News Story Segmentation in Multiple Modalities

Gert-Jan Poulisse[1], Marie-Francine Moens[1], Tomas Dekens[2], Koen Deschacht[1]
Katholieke Universiteit Leuven, Department of Computer Science[1]
Vrije Universiteit Brussel, Dept. of Electronics and Information Processing[2]
{Gert-Jan.Poulisse, Marie-Francine.Moens, Koen.Deschacht}@cs.kuleuven.be[1]
{tdekens}@etro.vub.ac.be[2]

Abstract In this paper, we describe an approach to segmenting news video based on the perceived shift in content using features spanning multiple modalities. We investigate a number of multimedia features, which serve as potential indicators of a change in story, in order to determine which are the most effective. The efficacy of our approach is demonstrated by the performance of our prototype, where a number of feature combinations demonstrate an up to 18% improvement in WindowDiff score compared to other state of the art story segmenters. In our investigation, there is no, one, clearly superior feature, rather the best segmentation occurs when there is synergy between multiple features. A further investigation into the effect on segmentation performance, while varying the number of training examples versus the number of features used, reveal that having better feature combinations is more important than having more training examples. Our work suggests that it is possible to train robust story segmenters for news video using only a handful of broadcasts, provided a good initial feature selection is made.

1. Introduction

The aim of our research is to implement accurate methods for story segmentation in news video. In this context, this means detecting the specific time event at which one news story stops being discussed and a new story starts. In text, a story is a coherent grouping of sentences, discussing related topics and names. The multimedia equivalent, such as found in news video, would be a temporal segment containing imagery accompanied by a spoken description of the single news event.

Three different channels, text, video, and audio are at our disposal for the segmentation task. Our aim is to base the segmentation decision on the detected change in content across the various media. Although considerable work has been done in developing story segmenters that utilize numerous multimodal features, we would like to investigate some of the text based features and methods developed in research to date. We wonder whether combining the various approaches into a single, unified segmentation algorithm might not improve performance of segmenting broadcast news video. In order to effectively operate on this multi-modal domain, we also include video and audio features in our investigation. Since our segmentation results form a basis for additional tasks, such as summarization and concept detection, we wish to obtain the lowest possible error rate and so we introduce supervision to our segmentation efforts. In order to do this, we train a maximum entropy classifier on various multimedia features.

In order to get an idea of how much the size of the training set affects the segmentation performance, we perform an additional experiment varying the training size for various feature combinations.

We briefly mention previous document segmentation approaches in section 2. In section 3 we discuss the various multimedia features that could aid us in the segmentation task. In section 4, we describe our proposed segmentation algorithm, and test it against two other state of the art segmenters in section 5. We also investigate the effect of variations in the number of training examples for various feature combinations. We analyze the results and conclude in sections 6 and 7.

2. Related Work

Initial efforts at topic segmentation in text determine the lexical cohesion by measuring vocabulary repetition, as expressed by the cosine score of the term vectors representing two adjacent blocks of text. Hearst [10] assigns a story break between text blocks whose cosine scores differ greatly. Increasingly sophisticated segmentation algorithms based on the cosine metric are presented in [4] and [9].

Other approaches such as [6, 13, 19] identify story segments by determining the semantic similarity of passages based on previously learned word collocations.

In a similar vein, latent semantic analysis is used by [8] and [15] to segment texts. In latent semantic analysis words are related to a topic space. The degree to which two topic spaces, generated by two segments, are similar can be used to determine whether they are part of the same story. Latent semantic analysis is also capable of recognizing that synonyms are related, because they will map to the same topic space.

[9] and [12] use entity repetition as expressed by lexical chains to compute a lexical cohesiveness score. This score is then used to identify story boundaries. Beeferman [2] uses language models, augmented by the use of cue words indicating a story shift, to determine cohesiveness.

Segmentation of spoken discourse includes work done by [9, 11, 17, 22], and makes use of a number of indicators such as cue-words, pause duration, and other forms of speech prosody.

[14] was an early attempt at combining multi-source features in order to segment video. They examined cue words, the presence of indirect vs. direct speech, lexical cohesion, and visual features, such as the presence of a face, which might indicate an interview.

Work done for the TRECVID 2004[1] story segmentation task (of news video) is noteworthy as the approaches taken are more grounded in video retrieval. Some examples are IBM [1], who combine numerous visual features with specialized commercial and anchor (news reader) detectors, speech prosody, and textual features in order to find story boundaries. Quenot [20] uses pause duration, shot cuts, rapid changes in audio, cue words, broadcaster specific jingle detectors, and anchor detectors as input to their story segmenter. Rather than adopting such a video-centric approach, like [21], our segmentation efforts on news video operate primarily on the text (transcribed speech), augmented by features from the video and audio modalities.

3. Multimedia Features

Our intent is to identify story boundaries, using sentences as the candidate points between which story breaks occur. This approach is standard in text-based approaches, but differs from video–based methods, which mark story boundaries at temporal locations. We chose this approach, as it seems more suited to follow-up tasks such as document summarization. Nonetheless, we incorporate a number of multimedia features from the video and audio channels.

The following sections describe the features extracted from a multimedia document, and the motivation behind their choice as indicators for topic shifts. Ultimately these features will be used to train a maximum entropy classifier, which will determine the existence of a story break at a particular sentence.

We are curious as to how our approach (described in detail in Section 4.), which uses a maximum entropy classifier, will compare to state of the art approaches of [4] and [9]. The results of this comparison are described in section 5. In contrast to them, we consider many more features, which we detail in the following sections.

3.1. Lexical Cohesion: Cosine Similarity

Vocabulary repetition has often been cited as a measure of detecting whether two adjacent passages are part of the same story or not. A common practice is to apply the cosine similarity measure over the term frequencies between two passages. Prior to doing this, we remove all punctuation, capitalization, and stop words. Given term vectors v_1 and v_2 from two successive passages, the cosine similarity function is expressed by:

[1] http://www-nlpir.nist.gov/projects/tv2004/tv2004.html

$$\text{cosine}(v_1, v_2) = \frac{\sum_j v_{1j} \times v_{2j}}{\sqrt{\sum_j v_{1j}^2 \times \sum_j v_{2j}^2}}$$

where v_{ij} is a term in v_i

A high score indicates that the two passages belong to the same story, while a low score implies the opposite, and suggests a story boundary.

Besides a standard cosine similarity computed from raw term frequencies, we also considered the cosine similarity using words that had been previously stemmed. We ultimately abandoned stemming, as it did not seem to affect segmentation performance.

3.2. Topic Similarity

Like [8] and [15], who use latent semantic analysis (LSA) to determine segment boundaries, we attempt to do the same by employing Latent Dirichlet Allocation [3] to measure the semantic change in content between two passages. A trained LDA maps words to a mixture of topic distributions. Given two topic distributions, A_i and B_i, derived by the LDA from two text passages, the change in topic distributions between them indicates whether they form one whole, or two separate, stories.

Our LDA model has 100 topics and is trained on a Reuters corpus. The change in topic distributions is measured by taking the symmetric Kullback-Leibler divergence of the topic distributions, A_i and B_i, respectively produced by two consecutive passages. Formally,

$$KL(A, B) = \frac{\sum_{i=1}^{T} A_i \log \frac{A_i}{B_i} + \sum_{i=1}^{T} B_i \log \frac{B_i}{A_i}}{2}$$

T is the total amount of topics in the LDA model. The resulting score is used as a feature in our classifier.

3.3. Lexical Cohesion: Likelihood

When considering whether to place a boundary at a candidate point candidate, one can gauge the effect of preserving the integrity of the story segment (original) versus splitting it at boundary candidate into two new story segments (new_segments), by computing the difference of the likelihoods that words within a segment are generated from the original segment or from one of the two new segments as per equation 1.

$$\text{Score}(\text{candidate}) = \frac{\mathcal{L}(\text{original})}{\mathcal{L}(\text{original}) + \mathcal{L}(\text{new\_segments})} \tag{1}$$

$$\mathcal{L}(\text{segment}) = \prod_i \text{words}_i \text{ in segment } \mathcal{L}(\text{word}_i \mid \text{segment}) = \prod_i \text{words}_i \text{ in segment } (\alpha \mathcal{P}(\text{word}_i \mid \text{segment}) + (1-\alpha)\mathcal{P}(\text{word}_i \mid \text{wiki})) \tag{2}$$

$$\mathcal{P}(\text{word}_i \mid \text{segment}) = \frac{\#\,\text{word}_i \text{ in segment -1}}{\text{total}\,\#\,\text{words in segment -1}} \tag{3}$$

The likelihood function measures term repetition within a segment smoothed by the chance of the term occurring naturally, as per equation 2, as defined by term frequencies gathered from a large external corpus, in our case Wikipedia. Experimentally we determined that α=0.7 performed well on a held out validation set. Because of its diversity, we consider this corpus to be topic neutral. The resultant score is used as a feature.

3.4. Layout Features: Program Structure

News broadcasts have a fairly structured format. The format may vary between different broadcasters, or depending on the particular time slot (the 8 o'clock news may be longer than the 10 o'clock news), but for a particular time slot the format is usually reasonably consistent, barring abnormalities, such as when there is a breaking news item right in the middle of a broadcast. We can make use of this consistency in the format to identify the points in the news broadcast where frequent topic shifts occur. For example, the broadcasts in our corpus are characterized by having a set of story highlights, lasting less than a minute at the beginning of the news program. This set of highlights is repeated at least once during the remainder of the program. Thus, although regular feature length stories produce story segment breaks at random intervals, the highlights segments in broadcasts have a tendency to occur at specific temporal positions within a news broadcast, usually during the opening and conclusion of the broadcast. We generate a distribution of story breaks at one-minute intervals based on our training data. This distribution was used to assign a probability score to every sentence in our test set, based on its timestamp. The resultant probability was used as a feature in our classifier.

3.5. Layout Features: Story Size

Another layout related feature that we kept track of is the story size of the previous segment. The reasoning behind this is that the highlights section of a news broadcast consists of many, short consecutive passages. Thus the presence of a short story segment, corresponding to such a story highlight, immediately preceding a candidate boundary point might in itself be a strong indicator. This feature is certainly domain driven, but not entirely inconceivable.

3.6. Speech Pauses

Work such as [17] and [22] have shown that speech prosody can contribute to the detection of story segments, with speaker pause duration often being the most important feature. Often when a news reader ends a particular story segment, there is a noticeable pause before the reader continues on with the next story, while the silences between sentences within the same story are much shorter. We therefore developed a voice-activity detector (VAD) based on work in [5] to extract the duration of all the silences in the audio channel.

Like most VAD algorithms, this algorithm makes use of an estimation of some background noise characteristics, which are then compared to the signal characteristics. The extent to which they differ is used to make the speech/pause decision. A VAD will only work as a true voice activity detector provided the signal consists of only background noise and speech. Music, or sounds whose signal characteristics differ greatly from the estimated baseline, may pose problems and may be incorrectly classified as speech. However, if these sounds are not defined as a silence, a VAD is well suited as a pause detector.

Pauses can be construed as portions of the audio signal with little energy, which can be detected by monitoring the instantaneous energy of the signal.

The feature used in the current VAD algorithm is the smoothed energy, contained in the frequency region of interest. Let Y(m,k) be the short-time Fourier transform of the input signal y(t), with m the frame number and k the frequency index. A frame is obtained by windowing the signal with a Hamming window. The smoothed energy is then calculated below, where N_{fft} is the number of fast-Fourier transform points used:

$$E(m) = \text{mean}\left\{ \frac{2}{N_{fft}} \sum_{k=k_1}^{k_2} \left|Y'(m+j,k)\right|^2 \right\}_{j=-N}^{j=+N} \quad 0 \le k_1, k_2 \le N_{fft}/2 \qquad (4)$$

Where $Y'(m,k)$ is the same as Y(m,k), except when k corresponds to DC or half the sampling frequency, then $Y'(m,k)$ is $Y(m,k)/\sqrt{2}$. This ensures that the energy at these frequency bins is only

considered once. One can see that the parameter N_{fft} in (4) can control the extent to which the feature is smoothed. By adjusting the range $[k_1,k_2]$ a certain frequency band can be selected. This could be useful if it is known that speech will only cover a fraction of the full signal frequency range, or if the noise energy is small compared to the signal energy in a certain frequency band.

During the initialization phase the first frames of the input signal are used to calculate the noise energy using (4); the mean of these noise frame energies gives us the initial estimation of the smoothed noise energy E_{noise}. This approach, however, is unsuited if the sound files do not start with a pause. In that case a certain percentage of the initial energy can be used as an estimation of the noise energy E_{noise}. Next, the smoothed energy is calculated for each signal input frame. This energy is then divided by E_{noise} and the logarithm is taken:

$$E_{ratio}(m) = 10\log_{10}\frac{E(m)}{E_{Noise}(m)} \tag{5}$$

This ratio is then compared to a threshold. When the ratio is smaller than this threshold the frame is considered to contain only noise and the noise energy is updated:

$$E_{Noise}(m+1) = \alpha E_{Noise}(m) + (1-\alpha)E(m) \quad 0 \le \alpha \le 1 \tag{6}$$

If the ratio is larger than the threshold, speech is detected and the current noise spectrum estimate will be kept. According to the expected signal-to-noise ratio of the input signals, an appropriate threshold value can be selected. Besides the relative energy ratio (5), an absolute power measure is also used. This can make the VAD deaf to signals whose power is below a certain value. In our pause detector, only when a sound's energy has sufficient power and is a certain dB above the estimated silence energy, will it not be classified as a silence.

Experimentation has shown that in our broadcasts, in particular for a feature length news story, the audio pause is indeed quite noticeable, averaging between 2-3 seconds where there is a story break. The audio silence between sentences without story breaks generally is around 1 second, and between words, less than 0.5 seconds.

Of note is that the speech silence detector samples at a very sensitive setting. As a result, many more pauses are detected than there are actual sentences. This is because intra-sentence pauses (between words) as well as inter-sentence (between sentences) pauses are detected. For the purpose of story shift detection, only inter-sentence pauses are of significance, as these potentially contain a newsreaders' cue of a topic shift through a long pause.

This requires aligning all the detected audio pauses with the corresponding sentences in the text. This is accomplished by identifying the longest silence fragment immediately preceding a sentence. This has the effect of also removing all intra-sentence pauses. We expect that long inter-sentence pauses are indicative of a story shift.

3.7. Shot Cut Detection

The rapid change in visual content, usually due to some rapid camera motion or change in scenery is referred to as a shot cut. Given a news broadcast, one would suppose that such a visual change would be correlated with a change in story content; that a change in visual content reflects a change in semantic content. Like [1] and [20], we use shot cuts from [16] as visual cues to check for story boundaries.

Unfortunately, shot cuts do not necessarily indicate a story boundary. There are generally many shot cuts within a single story unit. An additional complication is that shot cuts do not necessarily occur between two sentences, sometimes the visual transition will occur as the sentence is being read out.

In practice, the above considerations mean that each sentence in the document is given a binary feature, indicating the presence or absence of a shot cut during the time period in which the sentence is uttered. This time window is slightly offset, in order to catch shot cuts that occur immediately prior to, or after, the sentence is read.

We believe however, that the presence of a shot cut, in conjunction with other features, will indicate the presence of a topic break at a candidate sentence.

3.8. Cue Words

In many works, such as [2] and [9], cue words and phrases are used as a feature for the detection of topic breaks when segmenting single texts. For our purposes, cue phrases, such as "good morning," or "this is Alastair Yates," which are commonly said by news anchors or reporters to begin or end a particular news story, are useful indicators in detecting story breaks. We developed two procedures to obtain these cue phrases.

3.8.1. Chi-Square

We examined all the phrases occurring in sentences immediately preceding or following a story break in a training set, and compared them with how often they occurred in the rest of the corpus by applying the $\chi 2$ test at a significance level of 0.01. We display a selection of cue phrases whose $\chi 2$ values reject the null hypothesis (where $\chi 2 \geq 6.635$), thus indicating a correlation between a cue phrase and a story break.

Table 1. Cue words identified through their $\chi 2$ value

Phrase	**$\chi 2$ value**
hello and welcome to BBC news	8.69
good evening	14.62
stay with us	12.17
headlines	17.029

When performing feature extraction, each sentence receives a binary score indicating the presence or absence of cue phrases. This feature then is an added hint, lending emphasis to a classifier about the presence of a topic break at a candidate sentence.

3.8.2. Implicit Cue Words

We trained a maximum entropy classifier that returns the probability that the words of a sentence are indicative of a story boundary. The classifier was trained by taking each sentence in our training set, which was adjacent to a story boundary, as a positive training example. The sentences for which this was not the case were used as negative training examples. In this way, cue phrases are implicitly learned by the classifier. The probability score returned by this classifier in turn forms one of our features.

3.9. Lexical Chains: Named Entities

[9, 10, 12] use lexical chains to measure term repetition and thus infer the cohesiveness of a prospective story segment. We observed that stories read out by the news anchor often had disjoint sets of named entities. Applying a limited form of lexical chaining, specific only to named entities - i.e. the proper names of people, places, or organizations, the number of these chains spanning a particular text segment closely follows the actual story positions; boundaries occur where there are few chains.

It should be noted that longer interviews did not as closely follow this pattern as there generally were less named entities due to the dialogue. Also, very short story segments, such as the highlights section of a news broadcast, cannot be distinguished using this method as they generally have a similar amount of named entities.

We used the Stanford Named Entity Recognizer [7] to label all the named entities in the text. Within the context of a sliding window, chains were made linking sentences containing identical named entities. The size of the window was equivalent to a generous interpretation of the average size of a story in our training set, so that the next occurrence of a particular named entity had to occur within one window's length of the previous occurrence. This constraint was imposed to avoid the not too inconceivable case of two distinct stories occurring within one broadcast containing identical named entities.

We utilized the generated named entity chains in two ways. First, the number of chains at every sentence formed a feature, as sentences with few chains are potential boundary points. Secondly, at each candidate boundary, we computed the ratio of chains bisected by the boundary to the number of chains left intact. The resultant ratio indicates the coherence of a potential

segment. If the number of bisected chains dominates, this indicates that the candidate boundary is unlikely to be an actual boundary, whereas the converse is true if the number of intact chains dominates. Both features were passed to our classifier.

3.10. Lexical Chains: Galley

We include the score feature developed by Galley [9]. He combines the frequency of term repetition with chain compactness to arrive at a descriptor for lexical cohesiveness. This is then used to compute the rate of change in lexical cohesiveness, where a low score indicates a story boundary.

Repeated terms are collected into lexical chains, spanning an entire document. Each chain is then divided up into sub-chains when the distance between term occurrences exceeds a threshold value. These chains are then scored per equation (7).

$$\text{Chainscore}(R_i) = \text{freq}(t_i)\log(\frac{L}{L_i}) \tag{7}$$

Thus the score for chain R_i is the product of term frequencies given by $\text{freq}(t_i)$ and the log of the length of the whole document L divided by the length of the individual chain L_i. In this way, short, more compact chains are favored, as they more likely reflect the actual structure of the text. In order to plot the rate of change of lexical cohesiveness, LCF(c), a sliding window is passed over the text, such that when regions w_1 and w_2 are bisected by a candidate story boundary c:

$$\text{LCF}(c) = \text{cosine}(w_1, w_2) = \frac{\sum_j w_{1j} \times w_{2j}}{\sqrt{\sum_j w_{1j}^2 \times \sum_j w_{2j}^2}}$$

$$w_i = \begin{cases} \text{Chainscore}(R_i) & \text{if chain } R_i \text{ is present in } w_1, w_2 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The resultant plot of every LCF(c) illustrates the rate of change in lexical cohesion. Minima then, form potential story boundaries, whereas maxima correspond to the center of lexically coherent story segments. Given the LCF computed for a potential boundary to the left, l, right, r, of a candidate boundary c, the probability of a story boundary at c is expressed by:

$$p(c) = \frac{1}{2}(\text{LCF}(l)) + \text{LCF}(r) - 2\text{LCF}(c)) \tag{9}$$

Thus c is likely to be a story boundary when it is sandwiched between two very coherent segments, which occurs when LCF(l) and LCF(r) are maxima, and LCF(c) is a minimum. The resultant probability, indicating the likelihood of a boundary, forms a feature in our classifier.

4. Maximum Entropy Story Boundary Selection

The previous section described the various methods in which relevant features could be extracted indicative of a topic shift in a multimedia document. In order to select story boundaries for removal, and to select new positions for insertion, we use a fitness function that indicates the likelihood of a boundary at a certain position in the text. This function is a maximum entropy classifier that is trained on positive and negative examples from our training set. Positive examples are segments of text bounded by ground-truth story boundaries; negatives examples are all other potential text segments.

```
Let POS be the set of positive training examples in the training set.
Let NEG be the set of negative training examples in the training set.
TRAIN-MODEL(trainingSet)
1.  positive={POS}
2.  negative={}
```

```
3.  for i=1 to |NEG|
4.     k=randomPosition()
5.     if(!positive.contains(k) AND !negative.contains(k))
6.            negative.add(k)
7.     endif
8.  endfor
9.  positiveFeatures=extractFeatures(positive)
10. negativeFeatures=extractFeatures(negative)
11. maxentModel = trainClassifier(positiveFeatures, negativeFeatures)[2]
12. return maxentModel
```

In order to segment a new document, the number of story breaks, Q, has to either be specified or has to be estimated from the training data. In order to make this estimation, we adopt the following heuristic. We determine the ratio of sentences to story segments from the training data, and use this figure to determine the number of story breaks in any unseen documents from our test set. This approach is similar to other recent segmentation methods in literature, including amongst others, C99 and LCSeg. These other segmentation methods also either explicitly take the number of story segments as an input, or compute the number internally based on default parameter values during the segmentation process. We found that in practice our heuristic came close to the actual number of story segments.

After a random initialization, in which we place Q boundaries throughout the document, we use an iterative method to reassign story boundaries based on a fitness criterion; the probability of boundary assignment by a maximum entropy classifier trained using features from section 3. Over a number of iterations, we select one story boundary, and remove it from its current position (thus merging the two stories before and after the boundary position into one story) and insert it at another position in the document (thus splitting the existing story into two parts at that position).

During these iterations, we calculate the fitness scores of every boundary, which we use to generate a probability distribution over the current boundary positions. We then randomly select a position according to this distribution; a boundary with a low fitness scores is more likely to be selected. We remove this selected boundary and merge the two stories before and after the break. We then calculate the fitness score of every position S that is not a boundary in the text, again using this to generate a probability distribution over non-boundary positions. A new position is randomly chosen according to this distribution, such that positions with higher fitness scores are more likely to be selected. We break the existing story at that position in two, one story before the new boundary and one story after the boundary.

We perform P iterations, which in our case was 1000, and then store all the positions of the resultant boundaries. We then repeat this procedure with a new random initialization, again iterating a number of times and then storing the positions etc. We repeat this process M times, which in our case was 30. The resultant boundaries found are then averaged, so that the story boundaries are the positions where most often a boundary was placed during the iteration cycles. We can then return the amount of story segments desired by selecting that amount of boundary positions. We present this algorithm in the following pseudo-code below.

```
Let M be the number of times a document is randomly segmented before averaging
all segmentations.
Let Q be the number of story boundaries to find.
Let P be the number of iterations during which breaks are randomly added and
removed.
Let S be the number of possible candidate story boundary positions in the text.
DISCOVER-BREAKS(testDoc, maxentModel)
1.  counts=new int[S]
2.  for init=1 to M
3.     breaks={}
4.     for i=1 to Q
```

[2] This function invokes the maximum entropy classifier, which trains a model (maxentModel) based on the provided positive (positiveFeatures) and negative (negativeFeatures). We used the OpenNLP Maxent library: http://maxent.sourceforge.net

```
5.          k=randomPosition()
6.          if(!breaks.contains(k))
7.              breaks.add(k)
8.          endif
9.      endfor                          //breaks is now of size Q
10.     for j=1 to P
11.         REMOVE-RANDOM-BREAK(breaks, maxentModel)
12.         ADD-RANDOM-BREAK(breaks, maxentModel)
13.     endfor
14.     for each k in breaks
15.         counts[k] ++
//every break position found in each iteration is recorded here, over time
//certain positions will more often have a break, and thus counts[k] will have
//a higher value of to reflect this.
16.     endfor
17. endfor
18. finalBreaks = find-Q-Breaks(Q,counts)
19. return finalBreaks
```

```
REMOVE-RANDOM-BREAK(breaks, maxentModel)
1.  probOfRemoval = new double[breaks.size()]
2.  for each k in breaks                    //breaks is of size Q
3.      features=extractFeatures(k)
4.      probOfRemoval[k]=1-probabilityOfBreak(features, maxentModel)
5.  endfor
6.  k=randomPosition(probOfRemoval)
//randomly select a position based on the probability distribution generated.
//The complexity of helper function, randomPosition, is O(Q).
7.  breaks.remove(k)
```

```
ADD-RANDOM-BREAK(breaks, maxentModel)
1.  probOfBreak = new double[S]
2.  for j=1 to S
3.      if(breaks.contains(j))
4.          probOfBreak[j] = 0
5.      else
6.          features = extractFeatures(j)
7.          probOfBreak[j]=probabilityOfBreak(features, maxentModel)
8.      endif
9.  endfor
10. l=randomPosition(probOfNewBreak)
//randomly select a position based on the probability distribution generated.
//The complexity of helper function, randomPosition, is O(S).
11. breaks.add(l)
```

The complexity of helper function probabilityOfBreak is negligible as the maximum entropy model has been trained in advance and can be efficiently applied.

An important issue with story segmentation is that the placement of one story boundary influences the location of adjacent boundaries. This then raises the question of where to place the first boundary in a document. A greedy approach would place a boundary at the most probable location, but this very placement alters the probability of additional boundaries occurring nearby. If by some misfortune, the initial boundary placement was incorrect, successive assignments would only compound this error. In order to avoid this situation, our algorithm starts from a

random segmentation and by inserting and removing boundaries based on a fitness criterion, a trained maximum entropy classifier, converges over repeated trials to the most probable segmentation.

A brute force approach that places and evaluates Q boundaries at every possible position S in a document would have a complexity of:

$$\frac{S!}{Q!\ S-Q\ !}$$

In this brute force approach, which is computationally only feasible for short broadcasts, every possible segmentation of Q breaks is considered, and the best segmentation is chosen according to the fitness function discussed earlier.

While still trying a number of segmentations in our trials for added robustness, the use of a fitness function ensures that our algorithm ultimately converges to a likely story segmentation. The algorithmic complexity is then:

$$M \times (Q + P \times (2Q + 2S))$$

5. Evaluation

We have collected and annotated 14 news broadcasts from the BBC, which have a combined duration of around 7 hours. Annotation was done by one of the authors, based upon repeated viewings of the broadcasts. In addition, we have the corresponding Teletext (closed caption) transcripts for each broadcast, which consist of over 3000 sentences in total. The transcripts were sometimes noisy due to transmission errors, containing grossly distorted words and duplicates.

5.1. Evaluation Metrics

In text based segmentation, quite a few evaluation metrics have been proposed. Early papers such as [10] used the precision/recall metrics, although this metric is too strict in that it penalizes boundaries that have been placed very close to, but not on the ground-truth boundary. As a result, degenerate algorithms, which place a boundary after every possible sentence can actually achieve a higher precision and recall score. Beeferman [2] proposed a metric, P_k, which penalizes degenerate algorithms, yet also gives partial credit for boundaries which are close to the actual boundary. An improvement on the P_k metric, called WindowDiff (WD), was introduced by [18]. This metric increments a counter when the number of hypothesized boundaries, hyp_i, that occur within a window centered on each sentence differ from the actual number of story boundaries, ref_i. This number is then divided by the amount of possible candidate boundary positions S, and thus the WD metric expresses an error ratio.

$$WD(hyp, ref) = \frac{1}{S-k} \sum_{i=1}^{S-k} (|\, b(hyp_i, hyp_{i+k}) - b(ref_i, ref_{i+k}) \,| > 0)$$

The window size k is set as half the length of the average story size. S is the number of possible candidate boundaries, and b(i,i+k) is a function counting the number of boundaries between sentence i and sentence i+k. WindowDiff expresses an error ratio of how close the hypothesized boundaries are to the ground-truth boundaries, thus the lower the score the better the accuracy of the segmentation.

5.2. Segmentation Experiments

We have chosen to evaluate our system on our corpus of BBC broadcast recordings, using the WindowDiff(WD) metric. We performed leave-one-out (leaving out 1 broadcast every time) cross-validation to train and evaluate our segmentation algorithm. The average WD scores were computed from the held-out test sets. Our results were compared to Galley's segmenter [9] henceforth referred to as LCSeg, and Choi's segmenter [4] henceforth referred to as C99. These are two state of the art systems which segment text based on lexical cohesion. Our system however, also uses the available multimedia features. Both C99 and LCSeg use their default parameter values.

Table 2. Comparison of our baseline with other segmenters.

	WD
Baseline(known)	0.231
Baseline(unknown)	0.244
C99(known)	0.363
C99(unknown)	0.307
LCSeg(known)	0.276
LCSeg(unknown)	0.243

Known means the number of story boundaries was provided to each segmentation algorithm, unknown means the number had to be estimated.

We established a strong baseline using implicit cue words (3.8.2), story size (3.5), and likelihood (3.3). This can be seen in Table 2 by how the baseline compares favorably to segmentations produced by C99 and LCSeg. Results are for both modes of operation, when the number of segments was known and when it was estimated.

We then incrementally added other features to our baseline (where the number of segments is known), as seen in Table 3. This was done by combining the features into a single feature vector, which was passed to the maximum entropy classifier. While every possible combination was evaluated, the results presented below illustrate combinations which display a positive increase over the baseline for every additional feature.

Table 3. WD for various feature combinations and percent change from the baseline WD of 0.231

Baseline (known)	Location (3.4)	Pause (3.6)	Cosine (3.1)	NE count (3.9)	NE ratio (3.9)	Galley (3.10)	Cue words (3.8.1)	Shot cuts (3.7)	WD	% change from baseline WD	Late Fusion WD	% change after late fusion from original classifier
✓								✓	0.233	-0.85	-	-
✓							✓		0.232	-0.73	-	-
✓						✓			0.231	-0.10	-	-
✓									**0.231**	-	-	-
✓	✓	✓					✓		0.224	2.77	-	-
✓		✓							0.224	3.02	-	-
✓			✓			✓			0.223	3.22	-	-
✓				✓		✓			0.223	3.56	-	-
✓	✓								0.221	4.45	-	-
✓	✓	✓				✓	✓		0.220	4.82	-	-
✓			✓	✓		✓	✓		0.219	4.97	-	-
✓				✓					0.218	5.42	-	-
✓					✓				0.218	5.45	-	-
✓			✓						0.218	5.57	-	-
✓			✓	✓	✓		✓		0.217	6.04	-	-
✓	✓				✓	✓	✓		0.217	6.18	-	-
✓				✓	✓				0.216	6.30	-	-
✓			✓		✓		✓		0.216	6.56	-	-
✓			✓	✓	✓				0.215	6.75	-	-
✓	✓		✓	✓			✓		0.213	7.76	-	-
✓			✓	✓			✓		0.211	8.49	-	-
✓		✓			✓				0.211	8.57	-	-

✓					✓		✓		0.211	8.72	-	-
✓		✓	✓						0.211	8.73	-	-
✓			✓	✓		✓			0.210	8.80	-	-
✓		✓	✓	✓	✓		✓		0.210	8.89	-	-
✓			✓			✓			0.209	9.42	0.205	11.32
✓	✓	✓	✓						0.209	9.52	0.203	12.06
✓	✓		✓	✓	✓		✓		0.209	9.59	0.200	13.40
✓	✓		✓	✓		✓	✓		0.209	9.59	0.200	13.64
✓		✓	✓	✓					0.208	9.85	0.202	12.63
✓			✓		✓		✓	✓	0.208	9.95	0.201	12.74
✓		✓	✓				✓		0.208	9.96	0.197	14.72
✓		✓	✓	✓	✓	✓	✓		0.208	10.05	**0.190**	**17.78**
✓		✓	✓	✓				✓	0.206	10.65	0.208	10.04
✓		✓	✓	✓			✓		**0.205**	**10.98**	0.202	12.28

The table shows that all features made a positive contribution to the classification performance, but that the greatest increase comes from several features operating together in concert.

5.3. Varying Training Size Over Various Feature Combinations

We originally cleaned and annotated 14 BBC broadcasts as preparation for our experiments. Since we use leave-one-out cross validation, this gives 13 broadcasts to train on, and 1 for evaluation purposes. We were curious as to the effect of the number of training samples on the segmentation task, and to what extent the number of training examples required was affected by the amount of features used by a particular classifier.

Our experimental setup was as follows. We chose 27 feature combinations that used a varying amount of features. For each feature combination, we calculated the WD for every training set ranging in size from 1 to 13 broadcasts. The broadcasts making up each training set were randomly chosen, and to minimize the effect of an individual broadcast being an outlier, we took the average of three WD scores measured at each training set size. The resultant 27 by 13 matrix of WD scores was then analyzed using 2-way ANOVA without replication, at significance level 0.05, to discover the possible sources of variation in WD. The 2-way ANOVA test is an analysis of variance for two independent variables, in our case the number of features and the number of training examples. We present the ANOVA results as Table 4.

Table 4. 2-way ANOVA without replication at significance level 0.05

Source of Variation	*Sum of Squares*	*Degree of Freedom*	*Mean Square*	*F*	*P-value*	*F-critical*
Feature Combinations	0.0089536	26	0.00034437	17.877	9.94489E-46	1.531
Training Set Size	0.00139110	12	0.00011592	6.0177	9.13567E-10	1.783
Interaction Effect (Residual Error)	0.00550926	312	0.00001765			
Total	0.01585395	350				

The fact that the F for both Feature Combinations and Training Set Size exceed their respective F-critical values demonstrates that at significance level 0.05 both are significant sources of variance in WD. That said, the choice of features influences the WD three times more than the number of training examples ($F_{\text{Feature Combinations}} \approx 3 \times F_{\text{Training Set Size}}$). Since the Mean Square Interaction Effect is somewhat close to the respective Mean Squares for both Feature Combinations and Training Set

Size, we may conclude that there is some interaction between the size of the training set and a feature combination when assessing WD.

6. Analysis

All features described in this paper made a positive contribution to the classification performance. It should be noted that we omitted the topic similarity feature (3.2). In combination with the baseline, it gave a WD of 0.215, but the inordinate amount of computation required prevented further exploration of this feature.

An interesting observation is that most features on their own provide only a minimal increase above the baseline, but in synergy with each other the combined contribution to story segmentation performance is greatly increased.

The ANOVA results bear out the logical expectation that changes in training set size affect the segmentation performance. The surprising result however, is that the choice of features has a much greater impact than the training size.

7. Conclusion

Our initial belief, that a unification of several features and methods from the textual modality with additional multimedia-specific features would result in improved segmentation performance, was validated by our final result. Our best classifier, with a WD of 0.190, gave an increase of close to 18% over our initial baseline and two other segmentation algorithms we examined.

The fact that multiple features in combination with each other give a more robust performance is clearly demonstrated. This suggests our approach is suitable to the generic segmentation task, as with a small training corpus (13 broadcasts), a classifier can quickly be tailored to a specific corpus. Our investigation into the changes in WD when varying the training set size over various feature combinations bore this out. A good feature selection has a greater impact on segmentation performance than the training set size; three times as much according to the ANOVA test.

We acknowledge that many previous segmentation efforts in research have focused on unsupervised methods, yet we also feel we have more than adequately demonstrated the improved performance made possible by the use of a supervised training method. Given the small training set requirement, and future mission critical applications (document summarization and concept detection), this seems a justified choice, as the performance of our segmenter exceeds that of other methods by a significant margin.

Acknowledgments

The work reported is supported by the EU-IST project CLASS (Cognitive-Level Annotation using Latent Statistical Structure, IST-027978) and by the IWT-SBO project AMASS++ (Advanced Multimedia Alignment and Structured Summarization, IWT 060051).

References

[1] A. Amir, J. Argillander, M. Berg, S-F. Chang, et al. "IBM research TRECVID-2004 video retrieval system." Proceedings of TRECVID 2004.

[2] D. Beeferman, A. Berger, J. Lafferty. "Statistical models for text segmentation." Machine Learning, 1999: 34(1)177–210.

[3] D. M. Blei, A. Y. Ng, M. I. Jordan. "Latent Dirichlet Allocation." Journal of Machine Learning Research, 2003: (3)993-1022.

[4] F. Choi, "Advances in domain independent linear text segmentation." Proceedings of North American Chapter of the Association for Computational Linguistic, 2000.

[5] T. Dekens, M. Demol, W. Verhelst, F. Beaugendre. “Voice activity detection based on inverse normalized noise likelihood estimation.” Proceedings of Convention of Electrical Engineering 2007, Santa Clara, Cuba, June 18-22, 2007.

[6] O. Ferret, "Using collocations for topic segmentation and link detection." Conference on Computational Linguistics, 2002.

[7] J. Finkel, T. Grenager, C. Manning. "Incorporating non-local information into information extraction systems by Gibbs sampling." Proceedings of Association for Computational Linguistic, 2005.

[8] P. Foltz, W. Kintsch, T. Landauer. "The measurement of textual coherence with latent semantic analysis." Discourse Processes, 1998: (25)285-307.

[9] M. Galley, K. McKeown, E. Fosler-Lussier, H. Jing. "Discourse segmentation of multi-party conversation." Proceedings of Association for Computational Linguistics, Sapporo, Japan, 2003: 562-569.

[10] M. A. Hearst, "TextTiling: Segmenting text into multi-paragraph subtopic passages." Computational Linguistics, 1997: 23(1)33-64.

[11] J. Hirschberg, D. Litman. "Empirical studies on the disambiguation of cue phrases." Computational Linguistics, 1993: 19(3)501–530.

[12] M-Y. Kan, J. L. Klavans, K. R. McKeown. "Linear segmentation and segment significance." Proc. 6th Workshop on Very Large Corpora, 1998.

[13] H. Kozima, "Text segmentation based on similarity between words." Proceedings of Association for Computational Linguistics, 1993: 286-288.

[14] Y. Nakamura, T. Kanade. "Semantic analysis for video contents extraction - spotting by association in news video." ACM Multimedia, 1997: 393-401.

[15] A. Olney, Z. Cai. "An orthonormal basis for topic segmentation in tutorial dialogue." Procedings of Human Language Technology Conference/Conference on Empirical Methods in Natural Language Processing, 2005.

[16] M. Osian, L. Van Gool. "Video shot characterization." Machine Vision and Applications, 2004: 15(3) 172-177.

[17] R. J. Passonneau, D. J. Litman. "Intention-based segmentation: Human reliability and correlation with linguistic cues." Proceedings of Association for Computational Linguistics, 1993: 148-155.

[18] L. Pevzner, M. Hearst. "A critique and improvement of an evaluation metric for text segmentation." Computational Linguistics, 2002: 28(1)19-36

[19] J. M. Ponte, W. B. Croft. "Text segmentation by topic." European Conference on Digital Libraries, 1997: 113-125.

[20] G. Quenot, D. Moraru, S. Ayache, M. Charhad, M. Guironnet, L. Carminati, P. Mulhem, J. Gensel, D. Pellerin, L. Besacier. "CLIPS-LIS-LSR-LABRI experiments at TRECVID 2004." Proceedings of TRECVID 2004.

[21] N. Stokes, J. Carthy, A. Smeaton, "SeLeCT: A lexical cohesion based news story segmentation system." AI Communications, 2004: 17(1)3-12.

[22] G. Tur, D. Hakkani-Tüur, A. Stolcke, E. Shriberg. "Integrating prosodic and lexical cues for automatic topic segmentation." Computational Linguistics, 2001: 27(1)31–57.