topic to transition to itself (the "self-loop" probability), which implicitly specifies an expected duration for that topic. Given a text stream, a probability can be attached to any particular hypothesis about the sequence and segmentation of topics in the following way:

1. Transition from the start state to the first topic and accumulate a transition probability.
2. Stay in topic for a certain number of words or sentences, and, given the current topic, accumulate a self-loop probability and a language model probability for each.
3. Transition to a new topic, accumulate the transition probability, and go back to step 2.

A search for the best hypothesis and corresponding segmentation can be done using standard HMM techniques and standard speech recognition tricks (using thresholding if the search space gets too large, for example).

Note that this algorithm does segmentation and topic assignment simultaneously. It may be the case, however, that the topics needed by the algorithm to optimize the segmentation are not appropriate or useful as identifiers for the segments. If a topic assignment is required in this situation, a separate identification step using a different set of topics must be done.

2.1. Constructing the Topic Models

The topic language models used by the segmenter were built from transcriptions of broadcast news from the period January 1992

through June 1994, part of a broadcast collection available through the LDC. This data was filtered to remove shows not included in the test corpus (see below), and to remove stories of fewer than 100 or more than 2,000 words. This left 15,873 stories of average length 530 words. A global unigram model consisting of 60,000 words was built from this data.

Topic clusters were constructed by automatically clustering the stories in the training data. This clustering was done using a multi-pass k-means algorithm that operates as follows:

- At any given point there are k clusters. For each story, determine its distance to the closest cluster (based on the measure described below), and if this distance is below a threshold, insert the story into the cluster and update the statistics. If this distance is above the threshold, create a new cluster.
- Loop through the stories again, but now consider switching each story from its present topic to the others, based on the same measure as before. Some clusters may vanish; additional clusters may need to be created. Repeat this step as often as desired.

The distance measure used in the clustering is based on the Kullback-Leibler metric:

$$d = \sum_n (s_n/S) \log \frac{s_n/S}{(c_n + s_n)/(C + S)} + (c_n/C) \log \frac{c_n/C}{(c_n + s_n)/(C + S)} ,$$

where s_n and c_n are the story and cluster counts for word w_n, with $S = \sum_n s_n$ and $C = \sum_n c_n$. The first term in this measure can be interpreted as the distance of the story from the updated cluster, while the second term is the distance the cluster itself moves as a result of adding the story.

In order to prevent very common words and punctuation symbols from dominating the computation, we introduced a stop list containing 174 entries. These words did not participate in the computation of the distance measure.

A topic language model was built from each cluster. To simplify this task, we limited the number of clusters to 100 and chose to model each topic using unigram statistics only. These unigram models were smoothed versions of the raw unigram models generated from the clusters. Smoothing each model consisted of performing absolute discounting followed by backoff [3] to the global unigram model; in other words, a small fixed count (about .5) was stolen from the non-zero raw frequencies, and the liberated counts were redistributed to the rest of the words in the model in proportion to the global unigram distribution built from the training data. The raw cluster unigrams were quite sparse, typically containing occurrences of only 6,000 distinct words from the training list of 60,000 words. Words on the stop list were removed from the models.

We will frequently refer to these topic language models as *background topics* or *background models.*

2.2. Segmentation Results

Experiments were performed on the TDT Corpus, a collection of 15,863 transcribed CNN news stories and Reuters newswire, ordered by date, and covering the period July 1994 through June 1995. The corpus was prepared by removing all story and paragraph boundaries. Decoding of text was done by using a speech recognizer with 100 underlying "single node" models (corresponding to the topics), each of which was represented by a unigram model as described above. The text was scored against these models one *frame* at a time—a frame corresponding, in these experiments, to a sentence. The topic-topic transition penalties were folded into a single number, the topic-switch penalty, which was imposed whenever the topic changed between sentences.

The topic-switch penalty was tuned to produce the correct average number of words per segment on the first 100 stories from the test set. There are no other parameters to tune except the search beam width, which was set large enough to avoid search errors in our experiments.

The segmenter was first run on the TDT Corpus. The corpus was then split into its CNN and Reuters components and the segmenter was run on these separately. This latter experiment was included to test the robustness of the system to a mismatch between the training material for the background models, which was all from CNN transcripts, and the test material.

2.2.1. TDT Corpus

On the TDT Corpus the segmenter hypothesized 16,139 segment boundaries, compared to the 15,863 story boundaries in the test set. Of these, 10,625 were exact matches, yielding a recall rate (percentage of true boundaries found) of 67.0% and a precision (percentage of hypothesized boundaries which are true) of 65.8%. Many other hypothesized boundaries were shifted from the correct break by only a few words; if a hypothesized boundary within 50 words (1/10 of the average story length) of a true boundary is considered a "match," for example, the recall rate rises to 81.9% and the precision to 80.5%.

Another measure of segmentation quality is the error metric proposed in the TDT Evaluation Plan (a modification of a metric proposed in [1]), which is the probability that two words separated by distance k in the corpus are classified (as belonging to the same story or to different stories) by the segmenter in the same way as in the true corpus, where k is taken to be half the average document length in the corpus. Using this measure, the output of our segmenter misclassifies words 12.9% of the time on the TDT Corpus.

We did an alignment of the hypothesized and reference story boundaries to explore the kinds of the errors made by the segmenter. Some of the problems that were found included:

- Failure to distinguish a boundary between successive stories because they were assigned to the same background topic. This didn't seem to be a large source of errors, but the effect can be reduced by increasing the number of background models.
- Failure to accurately position boundaries relative to "broadcast filler", such as, "More news after this." This is a weakness of a system that does not model story structure.
- Splitting of stories at internal topic shifts. This "problem" actually goes to the heart of what it is we are trying to accomplish, and it is not clear that this is always undesirable behavior.
- Oscillation of topic in stories not well-modeled by the background topics. This might be solved by using models with better discrimination, such as bigram models, or models that adaptively train so as to stay current.

2.2.2. CNN vs. Reuters

The segmenter hypothesized 8,706 boundaries on the CNN portion of the TDT Corpus, which consisted of 7,898 stories. 4,596 were exact matches, for a recall rate of 58.2% and a precision of 52.8%. Note that this is *worse* than for the TDT Corpus as a whole, despite the fact that the training data was well-matched to this subcorpus. The figures for the Reuters data are more surprising: 8,487 hypothesized boundaries for 7,965 stories, of which 6,138 were exact matches, giving a recall rate of 77.1% and a precision of 72.3%. This is *better* than for the TDT Corpus as a whole, despite the mismatch between training and test.

This result is also reflected in the computation of the segmentation error metric, which yields an error rate of 16.8% on CNN and 12.3% on Reuters.

The likely explanation of this result is that the CNN is simply more difficult than Reuters for a content-based segmenter such as ours. For example, written news tends to be more concise than broadcast news, with none of the typical "fillers", such as introductions, greetings, and sign-offs. It is also the case that the length of CNN stories varies much more widely than Reuters stories, a problem for our segmenter, which has a single parameter controlling for length.

3. THE TRACKER

The event tracker is an adaptation of the segmenter. As discussed above, the segmentation algorithm does segmentation and topic assignment simultaneously. In general, the topic labels assigned by the segmenter (which are drawn from the set of automatically derived background topics) are not useful for classification, as they are few in number and do not necessarily correspond to categories a person would find interesting. However, by supplementing the background topic models with a language model for a specific event of interest, and allowing the segmenter to score segments against this model, it becomes possible for the segmenter to output a notification of an occurrence of that event in the news stream whenever it assigns that event model's label to a story. In this implementation, the topic models have the role of determining the background against which the event model must score sufficiently well to be identified.

In this incarnation, the segmenter is not asked to identify story boundaries. Its job is merely to score each story against its set of background models, as well as against the event model, and report the score difference between the best background model and the event model. A threshold is applied to this difference to determine whether a story is about the event or not, and this threshold can be adjusted to tune the tradeoff between missing and falsely reporting stories on the event.

Because an "event" as defined in the TDT Evaluation Plan is expected to be something localized in time (such as a bombing or an earthquake), we included a *duration penalty* in the score of the event model, making a story less likely to be on an event as time goes on. The penalty was taken to be a global constant times the number of stories in the corpus separating the current story from the event's last training example.

3.1. Constructing the Event Models

An event model is built from N_t training stories, where N_t may have the values 1, 2, 4, 8, or 16. As language models, these are extremely sparse and must be smoothed, which we do in a manner similar to the background models:

1. Apply our stop list of 174 common words and punctuation, so that they don't participate in the topic determination.
2. Steal a small amount (about .5 count) from the non-zero raw frequencies using absolute discounting.
3. Redistribute the liberated counts to the rest of the words in the event model in proportion to their occurrence in a backoff unigram distribution.

In this case, in order to provide a more accurate smoothing for the event model, the backoff unigram distribution is not the one generated from the segmentation training data. Instead, we take as the backoff distribution the mixture of the background topic models that best approximates the unsmoothed event model. There is therefore a different backoff model for every event and every value of N_t.

To construct these backoff models we proceed as follows. If we refer to the probability of word w_n in the unsmoothed event model as e_n, and its probability in the topic model $T^{(i)}$ as $T_n^{(i)}$ (there are 100 topic models, so $1 \leq i \leq 100$), then its probability in the backoff distribution $p_n^{(e)}$ is defined to be:

$$p_n^{(e)} = \sum_i \lambda^{(i)} T_n^{(i)} \ .$$

The $\lambda^{(i)}$ are *mixture coefficients*, which sum to one. The best choice for the $\lambda^{(i)}$ is the one that makes the distribution $p^{(e)}$ "closest" to the unsmoothed distribution e, which we define by maximizing the probability that $p^{(e)}$ generates the training stories that make up e. This maximization leads via the EM algorithm to an iterative solution for the $\lambda^{(i)}$:

$$\lambda^{(i)} = \sum_n e_n \frac{\lambda^{(i)} T_n^{(i)}}{\sum_j \lambda^{(j)} T_n^{(j)}} \ .$$

For each event and N_t value, therefore, we compute a distribution $p^{(e)}$ by solving the above equation, plug $p^{(e)}$ into step 3 above as the backoff model, and generate a smoothed event model for use in the tracker.

3.2. Tracking Results

About 8% of the stories in the TDT Corpus are labeled from a set of 25 events that occurred in the July 1994 through June 1995 timeframe (examples include Bobby Wayne Hall's helicopter crash and the Oklahoma City bombing). The number of stories assigned to a particular event varied from 2 to 273, with an average of about 50 per event. These were the events used in the tracking experiments.

The tracking results comprise a large number of experiments, each performed as follows:

1. Choose an event E and a number of training examples N_t (N_t is taken to be either 1, 2, 4, 8, or 16).
2. Find the N_t^{th} story labeled as event E in the corpus.
3. Train a language model for event E (as described above) from all stories with label E prior to and including the story identified in step 2.
4. Run the tracker on all stories after the 16^{th} story labeled with event E. (Testing on all stories after the 16^{th} labeled story makes the test set the same for all values of N_t.)

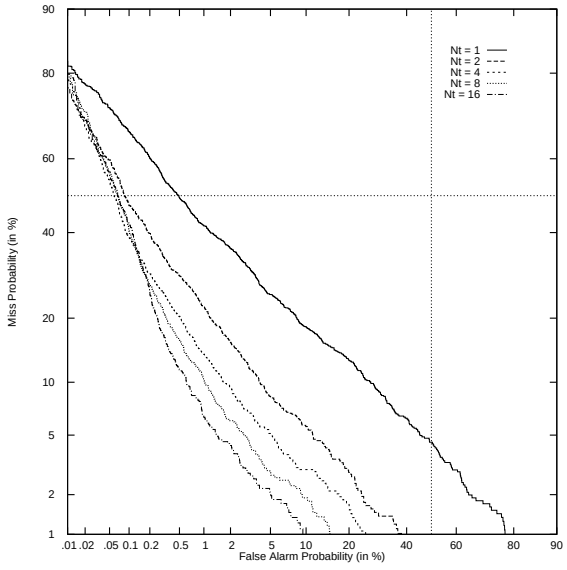

Figure 1: Tracking performance for different values of N_t, with duration penalty.

This procedure was repeated for all possible values of E and N_t.

The only tunable parameters in this system are the duration constant and the threshold on the event-background score difference. As we are still exploring the limits of this technology, we did not try to set either *a priori;* instead, the duration constant was tuned on the test set to optimize performance, and the threshold was varied to produce full plots of miss *vs.* false alarm.

Figure 1 shows the results of our tracking experiments for each value of N_t, averaged over all events. Our tracker is clearly very sensitive to the number of training examples, performing poorly for $N_t = 1$ but quite well for $N_t = 16$. This indicates that the smoothing methods we are using are not effective on extremely sparse data (one story) but do improve quickly as the amount of data increases.

The system gives its optimum performance at very low miss rates, indicating that as the score threshold drops, the language models are doing a good job of distinguishing the target stories from the rest and keeping the false alarm rate low. On the other hand, the miss rate increases fairly rapidly as the score threshold increases and the false alarm rate drops, a sign that the density of on-target stories among the best-scoring is not as high as it should be. This suggests that the language models are getting fooled by stories that score well but are actually off-target. A more discriminating language model, such as a bigram model, might help here.

4. THE FUTURE

It is remarkable that this simple application of HMM techniques to segmentation and tracking achieves such promising results. This work represents just the beginning of what can be achieved with this approach; many improvements are possible, both by incorporating ideas found in other work and from generalizations of the techniques we have already employed.

In particular, some form of story modeling that attempts to recognize features around boundaries (a key aspect of the approach in [1], for example), should be incorporated into our framework. One way to do this, which continues in the spirit of the speech recognition analogy, is to use "multi-node" story models, in which a story is modeled as a sequence of nodes (for example, one node which models the story start, one which models the middle, and one which models the end) rather than a single node corresponding to the topic.

It is also possible to improve the topic modeling that already forms the basis of the segmenter. Some methods of achieving this include using bigram models in place of unigram models for topics, and adaptively training the background during segmentation.

One key to improving performance on the event tracking task is better smoothing of the event models. For this, we are looking at ideas from information retrieval, such as using a retrieval engine to find smoothing material for a story from a database formed from the training data. Given that our performance improves rapidly with more training examples, this might dramatically improve the behavior of the system for small values of N_t. In general, we believe that this task requires a smoothing algorithm that aggressively preserves topic, something that is more suited to information retrieval techniques.

Dragon looks forward to implementing these ideas in a future system.

5. REFERENCES

[1] D. Beeferman, A. Berger, and J. Lafferty, "Text segmentation using exponential models," in *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing,* Providence, RI, 1997.

[2] M.A. Hearst, "Multi-paragraph Segmentation of Expository Text," in *Proceedings of the ACL,* 1994.

[3] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," in *IEEE Transactions on Acoustics, Speech and Signal Processing,* ASSP-35(3):400–401, March, 1987.

[4] H. Kozima, "Text Segmentation Based on Similarity between Words," in *Proceedings of the ACL,* 1993.

[5] D.J. Litman and R.J. Passonneau, "Combining Multiple Knowledge Sources for Discourse Segmentation," in *Proceedings of the ACL,* 1995.

[6] J.M. Ponte and W.B. Croft, "Text Segmentation by Topic," in *Proceedings of the First European Conference on Research and Advanced Technology for Digitial Libraries,* pp. 120–129, 1997.

[7] S. Wegmann, L. Gillick, J. Orloff, B. Peskin, R. Roth, P. van Mulbregt, and D. Wald, "Marketplace Recognition Using Dragon's Continuous Speech Recognition System," *Proceedings of the DARPA Speech Recognition Workshop,* Harriman, NY, February 1996.