Deconstructing broadcast news using all sources of input from the multimedia stream.

Machine Learning *OF* Event Segmentation *FOR* News on Demand

Stanley Boykin and Andrew Merlino

There are many different types of multimedia videos found in the world today—consider home videos, surveillance camera videos, television broadcasts as general categories. Commercial users, government personnel, and home consumers all have specific requirements to search these videos for topics and/or events. In order to support user query for these elements of interest, multimedia systems must segment and retrieve relevant segments of information. With advances in video digitization, annotation and extraction, automated multimedia processing systems are being created for many of the various video types. In these systems, event segmentation occurs manually, semiautomatically, or automatically.

Each type of multimedia video has varying levels of structure. For example, a home *video* may contain stories of a vacation, child's birthday party, and Christmas morning. The birthday party *story* may contain *events* of a child blowing out the candles, opening gifts, and playing games. In some stories, there may only be one event per story. The event pertaining to the child blowing out the candles may contain *shots* of the child's excitement of the oncoming cake, the friends singing, and the child blowing out the candles. Shots are segmented by scene changes. The shots may contain individual video *frames.* (See Figure 1.) The terminology (i.e., video, story, event, shots, frames) and hierarchy of these terms can be transposed to other types of media sources. These terms are usually not what the end user is interested in searching but assist automated segmentation routines.

In the DARPA community, there is a topic detection and tracking (TDT) initiative for broadcast news and newswire sources [10]. This effort defines topics, events, and activities as follows:

- Topic: A seminal event or activity, along with all directly related events and activities.
- Event: Something that happens at a specific time and place (a specific election, accident, crime or natural disaster are examples).
- Activity: A connected set of actions that have a common focus or purpose (for example, specific campaigns, investigations, and disaster relief efforts) [6].

When an end user uses a system that references indexed multimedia sources, the end user is more likely to query the system by the TDT defined topic, event, or activity. This is especially true when there is a large collection of varying multimedia sources. The multimedia hierarchy described earlier maps to the TDT terms as follows: There may be one or more TDT topics in a story, an event maps directly to one or more stories, and activities may span multiple events.

Within this TDT project, there are currently three tasks for evaluation:

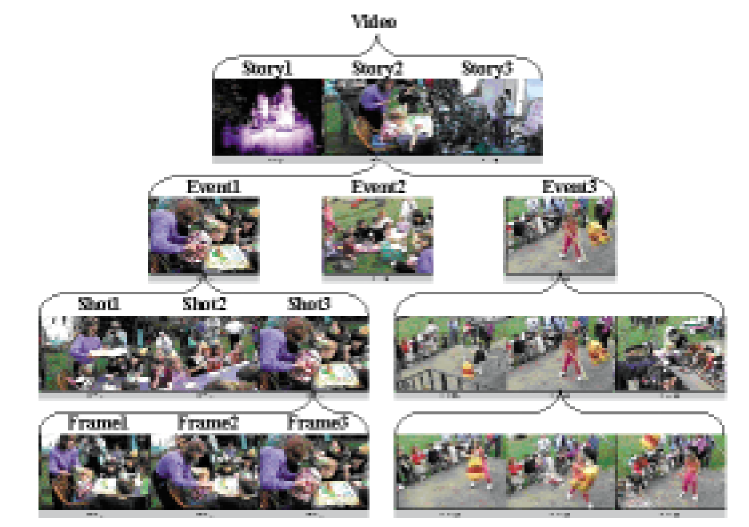
- Story Segmentation: Detection of story boundaries.
- Topic Tracking: Detection of stories that discuss a topic, for each given target topic.
- Topic Detection: Detection of stories that discuss an arbitrary topic, for all topics.

In order for the topic, event or activity retrieval effort to be useful to the end user, multimedia processing systems depend upon correct story segmentation, tracking and detection. In this article, we will discuss only story segmentation. The segmentation addressed in this article, unlike the TDT segmentation effort, is based on using all of the sources of input from the multimedia stream (audio, video, and text). The segmentation will be performed on broadcast news sources.

The MITRE-developed News on Demand (NOD) system Broadcast News Navigator (BNN) [7] contains a Finite State Machine (FSM) [1] based story segmentation routine. The segmentation routine allows the end user to view information by stories (See Figure 2). While skimming the story and associated metadata, the end user can select a story and view its associated named entities, video, or summary [9].

Within this article, we will compare BNN's FSM segmentation system with an automatically induced (machine-learned) segmentation system using hidden Markov models (HMMs). It is hoped that after reading this article, readers will understand that this HMM segmentation routine could be applied to other well-known NOD systems [3, 4, 11] and ultimately to other video types.

Figure 1. Hierarchy of video structure: video–story–sub-story–shot.

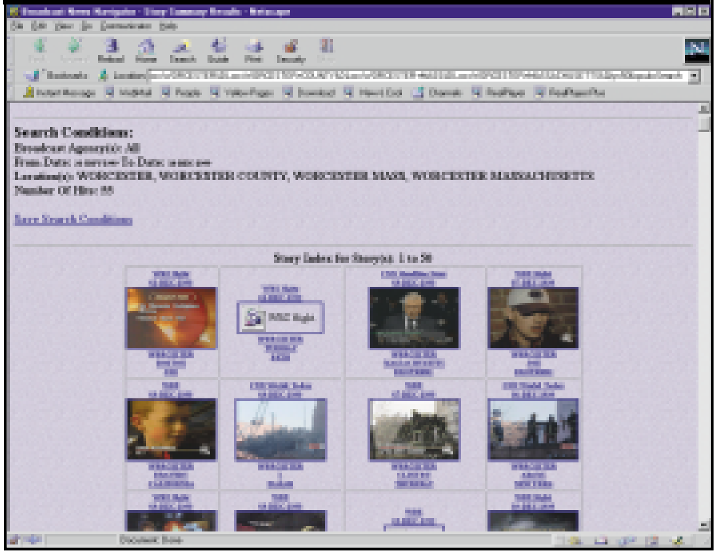


Gathering Truth for the Training and Evaluation

In order for our AI-based segmentation system to work successfully, we manually created a training set of annotated story segments. The training set consists of the programs and quantities shown in Table 1. In order to evaluate our system, we created an evaluation set of news programs. Similar to the training set, we manually created an evaluation set consisting of the programs and quantities shown in Table 2.

To gather the truth, two annotators watched each program and annotated the previously men-

Figure 2. Example Broadcast News Navigator story skim screen.



Search Conditions:
Broadcast Agency(s): All
From Date: [illegible] To Date: [illegible]
Location(s): WORCESTER, WORCESTER COUNTY, WORCESTER MASS, WORCESTER MASSACHUSETTS
Number Of Hits: 55

Save Search Conditions

Story Index for Story(s): 1 to 50

tioned broadcast, story, and advertisement begin- and end-frames using our multimedia video markup tool [8] (see Figure 3). Once truth was gathered from two annotators, an adjudicator reviewed the results and developed the final truth for the individual program. The adjudication was performed for each news program.

Why Use Hidden Markov Models?

We applied an HMM approach to perform event (story and advertisement) segmentation on news broadcasts. In general, an HMM structure is represented in three main pieces: by visible observations (facts and features that can be readily detected at a point in time); invisible (hidden) states that are abstract and not directly observable; and a set of probability vectors that reveal how the model is supposed to behave. One feature of HMMs is that if you witness a sequence of new observations for a particular model, you can use the previous behavior of that model on old observations to predict which hidden states the new observations describe. (This is the aspect of the Viterbi algorithm described later in this article.)

We decided to represent broadcast news as an HMM to see if we could improve upon our previous method of segmentation. Similar to our FSM model, we used a set of states to describe the different sections of a news program: Story Start, Advertisement, Other, and so forth. We hypothesized that since the detection of stories in the FSM model was based on non-adaptable, *manually created* transitions between these states, we could achieve accuracy in segmentation (in terms of precision and recall of event start boundaries) at least as comparable, and hopefully superior to, our previous method by using the *machine-learned*, HMM approach.

In our survey of other NOD systems, we noticed that several different techniques were used to perform story segmentation on broadcast news. In CMU's Informedia project [2] a method of applying a rule set to determine where story boundaries lie within a news broadcast is described; and similar to our system [1], these rules are based upon multimodal input (audio, video, and closed-captioned detections). Other systems exploit the features of a single mode of input. For example, BBN's Rough'n'Ready system [5] segments speech-transcribed text based on topics they generate on it. Zhang [12] describes a method of extracting stories by primarily using video-based features such as recognized anchor desk or weather report scenes. Our HMM-based method involves using a machine-learned model to segment broadcast news based on multimodal input.

The Machine-Learned Approach: Applying Broadcast News to an HMM

We first establish that any news broadcast can be split into equal time intervals 0,1, …, T. As mentioned in the previous section, the three pieces to an HMM consist of "visible" observations, "invisible"

Table 1. Training data.

Broadcast News Program	Type	Length (Minutes)	Quantity
CNN Headline News	Television Broadcast	30	5
CNN World View	Television Broadcast	30	5
CNN World Today	Television Broadcast	60	2
Fox News Now	Television Broadcast	60	2
MSNBC with Brian Williams	Television Broadcast	60	2
NWI International	Television Broadcast	30	2

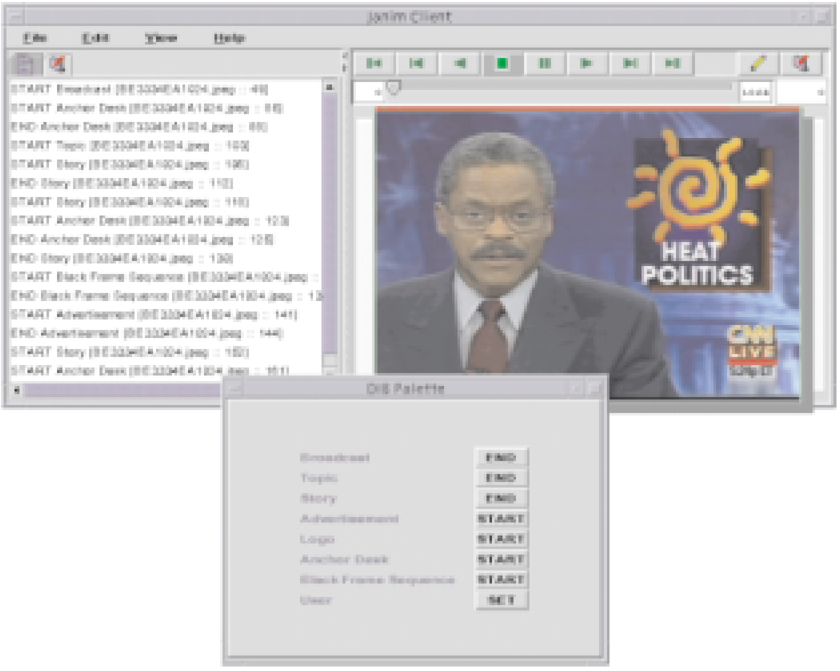
Table 2. Evaluation data.

Broadcast News Program	Type	Length (Minutes)	Quantity
CNN Headline News	Television Broadcast	30	1
CNN World Today	Television Broadcast	60	1
CNN World View	Television Broadcast	30	2
Fox News Now	Television Broadcast	60	1
MSNBC with Brian Williams	Television Broadcast	60	1
NWI International	Television Broadcast	30	1

states, and the behavior statistics that relate the two together. For our representation of broadcast news, the pieces fit together as described here:

- The visible observations consisted of the audio, video, and closed captioned metadata we detected from the multimedia source. Table 1 of [1] includes some of the media cues we used as observation features. An observation, therefore, is any combination of these features that were detected in a given interval of the broadcast.
- The hidden states we derived from the observations were:
  Story Start: The time between the beginning of a story boundary and an arbitrary time x seconds after the boundary. We chose to use x=5 seconds in this case because we noticed that the amount of detected observation features appeared to be greatest in this time period.
  Story End: The time between the end of the Story Start state and the end of the story boundary.
  Advertisement: The time between advertisement start and end boundaries.
  Other: We defined other segments to be periods in the broadcast that didn't neatly fit into any of the preceding categories. For example, anchor references to future broadcasts or anchor conversation—anything that didn't convey information on a particular news story.

Figure 3. Multimedia Workbench tool.



```
START Broadcast [BE3334EA1024.jpeg :: 49]
START Anchor Desk [BE3334EA1024.jpeg :: 88]
END Anchor Desk [BE3334EA1024.jpeg :: 88]
START Topic [BE3334EA1024.jpeg :: 103]
START Story [BE3334EA1024.jpeg :: 105]
END Story [BE3334EA1024.jpeg :: 112]
START Story [BE3334EA1024.jpeg :: 118]
START Anchor Desk [BE3334EA1024.jpeg :: 123]
END Anchor Desk [BE3334EA1024.jpeg :: 125]
END Story [BE3334EA1024.jpeg :: 130]
START Black Frame Sequence [BE3334EA1024.jpeg ::
END Black Frame Sequence [BE3334EA1024.jpeg :: 13
START Advertisement [BE3334EA1024.jpeg :: 141]
END Advertisement [BE3334EA1024.jpeg :: 144]
START Story [BE3334EA1024.jpeg :: 150]
START Anchor Desk [BE3334EA1024.jpeg :: 151]
```

Broadcast	END
Topic	END
Story	END
Advertisement	START
Logo	START
Anchor Desk	START
Black Frame Sequence	START
User	SET

- The three probability vectors that define the behavior of the news broadcasts were as follows (note that we are defining the set of possible states $q_0, \ldots, q_n$ and the set of possible observations $O_0, \ldots, O_T$):
  A = transition probability matrix = $a_{ij} = P(q_j \text{ at } t \mid q_i \text{ at } t\text{-}1)$: The probability the model is in state q_j in an interval of time, given that it was in q_i in the previous interval
  B = observation (output) probability matrix = $b_j(k) = P(O_k|q_j)$: The probability that observation O_k was witnessed in an interval of time, given that the model is in state q_j
  Π = initial probability vector $=\pi_i = P(q_j \text{ at } t = 0)$: The probability the model is in state q_j at the first time interval.

How Did We Train the HMM's Behavior Piece?

Each of the probability vectors had to be calculated from the analysis of the training data. The training data consisted of a set of programs for which we manually annotated the hidden states (so that we knew exactly when these states occurred in these broadcasts—we considered this set of data "truth"). We then created routines to calculate the statistics that, in turn, would describe our model's behavior. To compute the transition probability matrix (**A**), we tracked state progressions from interval to interval for each program in the training set. We created an algorithm to keep a count of all the transitions that occurred within the intervals of the training programs, and then calculated state transition probabilities from this data. For example, if in the training corpora there were 10 transitions into the 'Advertisement' state, and in eight of those occasions the previous state was also an 'Advertisement', then

$a_{Ad|Ad}$ = P('Advertisement' at t | 'Advertisement' at (t-1)) = (# of 'Advertisement' to 'Advertisement' transitions)/(total # of transitions to 'Advertisement') = 8/10 = 0.8.

The state transition matrix will be populated by calculating the probabilities for each state's likelihood to transition to any other state in the model.

Keeping track of observations and the states in which they were witnessed will derive the observation probability matrix (**B**). As another example, suppose there were 20 intervals in the training corpora in which only a speaker change was observed. If this observation took place in the 'Advertisement' state for 10 of those intervals, it naturally follows that b_{Ad}('speaker change') = P('speaker change' observed | 'Advertisement' state) = (# of 'speaker change' observations in 'Advertisement')/(total # of 'speaker change' observations) = 5/10 = 0.5.

By creating a routine to calculate these statistics for every observation in the training corpora, we build the observation matrix. In addition, the initial probabilities (∏) were computed based on the number of times a program would begin in a particular state. We can create the initial probability matrix by performing the same procedure for all of the HMM states.

So, for our purposes, generating the statistics that produce the behavior model essentially comes down to manipulating the training data. This allows the model to "learn" the probability vectors **A**, **B**, and ∏.

Imagine reviewing the events of a crisis situation by stitching together ATM videos, store surveillance cameras, broadcast cameras, and other possible sources.

Using the HMM and Viterbi to Segment Broadcast News

Once the HMM model has been trained, dynamic event segmentation can be performed for nightly news. We applied the principles behind the Viterbi algorithm to perform news event segmentation. The concept is that since we can witness observations (the metadata that we extract, broken into intervals) for a broadcast, we can use the behavioral vectors mentioned previously to construct the most likely state sequence for that broadcast. We then extract the story and advertisement starting times from this hidden state information. We describe this process in detail here.

Envision a news program as a grid, with each HMM state represented in all intervals of the broadcast t = 0, …, T. The reasoning behind Viterbi is that, at any time interval k between 1 and T, we can calculate the likelihood (δ) that the model is in any of those states at k. The state with the greatest likelihood is the most likely state the model is in at interval k; by "remembering" the states in previous intervals that led to this optimal state, Viterbi produces the best progression of states from 0 to k. Why? At the beginning of the program (t = 0), we can derive the likelihood of each state by using the probability vectors **B** and ∏:[1] $\delta_0(j) = \pi_j b_j(0)$.

Then, for every following interval t = 1, …, T, Viterbi calculates the "best" path to each state j (where j is the set of all states in the model) in the sequence using the vectors **A**, **B**, and the calculations of the previous costs to the states in the last interval (that is, the δ_{k-1} values):

$\delta_k(j) = \max_i[\delta_{k-1}(i)a_{ij}]b_j(k)$; i, j = {set of possible states}, $1 \le k \le T$

In other words: For each state in the model at the current time k, we determine the best path (and its cost) to that state from the states (and their costs) in the previous interval. We then multiply that best cost (remembering the state from which it was derived) with a) the probability of transitioning to the current state from the previous best state; and b) the probability of witnessing the current observation in the current state. (This produces the cost δ_k for each state in k.)

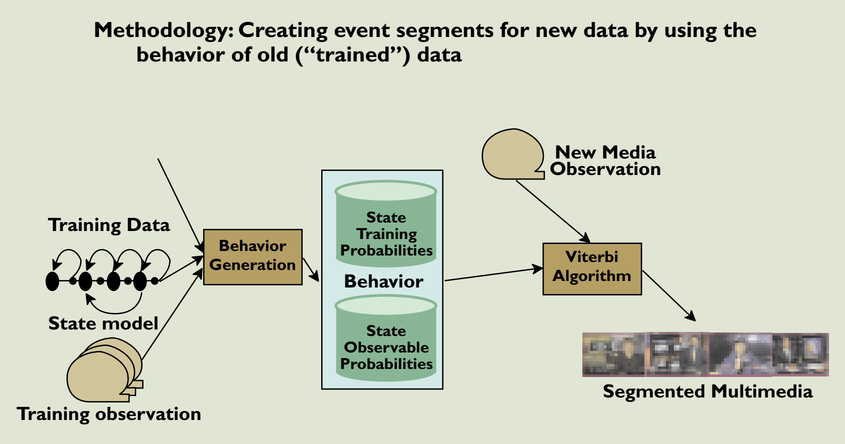
Once the algorithm reaches the last interval in the broadcast, the best path at that time would be the best path through the entire broadcast. To construct the most likely state sequence, we retrace our steps back to t = 0, identifying the best path states along the way. By using the Viterbi algorithm in this way, we can estimate which states occurred at which intervals. And by isolating 'Story Start' and 'Advertisement' state occurrences, we determine when these story and advertisement segments begin. The overview of the entire process is shown in Figure 4.

Results

After training the HMM and establishing the observations, we ran our evaluation data set against the FSM and the HMM. Note: both the FSM and HMM resided in a relational database to minimize the cost of retraining and rerunning the data sets. As seen in Figure 5a, as we increased the number of training programs for the generic model evaluation,

[1] Cohen, M. Hidden Markov Models: Introduction; screwdriver.bu.edu/cn760-lectures/l9/index.htm

Figure 4. Segmentation overview.


Methodology: Creating event segments for new data by using the behavior of old ("trained") data
Training Data
State model
Training observation
Behavior Generation
State Training Probabilities
Behavior
State Observable Probabilities
New Media Observation
Viterbi Algorithm
Segmented Multimedia

the story start FScore segmentation generally improved. The average FScore improvement of the HMM over the FSM was 0.24. Interestingly, the original FSM was modeled after the broadcast "CNN Prime News" (which no longer exists), and it was observed that the two CNN news programs showed a decrease in performance from the FSM to the HMM. As seen in Figure 5b, by building models for the specific news programs, the average segmentation improvement was almost zero. It is believed that this was due to the small training set sizes.

To the end user, the measurable improvement can be seen in the reduced time in processing an original news broadcast. The average time to create the FSMs currently used by BNN is 2.5 weeks. The average time to create training data and learn probabilities is a few days. With the accuracy of the HMM being comparable to the FSM, it is believed that creators of new multimedia segmentation systems will cost justifiably select to use the HMM approach.

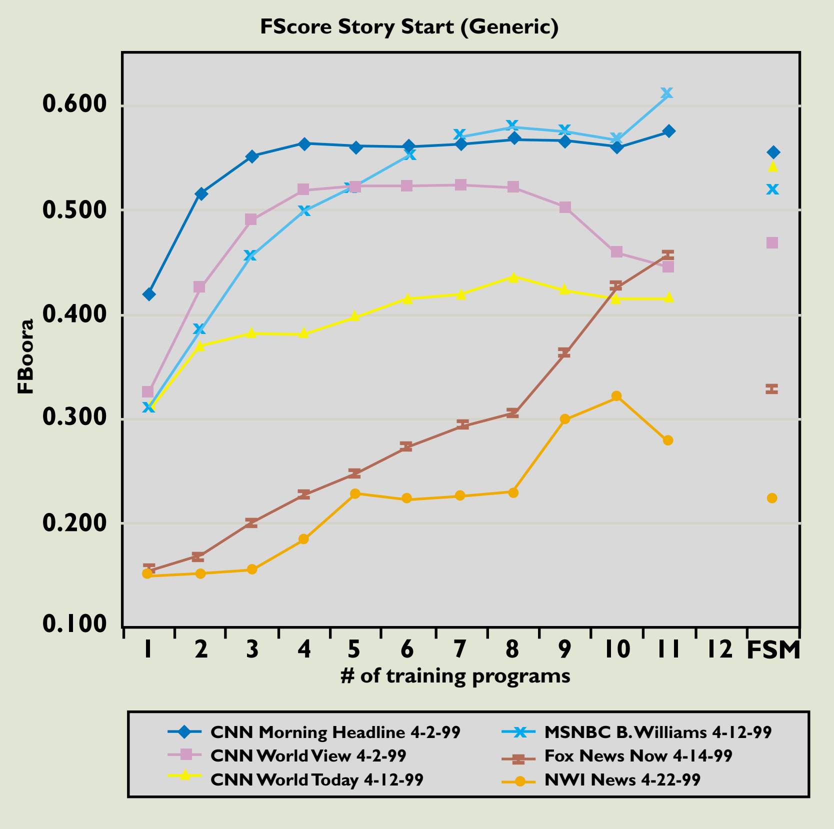
Future Work

During the evaluation process, we discovered that measuring the evaluation for exact detection down to the frame was very difficult. Thus, for future evaluations, we will build in the ability to evaluate segmentation boundaries over ranges of time. Currently, the TDT-2 effort gives 100% credit for exact segmentation detections and 50% for detections that are within 50 words or 15 seconds of a story boundary. We would like to vary the credit based on closeness to the correct story segmentation.

The accuracy of the time codes to the actual annotations is extremely important. During the evaluation process, we discovered that our transcription data was not 100% in synch with the other annotated data. We have already made modifications to our process flow to verify that annotations from any of the audio, video, and transcription streams are 100% in synch with one another. The problems inherent in using closed captioning can only be resolved by using word synchronization techniques as demonstrated by the CMU Informedia system.

Further work needs to be done on adding more

Figure 5a. FScore measurements for Story Start Segmentation using a generic model.

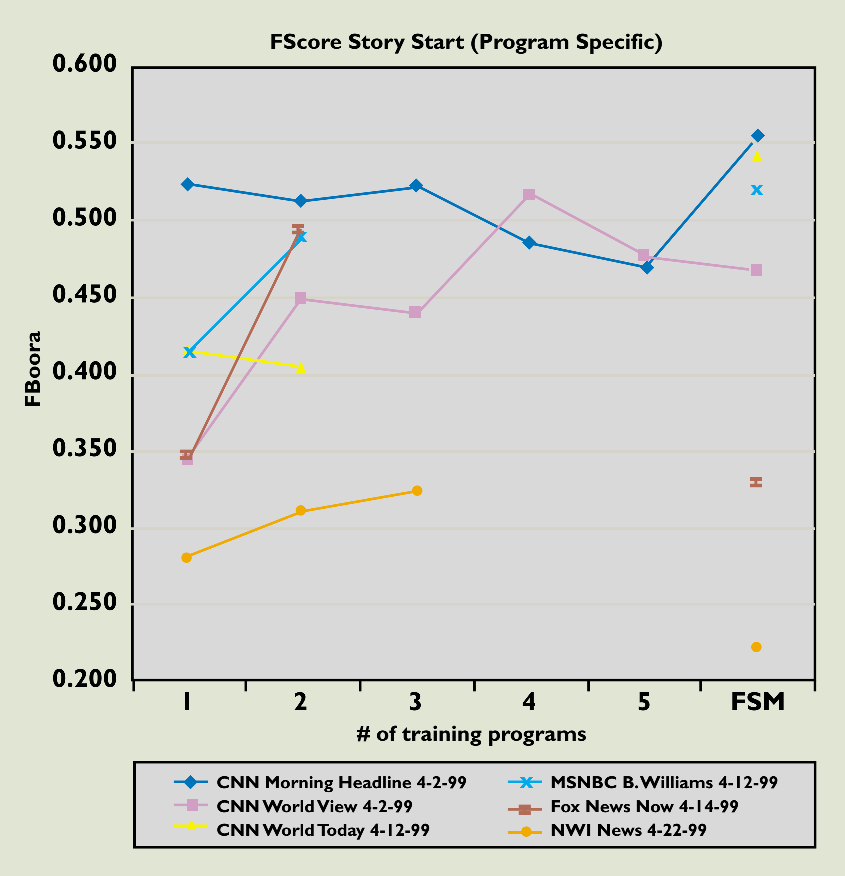


training data to our models to see when we reach the point that we are not affecting the FScore for segmentation. With our current training data, we can clearly show that we are not at that state yet. Another area of future work is to correlate temporally segmented streams using a geospatial dimension. For example, imagine reviewing the events of a crisis situation by stitching together sources and annotations from ATM videos, store surveillance cameras, broadcast cameras, and other possible sources.

Conclusion

The use of hidden Markov models to improve the automated detection of story segments has been desribed here. We have also shown that this technique will reduce the cost of quickly adapting to a new broadcast news program. The process of acquiring training data and evaluation data, training a model, running the results through the Viterbi algorithm and evaluating the results can be applied to other NOD systems as well as other domains. In our current research we are using this approach to evaluate the automated segmentation of unhelmed air vehicle surveillance video, collaboration video, and conference video. C

Figure 5b. FScore measurements for Story Start Segmentation using specific program models.



References

1. Boykin, S. and Merlino, A. Improving broadcast news segmentation processing. In *Proceedings of IEEE Multimedia Systems*, (Florence, Italy, June 1999), 744–749.
2. Hauptmann, A. and Smith, M. Text, speech and vision for video segmentation: The Informedia project. In M. Maybury, Ed., *Working Notes of IJCAI-95 Workshop on Intelligent Multimedia Information Retrieval*, Montreal, 1995.
3. Hauptmann, A. and Witbrock, M. Story segmentation and detection of commercials in broadcast news video. In *Proceedings of the Advances in Digital Libraries Conference* (Santa Barbara, CA, Apr. 1998).
4. Kubala, F. Intelligent collaboration and visualization track. *Information Management (IM) Intelligent Collaboration and Visualization (IC&V) PI Meeting*, (Oct. 1998, Hawaii), 1998.
5. Kubala, F., Colbath, S., Liu, D., Srivastava, A., and Makhoul, J. Integrated technologies for indexing spoken language. *Commun. ACM 43*, 2 (Feb. 2000).
6. Linguistic Data Consortium. *Annotation Guide.* morph.ldc.upenn.edu/TDT/Guide/label-instr.html
7. Maybury, M., Merlino, A., and Morey, D. Broadcast news navigation using story segments. In *Proceedings of the ACM International Multimedia Conference*, (Seattle, WA, Nov. 1997).
8. Merlino, A. and Dowling, W. *Multimedia Workbench*. MITRE Working Paper, MITRE, Bedford, MA, 1999.
9. Merlino, A. and Maybury, M. An empirical study of the optimal presentation of multimedia summaries of broadcast news. I. Mani and M. Maybury, Eds. *Advances in Automatic Text Summarization,* 1999.
10. Wayne, C. Topic detection and tracking (TDT) overview and perspective. DARPA Broadcast News, Transcription and Understanding Workshop, (Feb. 1998, Lansdowne, VA); www.nist.gov/speech/ tdt98/tdt98.htm
11. Sankar, A., Weng, F., Rivlin, Z., Stolcke, A., and Gadde, R.R. Development of SRI's 1997 broadcast news transcription system. In *Proceedings of the Broadcast News Transcription and Understanding Workshop*, (Feb. 1998, Lansdowne, VA).
12. Zhang, H.J., Low, C.Y., Smoliar, S.W., and Zhong, D. Video parsing, retrieval, and browsing: An integrated and content-based solution. In *Proceedings of ACM Multimedia '95* (San Francisco, CA), 1995.

Stanley Boykin (sboykin@mitre.org) is a senior database technology engineer at the MITRE Corporation in Bedford, MA.
Andrew Merlino (andy@mitre.org) is a department head at the MITRE Corporation in Bedford, MA.

The MITRE Research Program has funded the creation of the segmentation module as well as the news on demand system. BNN, the FSM and soon HMM modules are supported by DARPA and other agencies of the U.S. government.