Statistical Models for Text Segmentation

DOUG BEEFERMAN
ADAM BERGER
JOHN LAFFERTY

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Abstract. This paper introduces a new statistical approach to automatically partitioning text into coherent segments. The approach is based on a technique that incrementally builds an exponential model to extract features that are correlated with the presence of boundaries in labeled training text. The models use two classes of features: *topicality* features that use adaptive language models in a novel way to detect broad changes of topic, and *cue-word* features that detect occurrences of specific words, which may be domain-specific, that tend to be used near segment boundaries. Assessment of our approach on quantitative and qualitative grounds demonstrates its effectiveness in two very different domains, *Wall Street Journal* news articles and television broadcast news story transcripts. Quantitative results on these domains are presented using a new probabilistically motivated error metric, which combines precision and recall in a natural and flexible way. This metric is used to make a quantitative assessment of the relative contributions of the different feature types, as well as a comparison with decision trees and previously proposed text segmentation algorithms.

1. Introduction

The task we address in this paper seems on the face of it rather elementary: construct a system which, when given a stream of text, identifies locations where the topic changes. This work was motivated by the observations that such a seemingly simple problem can actually prove quite difficult to automate, and that a tool for partitioning undifferentiated text, speech, or video into coherent regions would be of great benefit in a number of settings.

The task itself is ill-defined: what exactly is meant by a topic break? We adopt an empirical definition. At our disposal is a collection of online data (including a corpus of Wall Street Journal articles and a separate corpus of broadcast news transcripts, both containing several million words) annotated with the boundaries between regions—articles or news reports, respectively. Given this input, the task of constructing a text segmenter may be cast as a problem in machine learning: learn how to place breaks in unannotated text by observing a set of labeled examples.

Though we have equated topic boundaries and document boundaries, in general documents are often comprised of multiple topics. Since many of the clues the system learns to identify with an article or news story boundary are related to the change in topic between one document and another, our approach could be applied to the task of topic segmentation. However, this paper focuses exclusively on the design of algorithms and evaluation procedures for the document segmentation task, and does not address the closely related task of sub-topic segmentation.

A general-purpose tool for partitioning text or multimedia into coherent regions will have a number of immediate practical uses. In fact, this research was inspired by a problem in information retrieval: given a large unpartitioned collection of expository text (such as a year's worth of newspaper articles strung together) and a user's query, return a collection of coherent segments matching the query. Lacking a tool for detecting topic breaks, an IR application may be able to locate positions in its database which are strong matches with the user's query, but be unable to determine how much of the surrounding data to provide to the user. This can manifest itself in quite unfortunate ways. For example, a video-on-demand application (such as the one described in Christel et al. [4]) responding to a query about a recent news event may provide the user with a news clip related to the event, followed or preceded by part of an unrelated story or even a commercial.

We take a feature-based approach to the problem of detecting segment boundaries. The field of machine learning offers a number of methods—such as decision trees and neural networks—to integrate a set of features into a decision procedure. We use statistical techniques based on exponential models for selecting and combining features into a predictive model. The rest of the paper will focus on this technique and its application to the segmentation problem.

In Section 2 we review some previous approaches to the text segmentation problem. In Section 3 we describe the statistical framework that we use for model building. After reviewing some language modeling basics in Section 4, we describe in Section 5 the candidate features that we make available to our feature selection algorithm. Section 6 shows examples of the algorithm in action. Since the algorithm is computationally expensive, we introduce in Section 7 some methods for speeding up the learning process. In Section 8 we introduce a new, probabilistically motivated metric for evaluating a segmenter. Finally, in Section 9 we report on a series of experiments to compare different approaches to the segmenting problem.

2. Some Previous Work

2.1. Approaches based on lexical cohesion

Several proposed approaches to the text segmentation problem rely on some measure of the difference in word usage on the two sides of a potential boundary: a large difference in word usage is a positive indicator for a boundary, and a small difference is a negative indicator.

The *TextTiling* algorithm, introduced by Hearst [9], is a simple, domain-independent technique that assigns a score to each topic boundary candidate (inter-sentence gap) based on a cosine similarity measure between chunks of words appearing to the left and right of the candidate. Topic boundaries are placed at the locations of valleys in this measure, and are then adjusted to coincide with known paragraph boundaries.

TextTiling is straightforward to implement, and does not require extensive training on labeled data. However, TextTiling is designed for a slightly different problem than the one addressed in this study. Since it is designed to identify the subtopics within a single text and not to find breaks between consecutive documents [10], a comparison of TextTiling with the system we propose is difficult. Furthermore, TextTiling segments at the paragraph level, while this work doesn't assume the presence of explicit paragraph boundaries. Applications such as video retrieval may use speech recognition transcripts or closed captions that lack structural markup. Since TextTiling is widely used and implemented, we examine its behavior on our task in Section 9.

Another approach, introduced by Reynar [21], is a graphically motivated segmentation technique called *dotplotting*. This technique depends exclusively on word repetition to find tight regions of topic similarity.

Instead of focusing on strict lexical repetition, Kozima [14] uses a semantic network to track cohesiveness of a document in a *lexical cohesion profile*. This system computes the lexical cohesiveness between two words by "activating" the node for one word and observing the "activity value" at the other word after some number of iterations of "spreading activation" between nodes. The network is trained automatically using a language-specific knowledge source (a dictionary of definitions). Kozima generalizes lexical cohesiveness to apply to a window of text, and plots the cohesiveness of successive text windows in a document, identifying the valleys in the measure as segment boundaries.

2.2. Combining features with decision trees

Passoneau and Litman [19] present an algorithm for identifying topic boundaries that uses decision trees to combine multiple linguistic features extracted from corpora of spoken text. These include prosodic features such as pause duration, lexical features such as the presence of certain cue phrases near boundary candidates, and deeper semantic questions such as whether two noun phrases on opposite sides of a boundary candidate corefer.

Passoneau and Litman's approach, like ours, chooses from a space of candidate features, some of which are similar to the cue-word features we employ. Their cue phrases are drawn from an empirically selected list of words [11], while in our approach we allow all of the words in a fixed vocabulary to participate as candidate features.

2.3. TDT pilot study

The Topic Detection and Tracking (TDT) pilot study [6] carried out during 1997 was a DARPA-sponsored research program to assess and advance the state of the art in technologies for managing large amounts of information in the form of newswire, TV and radio broadcasts. The study was organized around three specific tasks: segmenting sources into stories, detecting the occurrence of new events, and tracking labeled events in the data stream. The participants in the pilot study included research groups from Carnegie Mellon, the University of Massachusetts, and Dragon Systems.

The TDT study led to the development of several new and complementary approaches to the segmentation problem, and these approaches were quantitatively evaluated using the metric described in Section 8. Yamron [25] developed an approach to segmentation that treats a story as an instance of some underlying topic, and models an unbroken text stream as an unlabeled sequence of topics using a hidden Markov model. In this approach, finding story boundaries is equivalent to finding topic transitions, and the stories are generated using unigram language models that depend on the hidden class of the segment. Ponte [20] developed an approach based on information retrieval methods such as local context analysis [23], a technique that uses co-occurrence data to map a query text into semantically related words and phrases. A comparison of these techniques appears in [24].

3. A Feature-Based Approach Using Exponential Models

Our approach to the segmentation problem is based on the statistical framework of feature selection for random fields and exponential models [2, 5]. The idea is to construct a model that assigns a probability to the end of every sentence—the probability that that there exists a boundary between that sentence and the next. This probability distribution is chosen by incrementally building a log-linear model that weighs different "features" of the surrounding context. For simplicity, we assume that the features are binary questions.

To illustrate (and to show that our approach is in no way restricted to text), consider the task of partitioning a stream of multimedia data containing audio, text and video. In this setting, the features might include questions such as:

- *Does the phrase* COMING UP *appear in the last utterance of the decoded speech?*
- *Is there a scene change in the video stream in the last 20 frames?*
- *Is there a "match" between the current image and an image near the last segment boundary?*
- *Are there blank video frames nearby?*
- *Is there a sharp change in the audio stream in the next utterance?*

The idea of using features is a natural and common one in machine learning, and indeed other recent work on segmentation adopts this approach [18]. Our approach differs in how we collect and incorporate the information provided by the features, as described below.

3.1. *Feature selection*

We split the task of constructing a text segmenter into two subtasks:

a) Build a model $q(b \mid X)$, where $b \in \{\text{YES}, \text{NO}\}$ is a random variable corresponding to the presence (or absence) of a segment boundary in the context X.

b) Specify a decision procedure which, based on the values $q(\text{YES} \mid X)$ generated by applying the model to an input corpus, produces a list of hypothesized locations of segment boundaries within the corpus.

We take up the first of these tasks in this section, and defer a discussion of the decision procedure to Section 9.

By a *context* X, we mean a word position in a text corpus together with the surrounding K words on either side. Thus, the context X_i at position i can be represented as a word sequence

$$X_i = w_{i-K}, w_{i-K+1}, \ldots, w_{i-1}, w_i, w_{i+1}, \ldots, w_{i+K-1}, w_{i+K}\,.$$

Of course, if K is large, a given context is likely to appear only once in the corpus. In our experiments, K is on the order of 500 words, so each context is almost surely unique.

If the corpus is annotated with segment boundaries, then we can think of the segments as being given by an assignment of a label $b_i \in \{\textsc{yes}, \textsc{no}\}$ to each position i, where $b_i = \textsc{yes}$ in case there is a segment boundary between words w_{i-1} and w_i, and $b_i = \textsc{no}$ otherwise. This annotation then defines an *empirical distribution* $\tilde{p}(b \,|\, X)$ of labels over the contexts that appear in the corpus:

$$\tilde{p}(\textsc{yes} \,|\, X) = \frac{\#(\textsc{yes}, X)}{\#(\textsc{yes}, X) + \#(\textsc{no}, X)}$$

where $\#(\textsc{yes}, X)$ is the number of times that context X is labeled with a boundary, and $\#(\textsc{no}, X)$ is the number of times that it is not. If each context is unique, then $\tilde{p}(\textsc{yes} \,|\, X)$ is always either 0 or 1.

If there are N words in the corpus, then the empirical distribution of contexts X is given by

$$\tilde{p}(X) = \frac{\#(X)}{N}$$

where $\#(X)$ is the number of times that the context appears in the corpus. Again, for practical purposes this can be thought of as simply $\frac{1}{N}$.

Our choice of domain determines the distribution $\tilde{p}(X)$, and the labeling of the "true" segment boundaries in that domain determines the distribution $\tilde{p}(b \,|\, X)$. The modeling problem is to construct a distribution $q(b \,|\, X)$ that closely approximates the empirical distribution $\tilde{p}(b \,|\, X)$ when X is drawn from $\tilde{p}(X)$. The training sample that we are given to learn from can then be thought of as a number of examples $(X_1, b_1), \ldots, (X_T, b_T)$ drawn from the joint distribution $\tilde{p}(X, b)$. The degree to which $q(b \,|\, X)$ approximates $\tilde{p}(b \,|\, X)$ (in other words, the quality of the model q) is judged in terms of the Kullback-Leibler divergence

$$D(\tilde{p} \,\|\, q) = \sum_X \tilde{p}(X) \sum_{b \in \{\text{YES},\text{NO}\}} \tilde{p}(b \,|\, X) \log \frac{\tilde{p}(b \,|\, X)}{q(b \,|\, X)}\,.$$

When we hold $\tilde{p}$ fixed and search for a model $q(b \,|\, X)$, we can express this divergence as

$$D(\tilde{p} \,\|\, q) = -\sum_X \sum_b \tilde{p}(X, b) \log q(b \,|\, X) + \text{constant}(\tilde{p})$$

The first term on the righthand side above is $-\frac{1}{N}$ times the log-likelihood $L(q)$ of the model q with respect to the empirical sample. Thus, by minimizing the divergence we are in fact maximizing the likelihood that the model assigns to the data.

The challenge is to build a distribution that accounts for the training sample $\{(X_i, b_i)\}$ without overfitting, by learning the salient features of the examples. Toward this end we consider distributions in the *linear exponential family* $\mathcal{Q}(f)$ given by

$$\mathcal{Q}(f) = \left\{ q(b \,|\, X) = \frac{1}{Z_\lambda(X)} e^{\lambda \cdot f(X,b)} \right\}$$

where $\lambda \cdot f(X,b)$ is a linear combination of binary features $f_i(X,b) \in \{0,1\}$ with real-valued *feature parameters* λ_i:

$$\lambda \cdot f(X,b) = \lambda_1 f_1(X,b) + \lambda_2 f_2(X,b) + \cdots \lambda_n f_n(X,b)\,.$$

The normalization constants

$$Z_\lambda(X) = e^{\lambda \cdot f(X,\mathrm{YES})} + e^{\lambda \cdot f(X,\mathrm{NO})}$$

insure that this is indeed a family of conditional probability distributions. In our experiments we limit the class of feature functions to those that depend only on the context: $f(X,b) = f(X)$, and combine them so that

$$q(\mathrm{YES} \,|\, X) = \frac{1}{Z_\lambda(X)} e^{\lambda \cdot f(X)}$$

with $Z_\lambda(X) = 1 + e^{\lambda \cdot f(X)}$. Thus, our model is a form of additive logistic regression [8].

It can be shown that the maximum likelihood solution

$$q^\star = \operatorname*{arg\,min}_{q \in \mathcal{Q}(f)} D(p \,\|\, q) = \operatorname*{arg\,max}_{q \in \mathcal{Q}(f)} L(q)$$

exists and is unique. There are a number of "iterative scaling" algorithms for finding $q^\star$, all of which incrementally adjust the parameters $\lambda_1 \ldots \lambda_n$ until some convergence criterion applies. In the iterative step, a parameter λ_i is updated to $\lambda_i' = \lambda_i + \Delta\lambda_i$, where

$$\Delta\lambda_i = \frac{1}{M} \log \left(\frac{\sum_{X,b} \tilde{p}(X,b) f_i(X,b)}{\sum_{X,b} \tilde{p}(X) q_\lambda(b \,|\, X) f_i(X,b)} \right)$$

where q_λ is the model with parameters $\lambda_1, \ldots, \lambda_n$ and M is a constant. This formula makes clear that the algorithm is choosing the model so that the features' expected values with respect to the model are the same as their expected values with respect to the data. One can also employ the "improved iterative scaling" algorithm [5], which uses a slightly different update procedure, to achieve faster convergence.

This explains how to construct a model from a set of features $f_1, \ldots, f_n$, but how are these features to be found? The procedure that we follow is a greedy algorithm akin to growing a decision tree. Given a set of candidate features $\mathcal{C}$ and an initial exponential model q, let $q_{\alpha,g}$ denote the model q modified by the single feature $g \in \mathcal{C}$ with weight α:

$$q_{\alpha,g}(b \mid X) = \frac{e^{\alpha g(X,b)} q(b \mid X)}{e^{\alpha g(X,\text{YES})} q(\text{YES} \mid X) + e^{\alpha g(X,\text{NO})} q(\text{NO} \mid X)} .$$

The *gain* of the candidate feature g relative to q is then defined to be

$$G_q(g) = \sup_\alpha \Big(D(\tilde{p} \,\|\, q) - D(\tilde{p} \,\|\, q_{\alpha,g}) \Big) .$$

The gain $G_q(g)$ is the largest possible improvement to the model, in terms of reduction in divergence, that would result from adding the feature g and adjusting only its weight. After calculating the gain of each candidate feature, that candidate yielding the largest gain is added to the model, and all of the model's parameters are then adjusted using iterative scaling. Repeating this procedure yields an exponential model containing the most informative features.

Algorithm 1: Feature Selection for Exponential Models

Input: Collection of candidate features $\mathcal{C}$, training samples $\{X_i, b_i\}$, and desired model size n

Output: Selected features $f_1 \ldots f_n$ and their maximum-likelihood parameters $\lambda_1 \ldots \lambda_n$.

1. Set $i \leftarrow 1$, and let $q^{(0)}$ be uniform.
2. For each candidate feature $g \in \mathcal{C}$, compute the gain $G_{q^{(i-1)}}(g)$.
3. Let $f_i = \arg\max_{g \in \mathcal{C}} G_{q^{(i-1)}}(g)$ be the feature yielding the largest gain.
4. Compute $q^\star = \arg\max_{q \in \mathcal{Q}(f)} L(q)$ to obtain weights $\lambda_1, \lambda_2 \ldots \lambda_i$, using improved iterative scaling.
5. Set $q^{(i)} \leftarrow q^\star$.
6. If $i = n$ then exit.
7. Set $i \leftarrow i + 1$ and go to step 2.

Constructing a model with n features requires, according to Algorithm 1, n iterative scaling computations and n rankings of the candidate features. But does feature selection necessarily require so much work? A reasonable shortcut might be to select several of the top-ranked features in step 3. We will take up this matter of efficient construction in Section 7.1, where we provide some empirical results to illustrate the time-quality tradeoffs one can make during feature selection.

3.2. *Example: flipping coins*

A simple example of feature selection may help to explain some of the subtleties which arise in the segmentation applications we present in the following sections.

Suppose we flip a biased coin, whose probability of heads is $p(\mathtt{H}) = \frac{2}{3}$ and whose entropy is thus $H(\frac{2}{3}) \approx 0.918$ bits. Depending on the outcome of this random variable, we answer 50 questions in the following manner. For the i-th question, with probability $\left(\frac{9}{10}\right)^i$ the answer is YES (1) if the coin came up heads and NO (0) if the coin came up tails. With probability $1 - \left(\frac{9}{10}\right)^i$ the answer to the question is chosen uniformly at random. A sample of such events is exhibited below.

```
H 11111110101010100001100110110110101101000000001111
T 00100100010000110001111111110001000111100110111100
H 11111011011000100110101100100101110010101111100101
H 11111111100000011110001010111100101001100000110010
T 000010000000000010001001100010001011111101101010000
H 11111111101101111001111110001111111111100000100111
H 11100111111111111100101010101010101101000111100100
T 00001000110011101010001000010001011101111010001111
H 11011010110111101000101000011111111101100010101101
T 00100010011010100010000101101001001000111000000101
```

To learn the posterior distribution $p(b \,|\, X = X_1 \cdots X_{50})$ of the coin given the bitstrings, we carried out feature selection on 100 synthetic events of this form. Figure 1 shows the first six induced features. The table annotates each feature with the gain that resulted from adding the feature, the cross entropy of the model after the feature was added, and the initial and final values of the parameter $\beta = e^{\lambda}$.

Not surprisingly, the first feature chosen is the first bit, which by construction should be the most informative bit for predicting the outcome of the coin. Out of the 100 events that were generated, it happened that the first bit disagreed with the coin only four times, and among these events only one of the coin tosses was a tail. Thus, the first feature constrains the conditional probability of heads to be $p(\mathtt{H} \,|\, X, X_1 = 1) = 0.99$. After five iterations of iterative scaling training, this model probability is $p(\mathtt{H} \,|\, X, X_1 = 1) = \frac{41.5}{1+41.5} \approx 0.98$. After the first feature then, the cross entropy of the model with respect to the true distribution is $\frac{2}{3}H(0.98) + \frac{1}{3}H(0.5) \approx 0.427$ bits, and the cross entropy with respect to the actual 100 events turns out to be 0.435 bits.

The second feature chosen queries the 32nd bit. The weight λ initially used for this feature is $\lambda = \log(0.20) \approx -1.61$. As a consequence, the effect of this feature is to

feature	gain	entropy	initial β	final β
bit 1	0.563	0.435	41.5	54.7
bit 32	0.114	0.315	0.20	0.16
bit 35	0.024	0.280	0.45	0.29
bit 2	0.026	0.246	2.93	2.65
bit 16	0.035	0.206	0.34	0.25
bit 5	0.023	0.177	2.15	2.15

Figure 1. Statistics on the first six features induced for the toy coin-flipping example. The gain is the reduction in entropy after the feature is added. The rightmost column lists the value of the parameter $\beta = e^{\lambda}$ after all six features are added and iterative scaling is carried out.

lower the probability of heads when the 32nd bit is set. This a good idea since after the first feature was added, roughly one third of the time the distribution is still uniform on heads and tails. Since $\left(\frac{9}{10}\right)^{32} \approx 0.034$, this bit is not well correlated with the coin, and reducing the probability of heads when it is set improves the model. The third feature, which queries the 35th bit, similarly lowers the probability of heads. At this point the probability of heads has been pushed too far down, due to the overlap of events where the first, 32nd, and 35th bits are set. To compensate for this, the model chooses to query the second bit, and thereby reestablish the proper distribution on heads.

Similar effects appear in the feature selection results for the segmentation problem in the following sections. For the details on feature selection and examples of it in action, we refer to the papers by Berger and Della Pietra et al. [2, 5]. The latter also explains how the feature selection algorithm generalizes decision trees. While decision trees recursively partition the training data, the features in an exponential model can be overlapping, so that the scheme is much less prone to overfitting. This is an important distinction when drawing inferences from text, where the sparse data problem is typically so severe. Our experiments with segmentation bear this out since good results are obtained after only a handful of features are found, with no fussing over the issues of stopping, pruning, or smoothing. While these issues are certainly relevant to our approach using exponential models, they are not primary considerations in obtaining useful models.

4. Language Models

A *language model* is a conditional distribution $p(w_i \mid w_0 w_1 \ldots w_{i-1})$ on the identity of the i-th word in a sequence, given the identities of all previous words. Central to our approach to segmenting are two different language models, a *short-range* and a *long-range* model. Monitoring the relative behavior of these two models goes a long way towards helping our model sniff out natural breaks in the text, and so we devote this section to a brief review of language modeling.

4.1. A short-range model of language

A *trigram model* approximates language as a second-order Markov process, making the assumption that $p(w_i \mid w_0 w_1 \ldots w_{i-1}) \approx p(w_i \mid w_{i-2} w_{i-1})$. Typically, one computes the parameters of a trigram model using a modified maximum-likelihood approach, such as that described in Katz [13]. For the purposes of this study, we constructed two different trigram models. The parameters of the first, especially suited for financial newswire text, were tuned to approximately 38 million words of archived Wall Street Journal (henceforth WSJ) articles. The second model was trained on a collection of broadcast news transcripts (BN) containing about 150 million words. (Details on the text corpora employed in this study appear at the end of the paper.) In either case, the corpus served to define a "known word" set $\mathcal{W}$, which contained the top several thousand most frequently occurring words in the corpus. For the purposes of language modeling, all words outside $\mathcal{W}$ were considered as a single "unknown" word.

The assumption that English is well approximated by a second-order Markov process is highly dubious. However, although words prior to w_{i-2} certainly bear on the identity of w_i, higher-order models are impractical: the number of parameters in an n-gram model is $O(\mid \mathcal{W} \mid^n)$, and finding the resources to compute and store all these parameters is a daunting task for $n > 3$. Usually the lexical myopia of the trigram model is a hindrance; however, we will see how we can actually exploit this shortsightedness for the segmentation task.

4.2. A long-range model of language

One of the fundamental characteristics of language, viewed as a stochastic process, is that it is highly *nonstationary*. Throughout a written document and during the course of spoken conversation, the topic evolves, affecting local statistics on word occurrences. A model that can adapt to its recent context would seem to offer much over a stationary model such as a trigram model. For example, an adaptive model might, for some period of time after seeing the word HOMERUN, boost the probabilities of the set of words {HOMERUN, PITCHER, FIELDER, ERROR, BATTER, TRIPLE, OUT}. To illustrate the point, we provide an excerpt from the BN corpus.

Underlined words mark where a long-range language model might reasonably be expected to outperform (i.e., assign higher probabilities than) a short-range model:

> SOME DOCTORS ARE MORE SKILLED AT DOING THE PROCEDURE THAN OTHERS SO IT'S RECOMMENDED THAT PATIENTS ASK DOCTORS ABOUT THEIR TRACK RECORD. PEOPLE AT HIGH RISK OF STROKE INCLUDE THOSE OVER AGE 55 WITH A FAMILY HISTORY OF HIGH BLOOD PRESSURE OR DIABETES, AND SMOKERS. WE URGE THEM TO BE EVALUATED BY THEIR FAMILY PHYSICIANS AND THIS CAN BE DONE BY A VERY SIMPLE PROCEDURE SIMPLY BY HAVING THEM TEST WITH A STETHOSCOPE FOR SYMPTOMS OF BLOCKAGE.

One means of injecting long-range awareness into a language model is by retaining a cache of the most recently seen n-grams which is combined (typically by linear interpolation) with the static model [12, 15]. Another approach, using maximum entropy methods, introduces parameters for *trigger pairs* of mutually informative words, so that occurrences of certain words in recent context boost the probabilities of the words that they trigger [17].

The method we use here, described in Beeferman et al. [1], starts with a trigram model as a *prior*, or default distribution, and tacks onto the model a set of features to account for the long-range lexical properties of language. The features are trigger pairs, automatically discovered by analyzing a corpus of text using a mutual information heuristic described in Rosenfeld [22]. Figure 2 contains a sample of the (s, t) trigger pairs used in the BN long-range model. A five million word subset of the BN corpus served to create the long-range component of the BN model; a one-million word subset of the WSJ corpus was mined to create the WSJ long-range model.

To incorporate triggers into a trigram language model, we build a family of conditional exponential models of the general form

$$p_{\rm exp}(w \mid X) = \frac{1}{Z_\lambda(X)} e^{\lambda \cdot f(w,X)}\, p_{\rm tri}(w \mid w_{-2}, w_{-1})$$

where $X \equiv w_{-N}, w_{-N+1}, \ldots, w_{-1}$ is the *history* (i.e., the N words preceding w in the text), and $Z_\lambda(X)$ is the normalization constant

$$Z_\lambda(X) = \sum_{w \in \mathcal{W}} e^{\lambda \cdot f(w,X)}\, p_{\rm tri}(w \mid w_{-2}, w_{-1})\,.$$

In the models that we built, a feature f_i is an indicator function, testing for the occurrence of a trigger pair (s_i, t_i):

$$f_i(w, X) = \begin{cases} 1 & \text{if } s_i \in X \text{ and } w = t_i \\ 0 & \text{otherwise.} \end{cases}$$

To each trigger pair (s, t) there corresponds a real-valued parameter $\lambda_{s,t}$; the probability of t is boosted by a factor of approximately $e^{\lambda_{s,t}}$ for N words following the occurrence of s. The training algorithm we use for estimating these parameters is the same improved iterative scaling algorithm used to train our exponential segmentation models, as described in Section 3.

s, t	$e^{\lambda_{s,t}}$
RESIDUES, CARCINOGENS	2.3
CHARLESTON, SHIPYARDS	4.0
MICROSCOPIC, CUTICLE	4.1
DEFENSE, DEFENSE	8.4
TAX, TAX	10.5
KURDS, ANKARA	14.8
VLADIMIR, GENNADY	19.6
STEVE, STEVE	20.7
EDUCATION, EDUCATION	22.2
INSURANCE, INSURANCE	23.0
PULITZER, PRIZEWINNING	23.6
YELTSIN, YELTSIN	23.7
SAUCE, TEASPOON	27.1
FLOWER, PETALS	32.3
PICKET, SCAB	103.1

Figure 2. A sample of the 59,936 word pairs from the BN domain. Roughly speaking, after seeing the word s, the empirical probability of witnessing the corresponding word t in the next N words is $e^{\lambda_{s,t}}$ more likely than otherwise. In the experiments described herein, $N = 500$.

For a concrete example, if s_i =VLADIMIR and t_i =GENNADY, then $f_i = 1$ if and only if VLADIMIR appeared in the past N words and the current word w is GENNADY. Consulting Figure 2, we see that in the BN corpus, the presence of VLADIMIR will (roughly speaking) boost the probability of GENNADY by a factor of 19.6 for the next 500 words.

Using the model—that is, calculating $p_{\text{exp}}(w \mid X)$—is a three-step process:

1. Start with the probability p_{tri} assigned by the trigram model;

2. Multiply this probability by the boosting factor $e^{\lambda_{s,t}}$ corresponding to each "active" trigger pair: that is, each (s, t) for which s appeared in X and $t = w$;

3. Divide by the normalizing term $Z_\lambda(X)$.

One propitious manner of viewing this model is to imagine that, when assigning a probability to a word w following a history X, the model consults a cache containing words which appeared in X and which are the left half of some (s, t) trigger pair. In general, the cache consists of content words s which promote the probability of their mate t, and correspondingly demote the probability of other words. We say that a pair (s, t) is a *self trigger* if $s = t$, and a *non-self trigger* otherwise. In Section 9 we investigate the contribution of each trigger pair type to the performance of our segmentation model.

5. Features for Segmenting

5.1. Topicality features

A long-range language model uses words from previous sentences to bias itself regarding the identity of the next word. This is likely to make for a more accurate model if all of the previous sentences are in the same document as the current word. In the case of the trigger model described in Section 4.2, the cache will be filled with "relevant" words.

On the other hand, if the present document has just recently begun, the long-range model is wrongly conditioning its decision on information from a different—and presumably unrelated—document. A soap commercial, for instance, doesn't provide a helpful context to a long-range model in assigning probabilities to the words in the news segment following the commercial. In fact, a long-range model will likely be misled by such irrelevant context.

So at the beginning of a document, the myopia of the trigram model actually gives it an advantage over a long-range model. But sufficiently far into a document, the long-range model will, by adapting itself to the growing context, outperform the trigram model. By monitoring the long- and short-range models, one might be more inclined towards a boundary when the long-range model suddenly shows a dip in performance—a lower assigned probability to the observed words—compared to the short-range model. Conversely, when the long-range model is consistently assigning higher probabilities to the observed words, a boundary is less likely.

This motivates the measure of *topicality* $T(w, X)$, which we define as

$$T(w, X) \equiv \log \frac{p_{\text{exp}}(w \mid X)}{p_{\text{tri}}(w \mid w_{-2}, w_{-1})}$$

When the exponential model outperforms the trigram model, $T > 0$.

Observing the behavior of T as a function of the position of the word within a segment, one discovers that on average T slowly increases from below zero to well above zero. Figure 3 gives a striking graphical illustration of this phenomenon. The figure plots the average value of T as a function of relative position in the segment, in words, with position zero indicating the beginning of a segment. This plot shows that when a segment boundary is crossed (where the horizontal axis is labeled 0), the predictions of the adaptive model undergo a dramatic and sudden degradation, and then steadily become more accurate as relevant content words for the new segment are encountered and added to the cache.

This observed behavior is consistent with our intuition: the cache of the long-range model is unhelpful early in a document, when the new content words bear little in common with the content words from the previous article. Gradually, as the cache fills with words drawn from the current article, the long-range model gains steam and T increases. While Figure 3 shows that this behavior is very pronounced when averaged over many trials, our feature selection results indicate that topicality is also a very good predictor of boundaries for individual events.

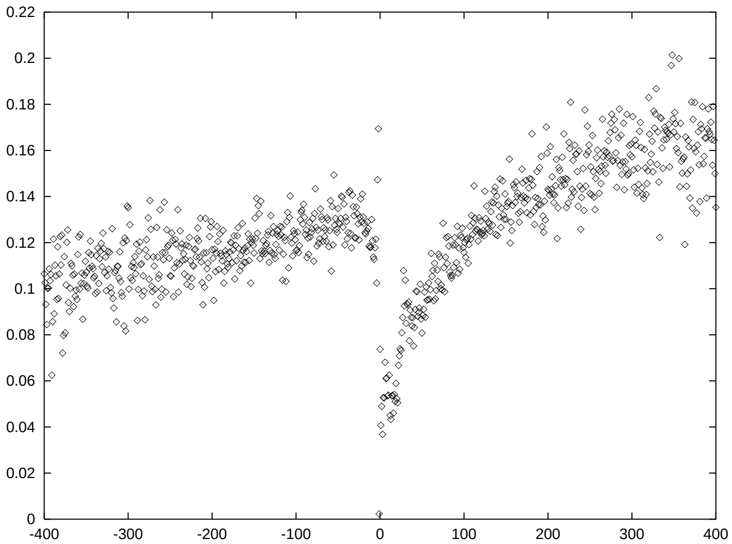


Figure 3. The average ratio of the logarithm of the adaptive language model to the static trigram model, as a function of relative position within a segment. The data was collected on Reuters stories from the TDT corpus [6]. In this plot the position labeled 0 on the x-axis corresponds to a boundary and the position labeled 100 (-100) corresponds to 100 words after (before) the beginning of a segment. It appears that the behavior of this simple ratio is highly correlated with the presence of boundaries.

The working assumption for the experiments reported in this paper is that sentence boundaries are provided, and so the system only concerns itself with the topicality score assigned to entire sentences normalized by sentence length, i.e., a geometric mean of language model ratios. To determine a set of candidate features, we fix a collection of intervals $[a, b]$ in advance and allow all candidate topicality features which ask whether the ratio T lies in a given interval.

5.2. Cue-word features

In certain domains, selected words can often act as cues, indicating the presence of a nearby boundary. In the BN domain, for example, we have observed that the word JOINS is evidence that a segment boundary has recently occurred. Many other "cue words" exist, not only in the BN domain, but in WSJ and other domains as well—though the cue words are different for different domains.

This motivates our inclusion of "cue-word features." For each word in the language model's vocabulary, we pose several questions as candidate features:

- *Does the word appear in the next few sentences?*

- *Does the word appear in the next few words?*

- *Does the word appear in the previous few sentences?*
- *Does the word appear in the previous few words?*
- *Does the word appear in the previous few sentences but not in the next few sentences?*
- *Does the word begin the preceding sentence?*

In posing these questions, we need not restrict ourselves to a single definition of *few*. To ensure that we ask the right question, we choose to ask more. Each question above is parameterized by not only by a vocabulary word, but also by a distance that ranges between one and ten in our experiments.

Having concluded our discussion of our overall approach, we present in Figure 4 a schematic view of the steps involved in building a segmenter using this approach.

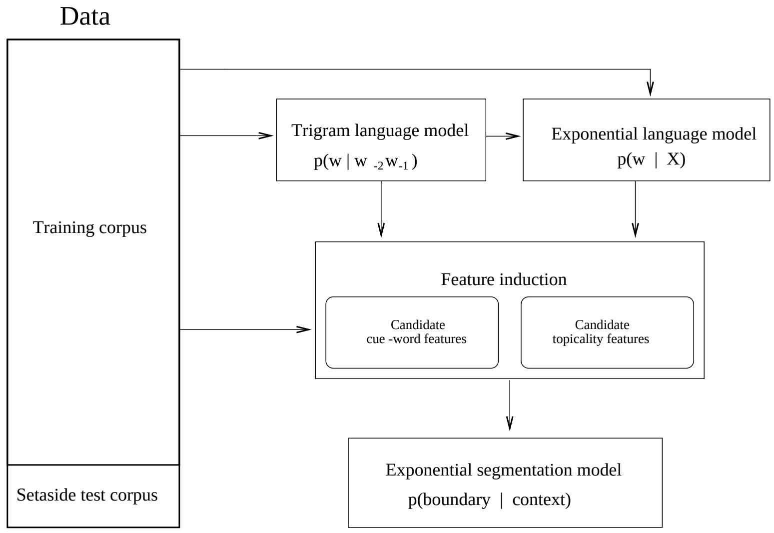


Figure 4. Data flow in training the segmentation model. Sentences from a large corpus of text serve to train both a short-range and a long-range statistical language model. The feature selection process makes use of these two models and of the training corpus itself to construct a set of the "best" features, which are combined into an exponential model of segmentation.

6. Feature Selection in Action

This section provides a peek at the construction of segmenters for two different domains. Inspecting the sequence of features selected by the selection algorithm reveals much about feature selection in general, and how it applies to the segmenting task in particular. The first segmenter was built on the WSJ corpus. The second was built on the CNN portion of the Topic Detection and Tracking (TDT) Corpus.

6.1. WSJ *features*

For the WSJ experiments, a total of 300,000 candidate features were available to the selection program. Figure 5 shows the first several selected features. The word or topicality score for each feature is shown together with the value of e^{λ} for the feature after iterative scaling is complete for the final model. The $\vdash\!\leftarrow \;\; \rightarrow\!\dashv$ figures indicate features that are active over a range of sentences. Thus, the symbol $\vdash\!\longleftarrow \underset{0.07}{\text{MR.}} \longrightarrow\!\dashv^{+1}$ represents the feature *Does the word* MR. *appear in the next sentence?* which, if true, contributes a factor of $e^{\lambda} = 0.07$ to the exponential model. The symbol $\vdash\!\longleftarrow \underset{0.07}{a < T_i < b} \longrightarrow\!\dashv^{+5}$ asks if the topicality statistic is in the interval (a, b) over the next five sentences. Similarly, the $\bullet\!\leftarrow \;\; \rightarrow\!\bullet$ figures represent features that are active over a range of *words*. For example, the figure $\bullet\!\longleftarrow \underset{0.08}{\text{HE}} \longrightarrow\!\bullet^{+5}$ represents the question *Does the word* HE *appear in the next five words?* which is assigned a weight of 0.08. The symbol $^{-5}\vdash\!\longleftarrow \text{SAID} \longrightarrow\!\dashv\!\vdash\!\longleftarrow \underset{2.7}{\neg\,\text{SAID}} \longrightarrow\!\dashv^{+5}$ stands for a feature which asks *Does the word* SAID *appear in the previous five sentences but not in the next five sentences?* and contributes a factor of 2.7 if the answer is *yes.*

Most of the features in Figure 5 make a good deal of sense. The first selected feature, for instance, is a strong hint that an article may have just begun; articles in the WSJ corpus often concern companies, and typically the full name of the company (ACME INCORPORATED, for instance) only appears once at the beginning of the article, and subsequently in abbreviated form (ACME). Thus the appearance of the word INCORPORATED is a strong indication that a new article may have recently begun.

The second feature uses the topicality statistic. If the trigger model performs poorly relative to the trigram model in the following sentence, this feature boosts the probability of a segment boundary at this location by a factor of 5.3, roughly speaking.

The fifth feature concerns the presence of the word MR. In hindsight, we can explain this feature by noting that in WSJ data the style is to introduce a person in the beginning of an article by writing, for example, WILE E. COYOTE, PRESIDENT OF ACME INCORPORATED... and then later in the article using a shortened form of the name, e.g., MR. COYOTE CITED A LACK OF EXPLOSIVES... Thus, the presence of MR. in the following sentence *discounts* the probability of an article boundary by 0.07, or by a factor of roughly 14.

The sixth feature—which boosts the probability of a segment if the previous sentence contained the word CLOSED—is another artifact of the WSJ domain, where articles often end with a statement of a company's performance on the stock market during the day of the story of interest. Similarly, the end of an article often contains an invitation to visit a related story; hence a sentence beginning with SEE boosts the probability of a segment boundary by the large factor 94.8. Since a personal pronoun typically requires an antecedent, the presence of HE among the first words is a sign that the current position is *not* near an article boundary, and this feature therefore has a discounting factor of 0.082.

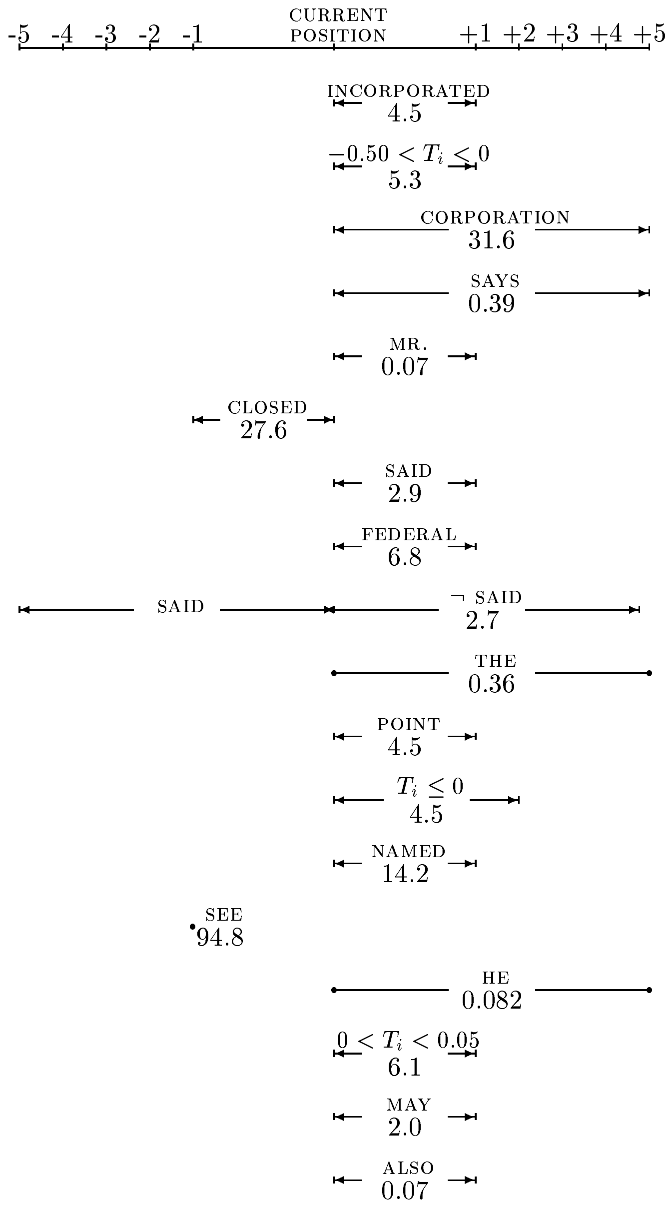


Figure 5. First several features induced for the WSJ corpus, presented in order of selection, with e^{λ} factors underneath. The length of the bars indicate active range of the feature, in words or sentences, relative to the current word.

6.2. Broadcast news features

For the CNN experiments, a larger vocabulary and roughly 800,000 candidate features were available to the program.

Figure 6 reveals the first several features chosen by the algorithm. The word C. appears in several of the first features. This is occurs because the data is tokenized for speech processing (whence C. N. N. rather than CNN), and the network identification information is often given at the end and beginning of news segments (e.g., C. N. N.'S RICHARD BLYSTONE IS HERE TO TELL US...). The first feature asks if the letter C. appears in the previous five words; if so, the probability of a segment boundary is boosted by a factor of 9.0. The personal pronoun I appears as the second feature; if this word appears in the following three sentences then the probability of a segment boundary is discounted.

The language model topicality statistic appears for the first time in the sixth feature. The word J. appearing in the seventh and fifteenth features arises from the the large number of news stories relating to O.J. Simpson. The nineteenth feature asks if the term FROM appears among the previous five words, and if the answer is "yes" raises the probability of a segment boundary by more than a factor of two. This feature makes sense in light of the sign-off conventions that news reporters and anchors follow (THIS IS WOLF BLITZER REPORTING LIVE FROM THE WHITE HOUSE). Many of the remaining features in Figure 6 have equally straightforward explanations.

7. Efficient Learning

A shortcoming of the feature selection approach is that it requires patience—constructing the models of Section 6 took over 24 hours to run on a 248 MHz Sun UltraSPARC II workstation. We now describe two efficiency measures for speeding the process of model construction.

7.1. Inducing features in batches

The feature selection summarized in Algorithm 1 grows the model by a single feature at each iteration. It is natural to consider a modified algorithm which adds several features at a time. That is, in step 3 of Algorithm 1, select the $B > 1$ features $\{g_1, g_2, \ldots g_B\}$ with maximal gain G_q.

The problem with this method is that the top-ranked features may be highly correlated with one another. Another way of saying this is that while the top ranked features each individually offer a significant gain over the current model, the gains are not necessarily additive. To illustrate, Figure 7 shows the 21 highest-gain features in the first iteration of Algorithm 1 on a $200,000$-word sample of BN text. Clearly, the list exhibits considerable redundancy, and a model of boundaries in BN data needn't include every feature on this list.

We call a set of features with non-additive gain *overlapping*. A model containing overlapping features is not only aesthetically unpleasant, but also undesirable in

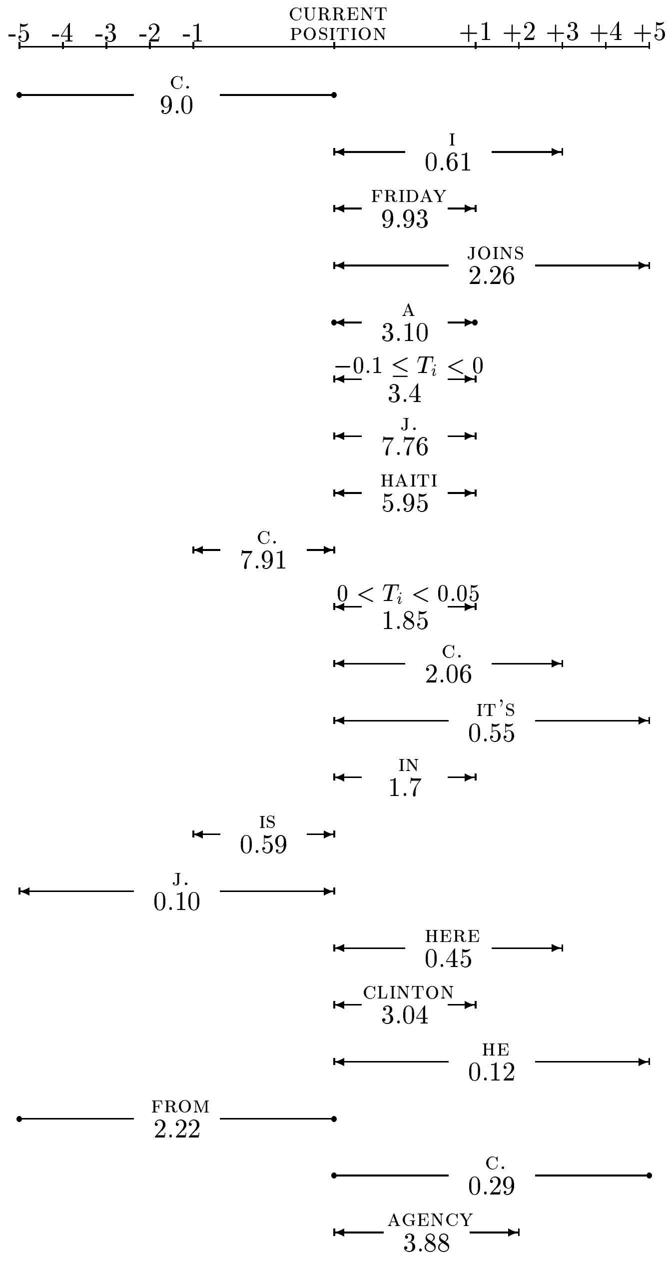


Figure 6. First several features induced for the CNN portion of the TDT corpus, presented in order of selection, with e^{λ} factors underneath.

feature	**gain** G_{q_0}
`N.` appears up to five words in past	0.0438
`C.` appears up to five words in past	0.0359
`C.` appears up to one sentences in past	0.0352
`N.` appears up to one sentences in past	0.0337
`C.` appears up to two sentences in past	0.0239
`N.` appears up to two sentences in past	0.0224
`C.` appears up to three sentences in past	0.0183
`N.` appears up to three sentences in past	0.0170
`C.` appears up to five sentences in future	0.0130
`N.` appears up to five sentences in future	0.0123
`C.` appears up to five sentences in past	0.0115
`N.` appears up to five sentences in past	0.0108
`LIVE` appears up to one sentences in past	0.0105
one or more unknown words appear within 40 words on both sides	0.0099
`C.` appears up to three sentences in future	0.0099
`LIVE` appears up to five words in past	0.0098
`N.` appears up to three sentences in future	0.0097
one or more unknown words appear within 20 words on both sides	0.0086
`LIVE` appears up to two sentences in past	0.0085
`THE` appears up to one sentences in past	0.0077
`LIVE` appears up to three sentences in past	0.0076

Figure 7. The 21 highest-gain features in a selected iteration of Algorithm 1 on a 200,000-word sample of BN text. There exists a high degree of redundancy among these features. The first eight, for example, are essentially synonymous: each encodes, in a slightly different manner, the fact that the words "C. N. N." is a harbinger of an imminent boundary.

practice, since it contains useless features which need to be evaluated in order to make predictions with the model. In a worst-case scenario, adding B features at a time might lead to a model no better (in a maximum-likelihood sense) than adding one feature at a time. The goal, then, is to induce multiple, non-overlapping features at each iteration.

The approach we take, outlined in Algorithm 2, is to select a batch of multiple features, sift a group of non-overlapping features from the batch, and then only add the smaller group to the model. All that remains unspecified in Algorithm 2 is the specifics of the sifting procedure: how to select a set of non-overlapping features from a batch of B features?

One can view each feature $f(X, b)$ as a binary random variable which takes on the value zero or one at any position in the training corpus. We denote the ith position in the training corpus by (X_i, b_i). A standard statistical test for independence (or lack thereof) between two random variables is the *correlation coefficient*,

Algorithm 2: Efficient Feature Selection for Exponential Models

Input: Collection of candidate features $\mathcal{C}$, training samples $\{X_i, b_i\}$, batch size B, and desired model size n.

Output: Selected features $f_1 \ldots f_n$ and their maximum-likelihood parameters $\lambda_1 \ldots \lambda_n$.

1. Set $i \leftarrow 1$ and let $q^{(0)}$ be uniform.
2. For each candidate feature $g \in \mathcal{C}$, compute the gain $G_{q^{(i-1)}}(g)$ of g.

3a. Select the B features $g \equiv \{g_1, g_2 \ldots g_B\}$ yielding the largest gain.

3b. Sift a set of non-overlapping features $g' \equiv \{g'_1, g'_2 \ldots g'_{B'}\}$ from g.

3c. Set $f_i \leftarrow g'_1$, $f_{i+1} \leftarrow g'_2$, $\ldots f_{i+B'-1} \leftarrow g'_{B'}$.

4. Compute $q^\star = \arg\max_{q \in \mathcal{Q}(f)} L(q)$ to obtain weights $\lambda_1, \lambda_2 \ldots \lambda_{i+B'-1}$, using improved iterative scaling.
5. Set $q^{(i+B'-1)} \leftarrow q^\star$.
6. If $i \geq n$ then exit.
7. Set $i \leftarrow i + B'$ and go to step 2.

$$\rho(f_i, f_j) \equiv \frac{Var(f_i, f_j)}{\sqrt{Var(f_i) Var(f_j)}}$$

where $Var(f)$ is the variance of the bit vector $\{f(X_1, b_1), f(X_2, b_2) \ldots f(X_N, b_N)\}$ representing the evaluation of feature f on each of the N positions in the training corpus, and $Var(f, g)$ is the covariance of the bit vectors corresponding to features f and g.

Our implementation of Algorithm 2 visits the members of the selected batch of features $g \equiv \{g_1, g_2 \ldots g_B\}$ in order, discarding g_i if $\rho(g_i, g_j) > \delta$ for some $j < i$. The value δ was set by trial and error at 0.2. Figure 8 lists the set of features which survived the sifting process applied to the features in Figure 7, and Figure 9 summarizes the computation time (in minutes) required by Algorithm 2 with various settings of B, stopping after the model contained at least 100 features. The error metric P_k will be introduced in Section 8 as a measure of the quality of the model (smaller is better).

(Note from Figure 9 that $B > 1$ sometimes actually leads to a *better*-performing model; this paradoxical situation arises because when adding features in batches, the training algorithm can't stop after adding exactly s features, and might generate a model with more features.)

7.2. Targeted events sampling

For the corpus used as training data in the experiments we report in Section 9, the marginal probability of a topic break was roughly 1/30. (Equivalently, the average document length was about 30 sentences.) Selecting events uniformly at random from the training corpus, then, one would expect positive examples of topic breaks to comprise about 1/30 of the training examples. For instance, a 200,000-word sample containing 11,303 training events harbored only 322 positive examples.

The statistical technique of *importance sampling* [7] typically used in performing Monte Carlo estimation of an integral, suggests an optimization: bias our event sampling to include more positive examples so that the feature selection algorithm has a better chance to learn the features which suggest the presence of a topic boundary. Since the complexity of the algorithm is linear in the total number of events, we might hope to reduce the training time of the model without compromising the resulting model by extracting fewer events overall but with the same number of positive examples. Figure 10 shows the performance of models trained using a fixed number of total events, but varying proportions of negative events.

8. A Probabilistic Error Metric

Precision and recall statistics are a popular means of assessing the quality of classification algorithms. For the segmentation task, recall measures the fraction of actual boundaries which an automatic segmenter correctly identifies, and precision measures the fraction of boundaries identified by an automatic segmenter which are actual boundaries. In this section we point out some shortcomings of the precision/recall metrics and propose a new approach to gauging the quality of an automatic boundary detector.

In almost any conceivable application, a segmenting tool that consistently comes close—off by a sentence, say—is preferable to one that places boundaries willy-

feature	**gain** G_{q_0}
`N.` appears up to five words in past	0.0438
one or more unknown words appear within 40 words on both sides	0.0099
`THE` appears up to one sentence in past	0.0077

Figure 8. Of the 21 features listed in the previous figure, the sifting procedure eliminated all but these three apparently uncorrelated features.

Batch size B	*Time to grow model*	P_k
1	1071	13.5%
10	395	13.2%
50	233	12.3%
100	209	13.4%

Figure 9. Increasing the size of the batches leads to a faster feature selection algorithm, with negligible change in the error rate. P_k denotes the probabilistic error metric introduced in Section 8. These numbers reflect training time (in minutes on a 248 MHz Sun UltraSPARC II) on a 200,000-word corpus until the model contained at least 100 features.

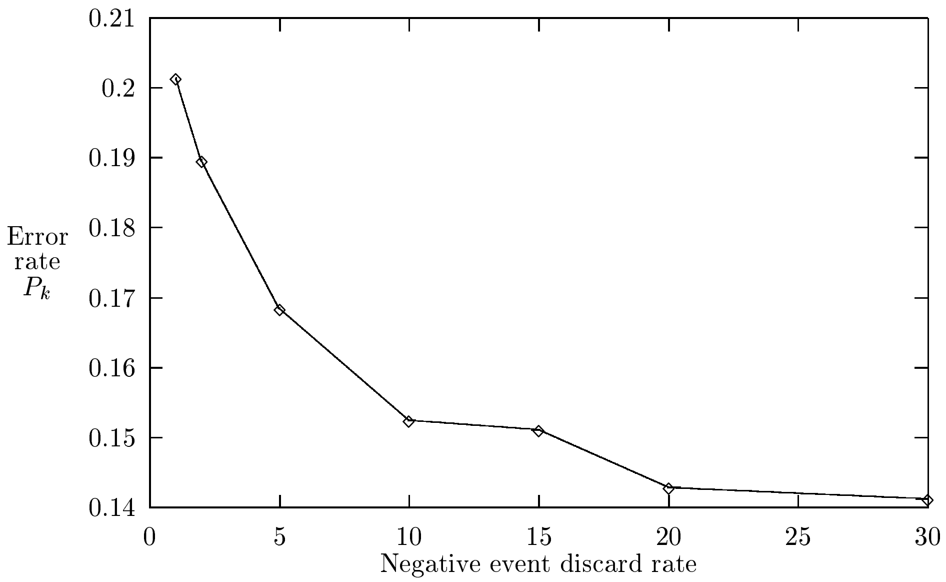


Figure 10. Performance of the 100-feature segmentation model on 1 million words of heldout CNN broadcast news data, as a function of the negative event sampling rate. Each point represents the performance of a model trained with a fixed number of total events (sentences)—only 5000—at a different ratio of negative to positive events.

nilly. Yet an algorithm that places a boundary a sentence away from the actual boundary every time actually receives *lower* precision and recall scores than an algorithm that hypothesizes a boundary at every position. It is natural to expect that in a segmenter, close should count for something. One suggestion [21] is to redefine correct to mean "hypothesized within some constant-sized window of units away from a reference boundary," but this approach seems too forgiving; after all, a "right on the nose" segmenter should outscore an "always close" segmenter.

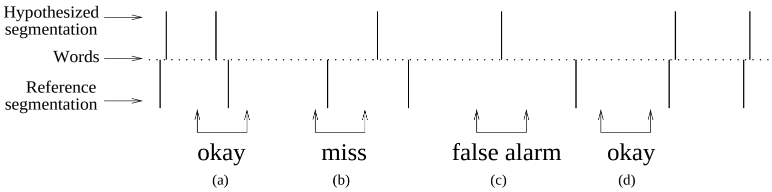


Figure 11. Failure modes of a segmentation decision procedure. The lower vertical lines represent "true" segment breaks, and the upper vertical lines represent hypothesized breaks. A fixed-width window slid across the corpus yields one of the following outcomes at each step: *acceptable* (a and d), in which a hypothesized break and a true break are both present or both absent within the window; *false negative* (b), in which a true break is present but not a hypothesized break; and *false alarm* (c), in which case a hypothesized break is present but not a true break.

Precision and recall have a complementary nature in most applications. Hypothesizing more boundaries raises recall at the expense of precision; an algorithm designer can tweak parameters to trade between the two in a way that matches the demands of the application. One compromise between precision and recall is the *F-measure*, a weighted combination of the two, but this is difficult to interpret as a meaningful performance measure.

8.1. *Co-occurrence agreement probability*

The error metric introduced here formalizes the notion that one segmenter is better than another if it is better able to identify when two sentences belong to the same document and when they do not.

As we have defined the task, a segmenter identifies boundaries between successive sentences in a corpus of text. A natural way to reason about developing a segmentation algorithm is therefore to optimize the likelihood that two sentences are correctly labeled as being related or being unrelated. Consider the error metric P_D that is simply the probability that two sentences drawn randomly from the corpus are correctly identified (as belonging to the same document or to different documents). More formally, given two segmentations `ref` and `hyp` for a corpus n sentences long,

$$P_D(\texttt{ref},\texttt{hyp}) = \sum_{1\leq i\leq j\leq n} D(i,j)\ \left(\delta_{\texttt{ref}}(i,j)\ \overline{\oplus}\ \delta_{\texttt{hyp}}(i,j)\right)$$

Here $\delta_{\texttt{ref}}$ is an indicator function which evaluates to one if the two corpus indices specified by its parameters belong in the same document, and zero otherwise. Similarly, $\delta_{\texttt{hyp}}$ is one if the two indices are hypothesized to belong in the same document, and zero otherwise. The $\overline{\oplus}$ operator is the `XNOR` function ("both or neither") on its two operands. The function D is a *distance probability distribution* over the set of possible distances between sentences chosen randomly from the corpus, and will in general depend on certain parameters such as the average spacing between documents.

There are several plausible distributions one could use for D. If D is uniform over the length of the text, then the metric represents the probability that any two sentences drawn from the corpus are correctly identified as being in the same document or not. In practice this yields a too-forgiving metric; for large corpora, most randomly drawn pairs of sentences are very far apart and even the most naive segmenter is likely to identify them as belonging to different documents. A more reasonable choice for D, which focuses the probability mass on small distances, is $D = E_{\mu}$, an exponential distribution with mean μ^{-1}, fixed at the mean document length for the domain.

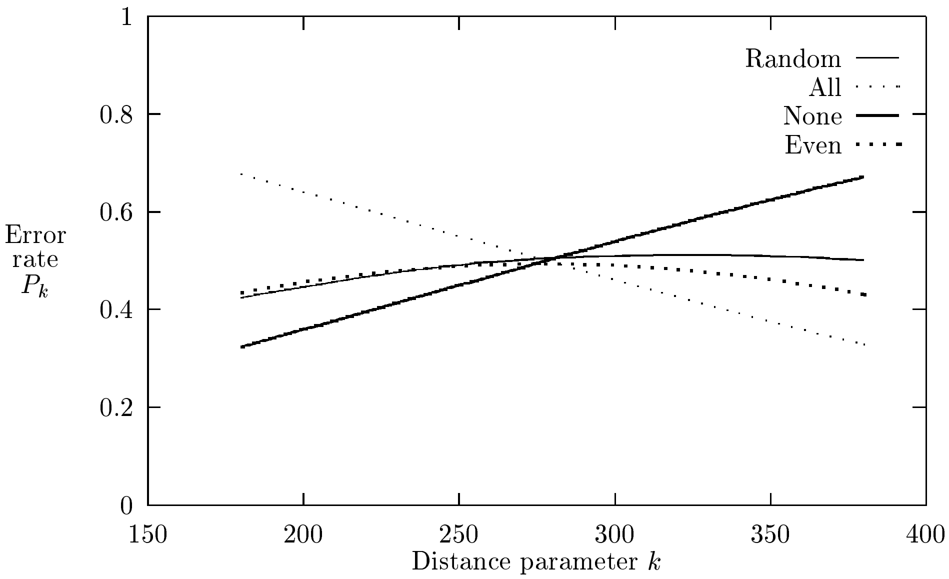


Figure 12. Performance of some degenerate algorithms on the segmentation of CNN broadcast news data, shown as functions of the "inter-probe distance" parameter k of the error metric. The *random* algorithm places segment boundaries uniformly at random, generating a number of documents equal to the number of reference segments in the (test) corpus. The *all* algorithm places a boundary after each sentence. The *none* algorithm places no boundaries. The *even* algorithm places a boundary after every mth sentence, where m is the average reference segment length.

Another, offered by Doddington [6], is to let $D = D_k$ have all its probability mass at a fixed distance k. The computation of the metric can then be visualized as two probes, a fixed distance apart, sweeping across the corpus (Figure 11). It turns out empirically, and can be shown analytically (under strong assumptions), that if the window k is chosen to be half the average reference segment length (in words), then all of the major "degenerate" algorithms—hypothesizing boundaries everywhere, uniformly, randomly, and not at all—have nearly the same low score of $P_k \equiv P_{D_k} \approx \frac{1}{2}$ (Figure 12). With this justification, we use the error metric P_k in our quantitative analysis.

This measure is a probability and therefore a real number between zero and one. An algorithm scores one with respect to some text if and only if it exactly predicts the location of the boundaries in the text. The metric captures the notion of nearness in a principled way, gently penalizing algorithms that hypothesize boundaries that aren't quite right, and scaling down with the algorithm's degradation. Furthermore, it is not possible to obtain a high score by "cheating" with a degenerate model, such as the *all* or *none* algorithms. We refer to Section 9 for sample results on how these trivial algorithms score.

The number $P_k(\texttt{ref}, \texttt{hyp})$ is the probability that a randomly chosen pair of words a distance of k words apart is inconsistently classified; that is, for one of the segmentations the pair lies in the same segment, while for the other the pair spans a segment boundary. This probability can be decomposed into two conditional probabilities, called the *miss* and *false alarm probabilities*:

$$\begin{aligned} &p(\text{error} \mid \texttt{ref}, \texttt{hyp}, k) = \\ &\quad p(\text{miss} \mid \texttt{ref}, \texttt{hyp}, \text{different } \texttt{ref} \text{ segments}, k)\, p(\text{different } \texttt{ref} \text{ segments} \mid \texttt{ref}, k) \\ &+ p(\text{false alarm} \mid \texttt{ref}, \texttt{hyp}, \text{same } \texttt{ref} \text{ segment}, k)\, p(\text{same } \texttt{ref} \text{ segment} \mid \texttt{ref}, k) \end{aligned}$$

The miss and false alarm probabilities give a more detailed look at the error, allowing an assessment in terms of precision and recall.

9. Experimental Results

This section presents the results of applying the feature selection algorithm discussed in the earlier sections to segment CNN broadcast news data and Wall Street Journal text. (See the end of the paper for a more detailed description of the data we used for training and testing our models.) These results are compared to those obtained using decision tree methods, and we evaluate the relative contributions made by the cue-word and topicality features. In order to give the reader an intuitive feel for the performance of these algorithms, we also present qualitative results by displaying graphs of the segmentations on test data.

9.1. Quantitative results

In Section 3 we divided the segmentation task into a modeling problem—constructing a model $q(b \mid X)$—and a decision problem—using the model to assign segment boundaries to a stream of data. The decision procedure we employ is straightforward: hypothesize a segment boundary at each position for which 1) $q(\text{YES} \mid X) > \alpha$ and 2) no higher scoring position occurs within $\pm\epsilon$ positions, where α and ϵ are fixed constants. The minimum separation ϵ was set to six sentences for CNN data, and two sentences for WSJ data. The error probability P_k is evaluated by fixing k to be half of the average reference segment length. The model threshold α is then determined on heldout data by requiring that the probability of a hypothesized boundary falling within a window of k words is equal to the probability of a reference boundary falling in the window. In other words, the threshold is set so that the number of segments hypothesized is approximately equal to the number of segments appearing in the reference set. The threshold is *not* chosen to minimize the error rate P_k. Of course, a given application may require trading off recall for precision, or vice-versa, which may motivate a different choice of thresholds.

Two sets of experiments are reported here on broadcast news data. The first set of experiments was carried out in the context of the TDT pilot study, using the CNN portion of the corpus specifically prepared for this study, and the second using CNN data in the broadcast news corpus. One of the main differences between these corpora, for our purposes, is that the average document length of the TDT broadcast news data is nearly 400 words smaller than that in the broadcast news corpus, since long documents were excluded from the TDT corpus when it was constructed.

9.1.1. Comparison to decison trees In order to compare our use of exponential models to a more common statistical learning algorithm, we grew a decision tree on each data set, using a candidate feature set that was identical to that available to the exponential model. We adopt the CART approach for inducing decision trees [3], using entropy as the impurity function to evaluate questions. The impurity of a tree $I(T)$ is evaluated as

$$I(T) = \sum_{t \in T} I(t) = \sum_{t} p(t)\, \phi(p(\cdot \,|\, t))$$

where the sum is over all leaves of the tree, and the entropy impurity function ϕ is taken to be

$$\phi(p(\cdot \,|\, t)) = -p(\text{YES} \,|\, t) \log p(\text{YES} \,|\, t) - p(\text{NO} \,|\, t) \log p(\text{NO} \,|\, t)$$

The change in impurity resulting from asking a question q at a node t is calculated by summing the impurities of the resulting children t_L and t_R:

$$\Delta I(q, t) = I(t) - I(t_L) - I(t_R)$$

In the usual decision tree terminology, our topicality features are ordered variables, and our cue-word features are categorical variables. In the two-class case, various efficient algorithms for optimal question selection of categorical variables are known, ([3], §4.2), however we do not make use of these, and only employ the most basic decision tree methodology.

Using the entropy loss function as a splitting criterion, a decision tree with 609 nodes was grown on two million words of CNN transcripts. This tree was then "smoothed," rather than pruned, in the following manner. The empirical distribution $p(b \,|\, n)$ at node n was used to estimate a smoothed distribution $\tilde{p}(b \,|\, n)$ as

$$\tilde{p}(b \,|\, n) = \lambda(n)\, p(b \,|\, n) + (1 - \lambda(n))\ \tilde{p}(b \,|\, \mathit{parent}(n))$$

where $\mathit{parent}(n)$ is the parent node of n, and $0 \leq \lambda(n) \leq 1$ is an interpolation weight. The resulting model is naturally thought of as being comprised of a series of

Hidden Markov Models, one for each path from root to leaf, with shared parameters. These parameters are trained using the EM algorithm on heldout data [16]. Our experience has been that this smoothing procedure compares favorably to CART pruning algorithms [3] that are used to reduce the size of the tree, and thereby improve the robustness of the distributions at the leaves.

Comparisons between decision trees and exponential models are presented in each of the following sets of experiments

segmentation model	P_k	*miss probability*	*false alarm probability*
exponential model	9.5%	12.1%	6.8%
decision tree	11.3%	16%	6.6%
interpolated (exp + dtree) models	7.8%	7.2%	8.4%
hidden Markov model	16.7%	16%	17.6%
random boundaries	49.5%	60.1%	38.9%
all boundaries	47.5%	0%	100%
no boundaries	49.7%	100%	0%
evenly spaced	50.3%	50.3%	50.3%

Figure 13. Quantitative results for segmentation of the broadcast news portion of the TDT corpus. The TDT models were trained on two million words of CNN transcripts furnished with segment boundaries, and tested on one million words from the TDT corpus. A total of 100 features were induced. The performance of a decision tree grown with exactly the same candidate feature set is also given. The tree had 609 nodes, and was smoothed using the EM algorithm as indicated in the text. A simple linear interpolation (weight $\frac{1}{2}$) of the decision tree model with the exponential model resulted in an error rate of $P_k = 0.078$, with window size $k = 289$ words, equal to half of the average segment size. The decision thresholds for the exponential and decision tree models were chosen so that the probability of a hypothesized boundary falling within a window of $k = 289$ words is roughly equal to the probability of a reference boundary falling in the window. The thresholds were thus *not* chosen to minimize the error rate P_k. The default segmentation models, described in the text, are also presented.

9.1.2. TDT experiments The quantitative results for the TDT models are collected in Figure 13. These results are part of a much more extensive study carried out by several research groups in the course of the TDT project [6, 24, 20]. The exponential model evaluated here was the result of inducing 100 features on a training corpus of two million words of CNN transcriptions, and evaluated on the CNN portion of the TDT corpus. No batch selection or event discarding was used to speed up training. The error probability for the resulting model on a test corpus of one million words was $P_k = 9.5\%$, with a miss probability of 12.1% and a false alarm probability of 6.8% (Figure 13, row 1). The performance of the segmentation model as a function of the number of features induced is presented in Figure 14.

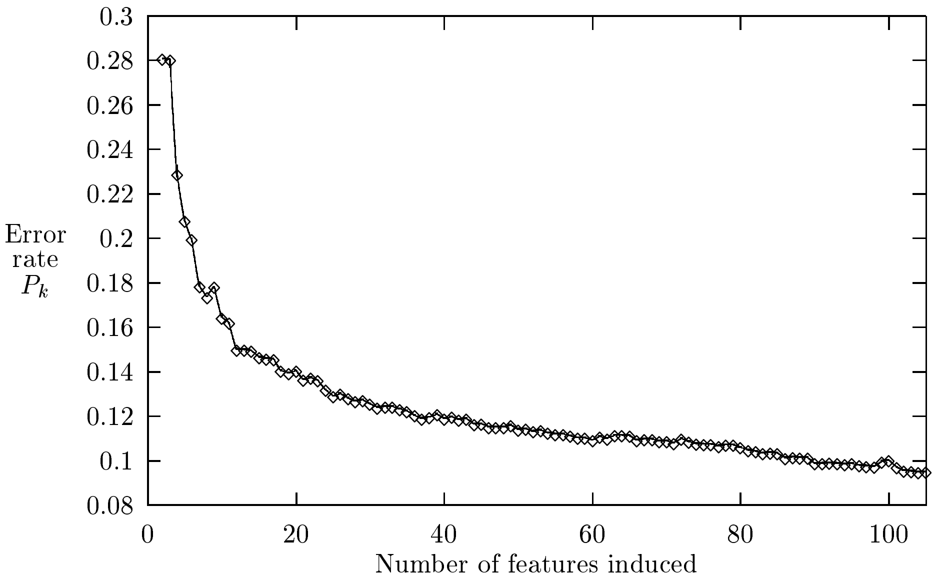


Figure 14. Performance of the segmentation model on 1 million words of heldout CNN broadcast news data, as a function of the number of features induced.

When evaluated on the one million word TDT test set, the decision tree model resulted in an error probability of $P_k = 11.3\%$, with a miss rate of 16% and a false alarm rate of 6.6% (Figure 13, row 2). To explore the possibility that the decision tree and exponential models learned different aspects of the training set, we interpolated the two models together with a fixed interpolation weight of $\frac{1}{2}$. After a threshold was set on heldout data, the resulting mixture model segmented the test data with an error probability of $P_k = 7.8\%$, which was a drop of 1.7% over the performance of the exponential model (Figure 13, row 3). Since the miss rate drops from 12.1% for the exponential model and 16% for the decision tree alone to 7.2% for the mixture, this result indicates that the two methods learned, at least in part, complementary aspects of the segmented training data.

In Figure 13 we also list the performance of Dragon's HMM approach, which was run on the identical test data set [24] (Figure 13, row 4). On this particular data set our approach using exponential models performed better than the HMM, with accuracies 9.5% and 16.7% respectively, but the exponential model performed worse on the portion of the TDT corpus comprised of Reuters newswire articles, where the accuracies were 15.5% and 12.3% [24].

segmentation model	P_k	*miss probability*	*false alarm probability*
exponential model	13.2%	16.0%	10.9%
decision tree	15.2%	19.3%	11.9%
interpolated (exp + dtree) models	11.8%	14.2%	9.8%
cue-word and $s = t$ trigger features	13.4%	16.9%	10.5%
cue-word and $s \neq t$ trigger features	13.6%	17.8%	10.1%
cue-word features only	18.3%	21.6%	15.5%
topicality features only	37.3%	42.1%	33.3%
TextTiling	34.6%	57.1%	18.6%

Figure 15. Quantitative results for segmentation of CNN broadcast news. The models were trained on two million words of CNN transcripts furnished with segment boundaries, and tested on one million words of previously unseen text. A 100 feature exponential model combining cue-word and topicality features had an error rate of $P_k = 0.132$, evaluated with a window size of $k = 498$ words, equal to half of the average segment size. To investigate the relative contributions of the cue-word and topicality features, four additional exponential models were trained. The first allowed topicality features derived from an adaptive language model that only used self triggers. The second allowed topicality features derived from an adaptive language model that only used non-self triggers. The third used only cue-word features, and the fourth used only topicality features (self and non-self triggers). The error rate of the TextTiling algorithm is presented for comparison, with the caveat that this approach is designed for sub-topic rather than document segmentation. The parameters of this algorithm were optimized on the test set to give the lowest possible error rate.

9.1.3. Additional broadcast news experiments An additional set of models was built on broadcast news (CNN) data in order to evaluate the relative contributions made by the different types of features. We began with an exponential model that was constructed from 100 automatically induced cue-word and topicality features, similar to the one constructed for the TDT experiments. As a result of the longer documents in the test set and the correspondingly larger window size of $k = 498$ words, the exponential model had a higher error rate of $P_k = 0.132$, with a miss rate of 16.0% and a false alarm rate of 10.9% (Figure 15, row 1). The performance of a decision tree grown with exactly the same candidate feature set was 15.2%, with a miss rate of 19.3% and a false alarm rate of 11.9% (Figure 15, row 2). A simple linear interpolation (weight $\frac{1}{2}$) of the decision tree model with the exponential model resulted in an error rate of 11.8% (Figure 15, row 3). To investigate the relative contributions of the cue-word and topicality features, four additional exponential models were trained. The first allowed topicality features derived from an adaptive language model that only used self triggers. The second allowed topicality features derived from an adaptive language model that only used non-self triggers. The third used only cue-word features, and the fourth used only topicality features (using both self and non-self triggers). The error rates of these models were 13.4%, 13.6%, 18.3% and 37.3% respectively (Figure 15, rows 4–7). Thus, we see that while the cue-word features are more powerful in this domain, they work in concert with the topicality features to make a more accurate model than either of the feature types can achieve alone. The effect of the topicality features is essentially the same when we use only self triggers or non-self triggers.

A comparison to the TextTiling approach was also made, using the "blocks" version of TextTiling [10] run with parameters (w, k, n, s) optimized on the test set data (Figure 15, row 8). Since paragraph boundaries are absent in the broadcast news data, each inter-sentence gap in the data was a potential boundary candidate. Boundaries were assigned to locations with depth scores exceeding a threshold that was optimized on the test set. We emphasize that this direct comparison with TextTiling is not intended to imply that the approaches are designed for or applicable to the same problems. Indeed, our use of cue-word phrases is well-suited to the article and story segmentation that we are carrying out in the WSJ and broadcast news domains, while TextTiling may be better suited for detecting more subtle sub-topic shifts in expository texts.

segmentation model	P_k	*miss probability*	*false alarm probability*
exponential model	19.0%	24.0%	15.75%
decision tree	24.6%	32.7%	19.4%
interpolated (exp + dtree) models	18.5%	24.7%	14.5%
cue-word features only	23.7%	35.6%	15.8%
topicality features only	35.8%	45.4%	29.6%
TextTiling	29.6%	45.7%	19.1%

Figure 16. Quantitative results for segmentation of Wall Street Journal text. The models were trained on one million words of WSJ text furnished with segment boundaries, and tested on 325,000 words of unseen text. A 100 feature exponential model combining cue-word and topicality features had an error rate of $P_k = 0.19$, evaluated with a window size of $k = 214$ words, equal to half of the average segment size.

9.1.4. Wall Street Journal experiments A similar set of experiments was carried out for the Wall Street Journal domain. For these experiments the models were trained on roughly one million words of labeled WSJ data, and tested on 325,000 words of unseen text. A 100 feature exponential model combining cue-word and topicality features had an error rate of $P_k = 0.19$, with a 24.0% miss rate and 15.75% false alarm rate, evaluated with a window size of $k = 214$ words, equal to half of the average segment size (Figure 16, row 1). The performance of a decision tree grown with exactly the same candidate feature set had an error rate of 24.6%, with a miss rate of 32.7% and a false alarm rate of 19.4% (Figure 16, row 2). A simple linear interpolation (weight $\frac{1}{5}$) of the decision tree model with the exponential model resulted in a small reduction of error rate, to 18.5% (Figure 16, row 3). To again investigate the relative contributions of the cue-word and topicality features, two additional exponential models were trained. The first allowed only cue-word features, and second used only topicality features (from language models using both self and non-self triggers). The error rates of these models were 23.7% and 35.8%, respectively (Figure 16, rows 4–5). The error rate of the TextTiling algorithm applied to this domain was $P_k = 0.29\%$, with a 45.7% miss rate and 19.1% false alarm rate (Figure 16, row 6). The parameters of this algorithm were again optimized on the test set to give the lowest possible error rate.

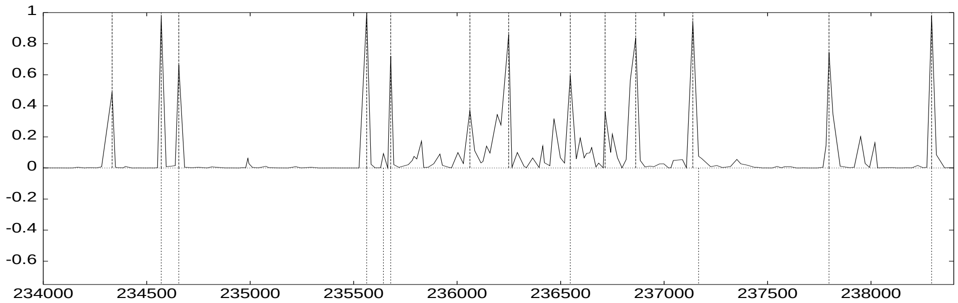
9.2. Qualitative results

We now present graphical examples of the segmentation algorithm at work on heldout data. Figure 18 shows the performance of the WSJ segmenter on a typical collection of data, in blocks of approximately $7,000$ contiguous words. In these figures the reference segmentation is shown *below* the horizontal line as a vertical line at the position between words where the article boundary actually occurred. The decision made by the automatic segmenter is shown as a vertical line *above* the horizontal line at the appropriate position. The fluctuating curve is the probability assigned by the exponential model constructed using feature selection. Notice that in this domain many of the segments are quite short, adding special difficulties for the segmentation problem.

An examination of the errors shows that many of the false positives can be explained by inconsistent labeling conventions. For example, several WSJ articles are collections of very brief summaries of unrelated news items. In such cases, the topicality features signal that a change of topic has occurred, and a boundary is hypothesized. Figure 17 shows a specific example of this.

Figure 19 shows the typical performance of the CNN segmenter on four blocks of roughly 7,000 words each. As these examples indicate, the most significant problem with the broadcast news models is the presence of false negatives where there is very little "signal" in the probability distribution, suggesting that a sufficiently rich candidate feature set is not available to the selection scheme.

We hasten to add that these results were obtained with no smoothing or pruning of any kind, and with no more than 100 features induced from the candidate set of several hundred thousand. Unlike many other machine learning methods, feature selection for exponential models is quite robust to overfitting since the features act in concert to assign probability to events through linear constraints rather than by splitting the event space and assigning probability using relative counts. We expect that significantly better results can be obtained by using cross-validation



REAGAN AIDES SAID MEESE WON'T BE CHARGED FOR HIS ROLE IN A PIPELINE PLAN. ACCORDING TO ADMINISTRATION AND LAW ENFORCEMENT OFFICIALS THE ATTORNEY GENERAL'S INVOLVEMENT IN A PRIVATE MIDEAST PIPELINE PROPOSAL MAY PRESENT MORE POLITICAL THAN LEGAL PROBLEMS... ⋆ A CHURCH AND AS MANY AS FIVE HUNDRED HOMES WERE BURNED TO THE GROUND AT A [UNK] CAMP NEAR CAPE TOWN'S AIRPORT AND RESIDENTS ACCUSED THE POLICE OF AIDING [UNK]. ⋆ THE WASHINGTON [UNK] WON PROFESSIONAL [UNK] SUPER BOWL DEFEATING THE DENVER [UNK] FORTY TWO TO TEN IN SAN DIEGO. ⋆ DIED JAMES R. [UNK] JUNIOR EIGHTY THREE FIRST SCIENCE ADVISER AT THE WHITE HOUSE AND EX CHAIRMAN OF THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY FRIDAY IN CAMBRIDGE MASSACHUSETTS.

Figure 17. Examples of strong false positives in the WSJ *test data.* In the above figure, the x-axis is labeled by the relative position in the test corpus, in number of words. The lower vertical lines indicate reference segment boundaries ("truth"), and the upper vertical lines indicate boundaries placed by the algorithm. The fluctuating curve is the probability of a segment boundary according to an exponential model. Several of the WSJ articles are in fact a collection of brief summaries of unrelated news items. The two reference segments between word positions 235,679 and 236,547 are such composites; an excerpt from this region is shown in the above text. The segmentation algorithm "wrongly" hypothesizes several boundaries in this region.

stopping techniques, allowing a richer set of features, and by incrementally building up compound features such as phrases.

10. Conclusions

We have presented and evaluated a new statistical approach to segmenting unpartitioned text into coherent fragments. This approach uses feature selection to collect a set of informative features into a model which can be used to predict where boundaries occur in text. In this work we rely exclusively on simple lexical features, including a topicality measure and a number of cue-word features, that are automatically selected from a large space of candidate features.

We have proposed a new probabilistically motivated error metric for the assessment of segmentation algorithms. Qualitative assessment as well as the evaluation of our algorithm with this new metric demonstrates its effectiveness in two very different domains, financial newswire text and broadcast news transcripts.

Acknowledgements

We thank Alex Hauptmann and Michael Witbrock for discussions on the segmentation problem within the context of the *Informedia* project. Participants in the Topic Detection and Tracking pilot study, including James Allan, Jaime Carbonell, Bruce Croft, George Doddington, Larry Gillick, Jay Ponte, Rich Schwartz, Charles Wayne, Jon Yamron, and Yiming Yang provided invaluable feedback on this work. We are grateful to George Doddington for his proposed modification to our original error metric, which we have adopted here. Thanks are also due to Stanley Chen, who provided useful comments on a preliminary version of this paper, and to the anonymous reviewers who made a number of very useful suggestions and identified several weaknesses in our original paper.

The research reported here was supported in part by NSF grant IRI-9314969, DARPA AASERT award DAAH04-95-1-0475, an IBM Cooperative Fellowship and the ATR Interpreting Telecommunications Research Laboratories.

Notes

1. A preliminary version of this paper was presented at the 1997 Conference on *Empirical Methods in Natural Language Processing (EMNLP)*, Providence, RI.

References

1. Beeferman, D. & Berger, A. & Lafferty, J. (1997). A model of lexical attraction and repulsion. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics.*
2. Berger, A. & Della Pietra, S., & Della Pietra, V. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.
3. Breiman, L. & Friedman, J. & Olshen, R. & Stone, C. (1984). *Classification and Regression Trees.* Wadsworth, Belmont.
4. Christel, M. & Kanade, T. & Mauldin, M. & Reddy, R. & Sirbu, M. & Stevens, S. & Wactlar, H. (1995). Informedia digital video library. *Communications of the ACM*, 38(4):57–58.
5. Della Pietra, S. & Della Pietra, V. & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.
6. Topic Detection and Tracking Workshop. (1997) Unpublished workshop notes.
7. Gelman, A. & Carlin, J. & Stern, H. & Rubin, D. (1995). *Bayesian Data Analysis.* Chapman & Hall, London.
8. Hastie, T. & and Tibshirani, R. (1990). *Generalized Additive Models.* Chaman and Hall.
9. Hearst, M. A. (1994). Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics.*
10. Hearst, M. A. (1997). TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.
11. Hirschberg, J. & Litman, D. J. (1993). Disambiguation of cue phrases. *Computational Linguistics*, 19(3):501–530.
12. Jelinek, F. & Merialdo, B. & Roukos, S. & Strauss, M. (1991). A dynamic language model for speech recognition. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 293–295.
13. Katz, S. (1987). Estimation of probabilities from sparse data for the langauge model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(3):400–401.
14. Kozima, H. (1993). Text segmentation based on similarity between words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics.*

15. Kuhn, R. & de Mori, R. (1990). A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:570–583.
16. Lafferty, J. (1993). Barking up the right tree: Estimating λ's for decision trees using the EM algorithm. IBM Research Report.
17. Lau, R. & Rosenfeld, R. & Roukos, S. (1993). Adaptive language modeling using the maximum entropy principle. In *Proceedings of the ARPA Human Language Technology Workshop*, pages 108–113. Morgan Kaufman Publishers.
18. Litman, D. J. & Passonneau, R. J. (1995). Combining multiple knowledge sources for discourse segmentation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics.*
19. Passoneau, R. J. & Litman, D. J. (1997). Discourse segmentation by human and automated means. *Computational Linguistics*, 23(1):103–139.
20. Ponte, J. & Croft, W. B. (1997). Text segmentation by topic. In *Proceedings of the First European Conference Research and Advanced Technology for on Digital Libraries*, 120–129.
21. Reynar, J. C. (1994). An automatic method of finding topic boundaries. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics.*
22. Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10:187–228.
23. Xu, J. & Croft, W.B. (1996). Query expansion using local and global document analysis. In *Proceedings of the Nineteenth Annual ACM SIGIR Conference on Research and Development in Information Retrieval.*
24. Yamron, J. (1998). Topic detection and tracking segmentation task. In *Broadcast News Transcription and Understanding Workshop.*
25. Yamron, J. & Carp, I. & Gillick, L. & Lowe, S. & Van Mulbregt, P. (1998). A hidden Markov model approach to text segmentation and event tracking. In *Proceedings of the International Conference on Acoustics and Signal Processing (ICASSP).*

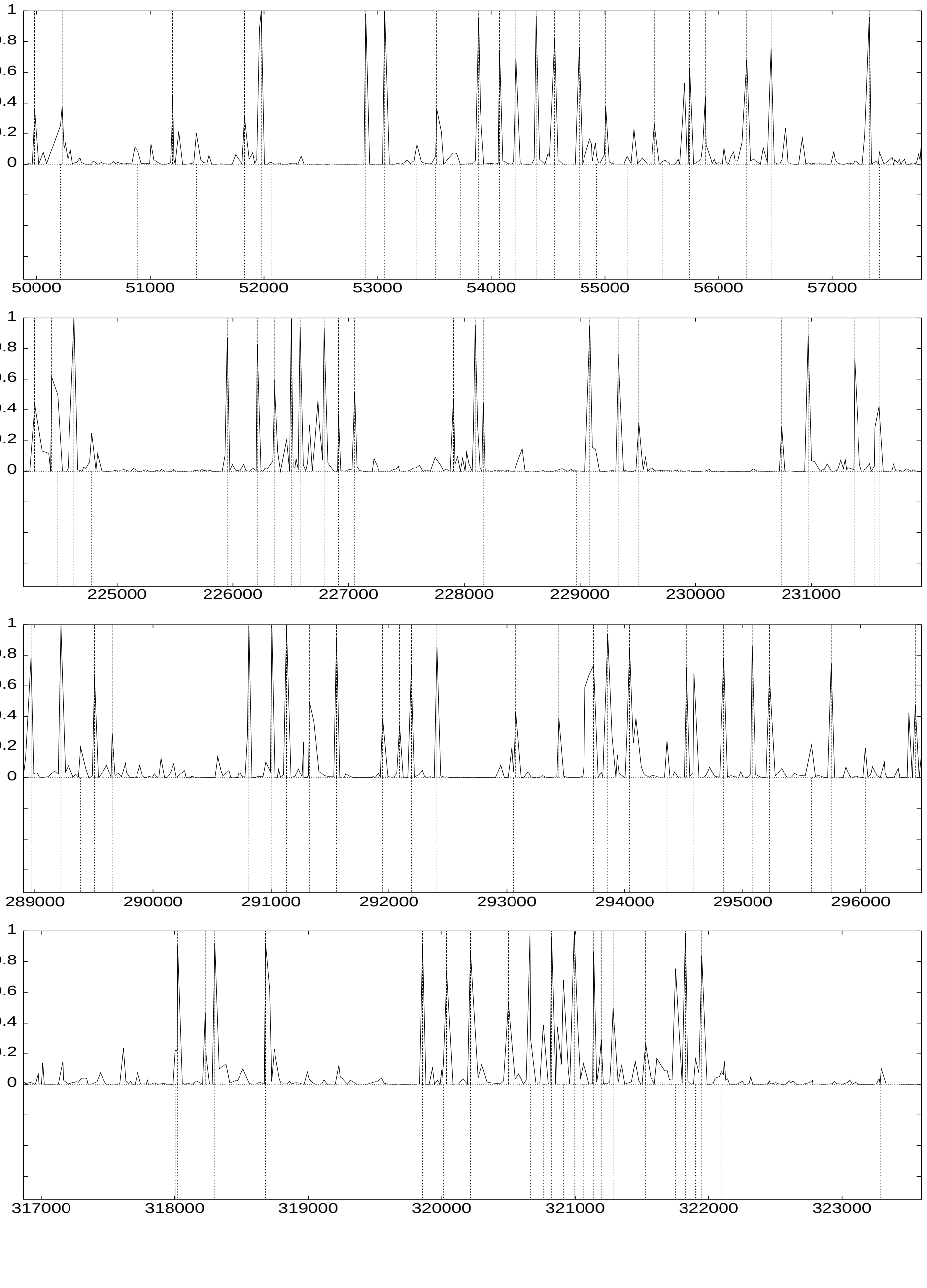


Figure 18. Typical segmentations of WSJ *test data.* The x-axis is labeled by the relative position in the test corpus, in number of words. The lower vertical lines indicate reference segment boundaries ("truth"), and the upper vertical lines indicate boundaries placed by the algorithm. The fluctuating curve is the probability of a segment boundary according to an exponential model constructed by automatically inducing 100 topicality and cue-word features. The error probability of the model was $P_k = 0.19$ with window size $k = 214$ words.

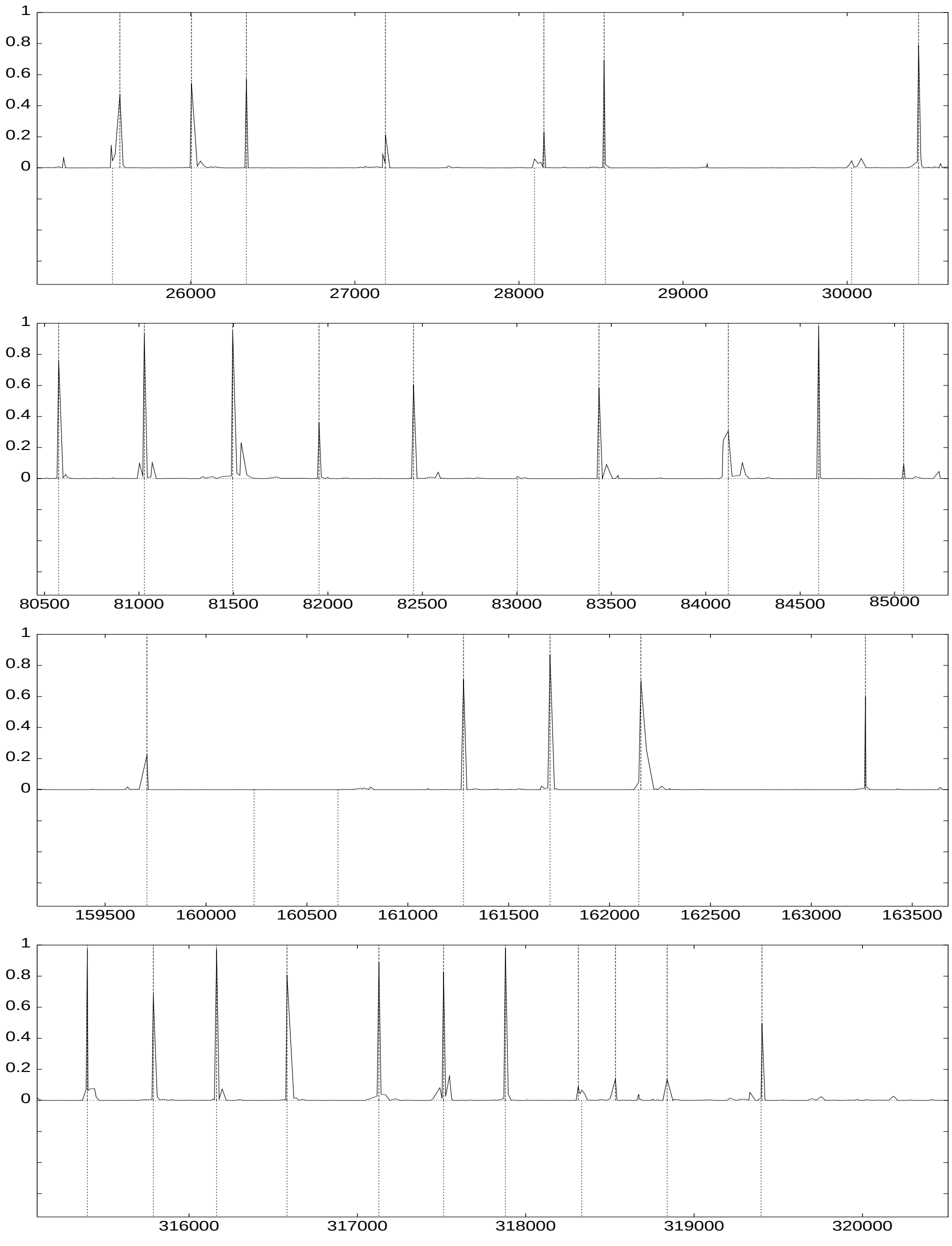


Figure 19. Typical segmentations of CNN test data. The exponential model used to segment the data was the result of automatically inducing 100 topicality and cue-word features. The error probability of the model was $P_k = 0.132$ with window size $k = 498$ words.