Capturing Term Dependencies using a Sentence Tree based Language Model

Ramesh Nallapati and James Allan
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{nmramesh, allan}@cs.umass.edu

ABSTRACT

We describe a new probabilistic Sentence Tree Language Modeling approach that captures term dependency patterns in Topic Detection and Tracking's (TDT) Story Link Detection task. New features of the approach include modeling the syntactic structure of sentences in documents by a sentence-bin approach and a computationally efficient algorithm for capturing the most significant sentence level term dependencies using a Maximum Spanning Tree approach, similar to Van Rijsbergen's modeling of document-level term dependencies.

The new model is a good discriminator of on-topic and off-topic story pairs providing evidence that sentence level term dependencies contain significant information about relevance. Although runs on a subset of the TDT2 corpus show that the model is outperformed by the unigram language model, a mixture of the unigram and the Sentence Tree models is shown to improve on the best performance especially in the regions of low false alarms.

Keywords : Term dependencies, Language modeling, Probabilistic approaches, Information retrieval, Story link detection, Word co-occurrences, sentences, maximum spanning tree, Topic Detection and Tracking

1. INTRODUCTION

Language Models have been found to be very effective in several information retrieval tasks. In the Language Modeling approach, we measure relevance of a document to a topic by the probability of its generation from the topic model [1]. One major assumption made in the unigram language modeling is the independence of all terms with respect to one another. This allows us to compute the probability of generation of a document as the product of probabilities of generation of each term in the document, as shown in the following equation:

$$P(D|M) = \prod_i P(w_i|M) \tag{1}$$

where D is the document in question, M is the topic model and w_i is the i-th term in the document.

But to quote the famous probability theorist De Finetti, "dependence is the norm rather than the contrary" [2]. From our own understanding of natural language, we know that the assumption of term independence is a matter of mathematical convenience rather than a reality. For example, a document that contains the term 'Bin Laden' is very likely to contain the terms 'Al-Qaeda', 'Afghanistan', etc.

However, the 'bag of words' approach of the unigram language modeling, as shown in equation 1, ignores all the dependencies and any other positional information of terms in the document. Hence, it seems desirable to have a more sophisticated model that is capable of capturing the semantics of documents rather than just the term distributions. A first step towards achieving this objective is capturing term dependencies, since dependencies establish associations between terms and may shed more light on the underlying semantics of the document than unigram term distributions alone. For example, if a document has strong dependencies between the terms 'white', and 'house' it may help increase our belief that the document speaks about the presidential residence rather than about the color white or about houses in general.

The present work is an attempt to capture term dependencies using a new variation of the language modeling approach that models a sentence, rather than a term as a single unit of occurrence.

The remainder of the report is organized as follows. In section 2, we present a brief description of Topic Detection and Tracking paradigm and the Story Link Detection task. Section 3 summarizes attempts made in the past in capturing dependencies and past work done in the Story Link Detection task. We present the methodology of the new *Sentence Tree* language modeling approach in section 4. In this section, we describe the motivation behind the approach, several modeling issues and also a short sketch of the intuitive understanding of the modeling of a sentence that we present in this approach. Section 5 describes the implementation details including the heuristic algorithm used in the sentence segmentation task. In section 6, we describe the experiments performed and present the results obtained on the training and test sets. Section 7 ends the discussion with a few observations and remarks on the performance of the Sentence Tree model.

2. TOPIC DETECTION AND TRACKING

The new model we present in this work is expected to address some of the issues in a body of research and evaluation paradigm called Topic Detection and Tracking (TDT). This section presents a brief overview of TDT and one of its core subtasks called Story Link Detection (SLD), on which all our experiments are performed.

Submission for CIKM 2002. Please do not cite or distribute this draft paper.

Topic Detection and Tracking (TDT) is a research program investigating methods for automatically organizing news stories by the events that they discuss. TDT includes several evaluation tasks, each of which explores one aspect of that organization–i.e., splitting a continuous stream of news into stories that are about a single topic ("segmentation"), gathering stories into groups that each discuss a single topic ("detection"), identifying the onset of a new topic in the news ("new event detection"), and exploiting user feedback to monitor a stream of news for additional stories on a specified topic ("tracking").

2.1 Story Link detection task

Another TDT evaluation task, Story Link Detection (SLD), requires determining whether or not two randomly selected stories discuss the same topic. Unlike the other tasks that have value in and of themselves, SLD is a component technology: it can be used to address each of the other tasks. For example, in order to recognize the start of a new topic, a candidate story might be compared to all prior stories to see whether the topic appeared earlier. Similarly, tracking stories on a specified topic can be done by comparing each arriving story to the user-supplied list of on-topic stories.

In the language modeling approach to Story Link Detection, we build a topic model $M(D_1)$ from one of the stories D_1 in the pair (D_1, D_2). A topic model, as the name indicates, is a mathematical representation of the topic and typically consists of estimates of probability distributions of tokens such as unigrams, bigrams or word pairs, etc. In the SLD task, the probability estimates are directly computed from the statistics of tokens in the story D_1. We then compute the probability that the second story D_2 is generated from the topic model $M(D_1)$.

Sometimes we may compute a *two-way score* to add symmetry to the formula, as shown below:

$$score(D_1, D_2) = \frac{1}{2}(P(D_2|M(D_1)) + P(D_1|M(D_2))) \quad (2)$$

If the score exceeds a pre-determined threshold, the system decides the two stories are linked. The system's performance is evaluated using a DET curve [3] that plots miss rate against false alarm at different values of decision-threshold. A Link Detection cost function C_{link} is then used to combine the miss and false alarm probabilities into a single normalized evaluation score [4].

In the present work, our model is implemented and evaluated entirely on the SLD task. We use C_{link} as the primary measure of effectiveness and show DET curves to illustrate the error trade-offs.

3. PAST WORK

In this section, we briefly summarize past work done on modeling dependencies in various areas of Information Retrieval and approaches used in the Story Link Detection task in specific.

3.1 Modeling dependencies

Van Rijsbergen tried to capture document level term dependencies in his probabilistic modeling approach using Expected Mutual Information Measure (EMIM) scores between terms [5] . A maximum spanning tree is constructed, with the terms as the nodes and the EMIM scores as the weighted edges. The tree captures the maximum dependencies between terms in the document. These dependencies are used in computing the similarity score between two documents. However, the approach is computationally expensive and also unfortunately did not show promising results.

Robertson and Bovey [6] tried including term pairs that have observable dependencies as separate terms with weights slightly different from the sum of weights or in some other way to allow for specific dependencies.

Turtle and Croft [7] investigated the use of an explicit network representation of dependencies by means of Bayesian inference theory. The use of such network generalizes existing probabilistic models and allows integration of several sources of evidence within a single framework.

More recently, Fung and Crawford [8] have worked on concept based information retrieval that captures dependencies between 'concepts' using a Bayesian inference network. One drawback of this approach is that the user has to identify the concepts manually in each document.

Attempts were also made to capture word dependencies using the vector space model. The generalized vector space model [9] is one such example which showed encouraging results.

In a related work, Conrad and Utt [10] developed a system to discover relationships between features such as name, organization, *etc.*, based on the strength of their stochastic dependencies. This is a good example of an application that addresses types of information needs that typical information retrieval systems based on term frequencies cannot handle.

In the area of language modeling, most attempts at capturing dependencies have been in the form of multigram language models [11]. Bigram and trigram models, though highly successful in the speech recognition task, have not met with great success in the domain of information retrieval systems.

3.2 Past work on SLD

Story link detection is a fairly new task that, apparently because it is not a compelling application itself, has been explored by very few researchers. The primary technique that has been deployed to date is based upon the vector space model [12, 13]. In that case, both stories are converted to vectors in a high-dimensional space. If the angle between the vectors is small enough (i.e., they are similar enough stories) then the stories are declared to be on the same topic. The threshold is determined empirically.

Most recently, some work has been done exploring the use of language models to address SLD [14]. This work compared the effectiveness of a simple unigram language model to a vocabulary expansion device known as relevance modeling. That work did not address any dependencies between terms, except indirectly to the extent that the expanded vocabulary was implicitly based on word co-occurrences within entire documents.

In the current work, we present a new Sentence Tree based Language Model (SenTree) that attempts to capture term dependencies within a sentence.

4. METHODOLOGY

Recall that our goal is to build a model of story D_1 and then decide whether story D_2 is predicted by that model. This section describes the process of constructing a model from D_1. We begin our discussion by presenting the motivation and ideas behind the new model.

4.1 Exploiting sentence structure

A universal feature of all documents is the syntactic structure of sentences. Webster's dictionary defines a sentence as "a word, clause, or phrase or a group of clauses or phrases forming a syntactic unit which expresses an assertion, a question, a command, a wish, an exclamation, or the performance of an action...". In other words, the semantics of a document are expressed through the syntax of sentences. Hence, we believe that a language model that reflects the syntactic structure of sentences in an explicit manner may capture the underlying semantics of the document.

As we have seen earlier, unigram language models completely ignore the syntactic formation of sentences in documents. In the present approach, we attempt to capture it by modeling a document as a collection of sentences rather than as a 'bag of words'. Figure 1 contrasts the view of a document as seen by the unigram model and the new Sentence Tree based language model. The unigram model views a document as a bin of words while the Sentence Tree model views it as a collection of smaller bins, each of which represents a sentence.

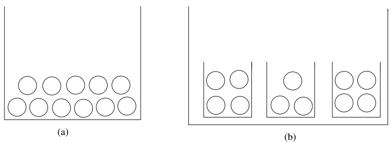

Figure 1: A document as viewed by (a) the Unigram Language Model and (b) Sentence Tree based Language model

In the unigram language model, one can think of the process of document generation from the model as a random experiment in which we pick up n terms with replacement from a bag of words that represents the topic model, where n is the length of the document. The relevance score of the document with respect to the topic model is then given by the probability that the outcome contains all the terms of the document.

In the Sentence tree language modeling approach, which we call the *SenTree* model in short from now on, the process of document generation is viewed as a random experiment in which we pick s sentences with replacement from the model, where s is the number of sentences in the document. The relevance score of the document is equal to the probability that the outcome contains all the sentences in the document. Since a sentence represents a semantic unit, we hope that computing the probability of generation of each sentence rather than each term better captures the semantics of the document. However, the probability of sentence generation is difficult to compute due to the sparse nature of data and hence we must use certain assumptions and simplifications to compute this probability.

4.2 Probability of a Sentence

In the SenTree approach, we assume each sentence to be independent of the other sentences. This assumption is certainly not valid but it is less stringent than the assumption of term independence. The assumption allows us to compute the probability of generation of a story from a topic model as follows:

$$P(D|M) = \prod_i P(S_i|M) \tag{3}$$

where M is a topic model and S_i is the i-th sentence in a story D. The tricky part is computing the probability of generation of each sentence. Ideally, one would have to compute the probability of generation of a sentence as follows:

$$P(S|M) = P(w_1, w_2, ..., w_n|M) \tag{4}$$

where w_i is the i-th term and n is the number of terms in the sentence S. However, the data from the topic is typically very sparse and it is almost impossible to compute to a reasonable level of accuracy the joint probability of terms in a sentence. To overcome this problem, we model the sentence as a maximum spanning tree similar to the approach presented by Van Rijsbergen [5].

4.3 Maximum spanning tree representation of a sentence

Using the chain rule, the joint probability in equation 4 can be expressed as

$$\begin{aligned} &P(w_1, w_2, .., w_n|M) = P(w_1|M) \times P(w_2|w_1, M) \times \\ &P(w_3|w_2, w_1, M) \times .. \times P(w_n|w_1, w_2, .., w_{n-1}, M) \end{aligned} \tag{5}$$

As an approximation to this exact formula, we ignore the higher order dependencies and select from the conditioning variables, one particular variable that accounts for most of the dependence relation. In other words, we seek an approximation of the form

$$P(w_i|w_{i-1}, ..., w_1, M) \approx \max_{1 \le j < i} P(w_i|w_j, M) \tag{6}$$

Note that the numbering of terms $(1, 2, ..., n)$ need not be same as the order in which they occur in the sentence. It could be any permutation $(m_1, ...m_2)$ of the natural order. To summarize, the approximate probability distribution is then given by

$$P(S|M) \approx P_a(S|M) = \prod_{i=1}^{n} P(w_{m_i}|w_{m_{j(i)}}, M) \;\; 0 \le j(i) < i \tag{7}$$

where $(m_1, m_2, ..., m_n)$ is a permutation of the natural order $(1, 2, ..., n)$, $j()$ is a function mapping i into integers less than i, and m_0 is defined such that

$$P(w_{m_i}|w_{m_0}, M) = P(w_{m_i}|M) \tag{8}$$

We need to choose a permutation and a function $j()$ that gives the best approximation to the probability in equation 5 and the approach we used is described in the following section.

4.4 Computing the best approximation $P_a(S)$

Now, our problem is to find a probability function of the form $P_a(S)$ which is the best approximation to the true joint probability function $P(S)$. For each sentence S in the story , an undirected graph is constructed with the terms as nodes and degree of dependency between term pairs as edge weights. The degree of dependency is measured by the Jaccard Coefficient (J) as shown below:

$$J(w_i, w_j) = \frac{n(w_i \cap w_j)}{n(w_i) + n(w_j) - n(w_i \cap w_j)} \tag{9}$$

where n_s is the total number of sentences and $n(w)$ is the number of sentences in which the argument w occurs, both values taken from the story from which the topic model is generated, while $\cap$ should be read as 'occurring in the same sentence as'.

The value of J is assumed to be zero for a word pair that does not occur in the story from which the topic model is obtained. Note that there also other measures such as the Pointwise Mutual Information Measure PMIM [15] for measuring the degree of dependence. In this work, we have used the Jaccard Coefficient for reasons of ease in computation.

The best approximation P_a is given by the Maximum Spanning Tree (MST) on this graph. The MST incorporates the most significant of dependencies between the terms of the sentence subject to the global constraint that the sum of them should be a maximum. We compute the MST using a greedy approximation algorithm. Once the MST has been computed, the approximating distribution P_a can be written down by traversing the tree in a breadth-first or depth-first manner, starting with any of the leaf nodes as the root node. It can be shown that the resulting distribution will be the

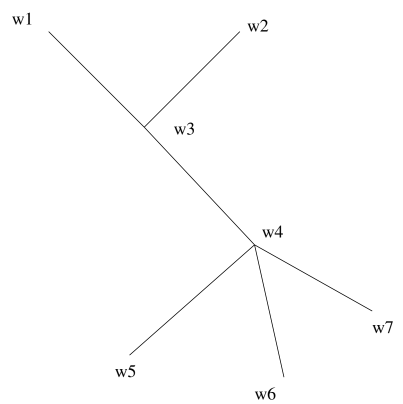

Figure 2: Generating the approximate distribution P_a from the MST of a sentence

same irrespective of the root node chosen [5]. As an example, consider the MST representation of a sentence $(w_1, .., w_7)$ as shown in figure 2. The approximate probability distribution $P_a(S)$, obtained by traversing the tree in a breadth-first manner starting from w_1 as the root node is as follows.

$$\begin{aligned} P_a(w_1, ..w_7|M) &= P(w_1|M) \times P(w_3|w_1, M) \times \\ & P(w_2|w_3, M) \times P(w_4|w_3, M) \times \\ & P(w_5|w_4, M) \times P(w_6|w_4, M) \\ & \times P(w_7|w_4, M) \end{aligned} \tag{10}$$

We have thus far seen how to obtain a distribution $P_a(S)$ that approximates the joint probability of generation of a sentence. It still remains to be shown how to compute the bigram probabilities of the form $P(w_i|w_j, M)$ in equation 10. The next sections shows how the topic model computes these probabilities using maximum likelihood smoothed estimates.

4.5 Constructing the topic model

As mentioned in section 2.1, a topic model provides us with the estimates of probabilities that we need in computing the relevance score of a document with respect to the topic. As shown in section 4.3, we need conditional probabilities $P(w_i|w_j, M)$ for all pairs of tokens that form edges in the MST representation of any sentence in the story under consideration. The topic model estimates these probabilities using the maximum likelihood estimate as shown below:

$$P(w_i|w_j, M) \approx \frac{n(w_i \cap w_j)}{n(w_j)} \tag{11}$$

where the terms in the equation have their usual meaning.

However, since the data that makes up a topic model is typically sparse, we encounter the problem of zero probabilities. If the term w_j does not occur in the story that the topic model is generated from, the probability $P(w_i|w_j, M)$ is assumed to be zero. It is also possible that there is no instance of w_i and w_j occurring in a single sentence in the topic model. In such scenarios, the conditional probability would vanish, forcing the entire probability of sentence generation to zero. In our model, we smooth every conditional probability term with the probability from a background model as shown below:

$$\begin{aligned} P(w_i|w_j, M_{smoothed}) &= \lambda_S P(w_i|w_j, M) + \\ & (1 - \lambda_S) P(w_i|w_j, M_B) \\ & 0 \leq \lambda_S \leq 1 \end{aligned} \tag{12}$$

where M_B is a background model of general English. The background model computes the background conditional probabilities as follows: If the terms w_i and w_j co-occur in at least one sentence in the database of the background model, we use

$$P(w_i|w_j, M_B) = \frac{n_B(w_i \cap w_j)}{n_B(w_j)} \tag{13}$$

Else, if w_j occurs in the database but w_i and w_j are not found to co-occur, we use the following approximation:

$$P(w_i|w_j, M_B) = \frac{1}{n_B(w_j)} \tag{14}$$

In the worst case, if neither w_i nor w_j is found in the database, we use the following approximation:

$$P(w_i|w_j, M_B) = \frac{1}{n_{s_B}} \tag{15}$$

where $n_B(w)$ is the number of sentences in the background database in which the argument w occurs and n_{s_B} is the total number of sentences in the background database.

The value of λ_S is determined by performing a parameter sweep over its entire range of values. This involves running the model on a training set of data for several values of λ_S and measuring the performance of the model each time. The best performing value is then chosen as the default system value of λ_S.

4.6 Likelihood ratio and Mixture model

Finally, the story's relevance score with respect to a topic is given in terms of likelihood ratio which is defined by the following equation:

$$\begin{aligned} Score(D, M) &= \frac{P(D|M_{smoothed})}{P(D|M_B)} \\ &= \frac{\prod_i P(S_i|M_{smoothed})}{\prod_i P(S_i|M_B)} \\ &= \prod_i \frac{P(S_i|M_{smoothed})}{P(S_i|M_B)} \end{aligned} \tag{16}$$

Note that in computing the probability of sentence generation with respect to the background model, we use the same approximate probability distribution function P_a that we generated from the topic model. Computing the likelihood ratio with respect to the background model provides a sound basis for comparison of relevance and also serves as a normalizing factor to sentence and document lengths.

Another variation in Language Modeling that is often employed is combining two different models. In the Story Link Detection task, unigram language models have been found be very effective discriminators. Hence, it makes sense to use the SenTree model as a sort of enhancement to the unigram approach. One way to accomplish this is a linear combination of the unigram and SenTree scores as shown below:

$$\begin{aligned} Score_{Mixture}(D, M) &= (1 - \theta) \times Score_{SenTree}(D, M) + \\ & \theta \times Score_{Unigram}(D, M) \\ & 0 \leq \theta \leq 1 \end{aligned} \tag{17}$$

Again, θ is determined by empirical experiments.

4.7 Algorithm of the SenTree model

In this subsection we summarize the model description with a step-by-step algorithm of the process of computing the relevance score of story D_2 with respect to a topic model of the story D_1.

1. Segment both the stories D_1 and D_2 into sentences.
2. Remove punctuation, perform case folding, remove stop words and stem all the word-tokens.
3. Index terms in both the stories with frequency counts as well as the indices of the sentences in which they occur. The topic model of D_1 is now readily accessible from the index.
4. For each sentence in D_2
   (a) Build a fully-connected undirected graph of the sentence with the terms as nodes and Jaccard coefficient between each pair of terms measured from the topic model of D_1 as weighted edges as described in section 4.4.
   (b) Construct an approximate maximum spanning tree of the graph using a greedy algorithm. (See section 4.4.)
   (c) Generate a probability distribution function $P_a(S)$ that approximates the joint probability of the sentence.
   (d) Evaluate the probability of the sentence using smoothed probability estimates (see section 4.5) from the topic model of D_1.
   (e) Evaluate the same probability of the sentence generation with respect to a background model and compute the likelihood ratio (see section 4.6).
5. Compute the product of the likelihood ratios of all the sentences. This is done on a log scale to avoid numerical underflow.
6. Return the product as the relevance score of story D_2 with respect to the model of story D_1.

4.8 Computational complexity

In this subsection we discuss the complexity of implementing the SenTree Model. Let the story contain N sentences and at most T terms per sentence. The running time of each step in the algorithm is presented below:

1. Constructing the graph of a sentence with weighted edges: This requires computing the Jaccard coefficient for all pairs of terms in the sentence, each of which can be done in constant time using hash tables. Thus, this step has a complexity of $O(T^2)$.
2. Building a maximum spanning tree: A greedy algorithm is used to build the MST. The running time of this step is $O(T^2 Log(T))$ if we use disjoint-set forest implementation with union by rank and path compression heuristics [16].
3. Computing the probability of occurrence of the sentence from the topic model and the background models: This requires generating the probability distribution function of the sentence using Breadth First Search. The probabilities of each edge in the MST can be computed in constant time. Hence, this step has a complexity of $O(T)$, which is the size of the MST.

Thus, the overall running time per sentence is $O(T^2 Log(T))$. Thus, for the entire document, the complexity is simply given by $N \times O(T^2 Log(T)) = O(T^2 N Log(T))$. In comparison, Van Rijsbergen's algorithm [5] of building a document level spanning tree has a complexity of $O((TN)^2 Log(TN))$. We have thus been able to reduce the run time by a factor of $O(N \frac{Log(N)}{Log(T)})$ by building only sentence level spanning trees.

4.9 An intuitive understanding of the MST representation of a sentence

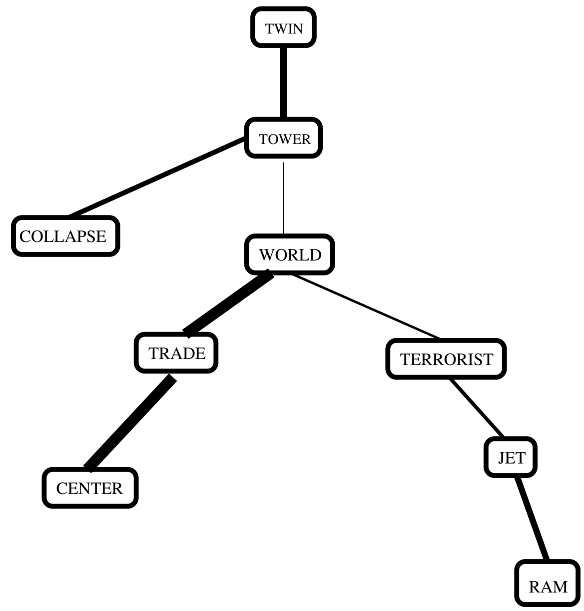

Figure 3: Maximum dependence tree of an on-topic sentence

Before we end the discussion on the methodology of the SenTree model, we will try to present an intuitive understanding of the MST representation of a sentence with the help of an example. We first note that the MST representation of any sentence in a story is topic dependent, since the edge weights are computed from the topic data. One may visualize this phenomenon as the topic-model's 'understanding' of the semantics of a sentence from its own knowledge of the topic. We expect the topic model to build a 'meaningful' representation only when the sentence is about the topic.

As an example, we have built the MST representation of the sentence "Twin towers collapse as terrorists ram jets into the World Trade Center" with respect to the topic model 'Terrorist attacks in America' constructed from a set of three stories collected from *www.cnn.com* published on September 12th, 2001. We have followed the usual procedure outlined in section 4.4, *i.e.*, building a sentence graph using Jaccard Coefficients first and then building an MST of the graph using a greedy algorithm. The MST representation of this sentence with respect to the model is shown in figure 3. The edge widths roughly represent the degree of dependence between the nodes as measured by the Jaccard Coefficient. We notice that the edges (world, trade) and (trade, center) have strong dependency weights. This is expected, as the terms together form a single phrase and occur frequently in any document concerning the WTC attacks. The same is true with the pair (twin, tower). The edge (jet, ram) also has a strong weight as the terms together contain very vital information on the event.

Thus we see that the MST representation of a sentence assigns strong weights to phrases as well as word pairs that are illustrative of the information content of the document. Such 'understanding' of a sentence may not be expected from a model that discusses a completely different topic.

5. SYSTEM DETAILS

In this section we present a brief overview of the system we used in our experiments. Figure 4 is a flow chart of the several stages in the process of computation of the relevance score of story D_2 with respect to the model of story D_1. Each block in the chart represents a stage and the arrows indicate the sequence in which the operations are performed.

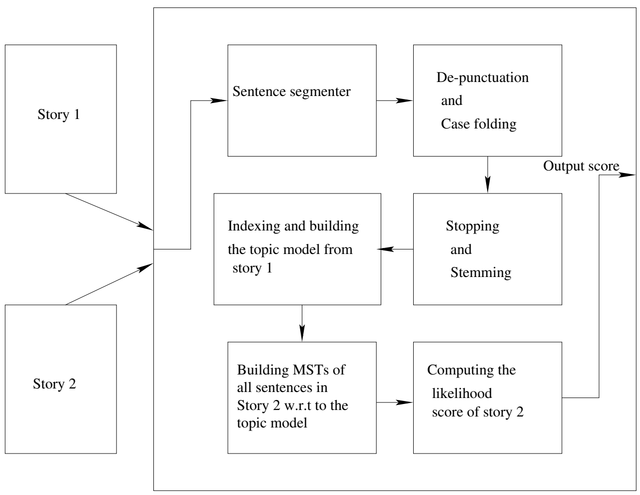

Figure 4: System overview

In the following subsections, we discuss some of the system implementation details that are not covered in the discussion on methodology of the model.

5.1 Sentence segmentation

We have used a simple heuristic-rule based sentence segmenter to segment sentence boundaries in the stories. The algorithm of the segmenter is as follows.

1. Remove quotes and replace repeated occurrences of periods, question marks or exclamation marks with just one occurrence.
2. Remove periods in the following abbreviations: Mr., Ms., Dr., St., Sr., Jr., etc.
3. If a period, a question mark or an exclamation mark is not immediately preceded by a string consisting of a period or a white-space character followed by any letter of the alphabet and succeeded by a white-space character, then mark the period, question mark or the exclamation mark as a sentence boundary.

Over a large number of observations, we have found that the algorithm correctly identifies the boundaries of most of the syntactically correct sentences. However, we have not yet done any quantitative evaluation on the algorithm. We also feel that we may need a more intelligent system with some background knowledge of the language to identify boundaries of sentences that may have incorrect punctuation. Incorrect punctuation are a common occurrence in stories obtained from automatic speech recognition systems. Such stories form a substantial part of the TDT2 corpus but are filtered out from the training and test sets used in the present evaluation.

Since the performance of the present algorithm is found satisfactory for syntactically correct sentences, the authors feel that any marginal improvement in its performance may not significantly improve the overall performance of the SenTree model.

5.2 Corpus and other information

The present evaluation is run on a subset of TDT2 English corpus that include six months of material drawn on a daily basis from six English news sources. The subset we collected is divided into training and test sets. The training set comprises 6300 story pairs while the test set comprises 2749 story pairs. Expert judgments are included with the corpus for all the story pairs in the training as well as the test sets.

A list of 423 high-frequency words is used to eliminate stop words. Stemming is done using the Porter stemmer [17] while the indexer and the MST algorithm are built using Java.

6. RESULTS AND DISCUSSION

This section presents the results of the experiments performed using the SenTree model. For all the experiments described below, the DET curve obtained from the best performing unigram language model is used as a baseline. The system is first trained using the training set of stories until the best performing values of parameters such as the smoothing parameter λ and the mixing parameter θ are found. Once the values are set, the performance of the system is evaluated on the test set. The following subsections present the results of the experiments performed on the training and test sets.

6.1 Training the system

Training essentially involves searching for the values of various parameters that deliver the best performance. In our training experiments, we trained our system using the SenTree only first and compared the performance with that of the baseline. Next, a system that implements a mixture model is trained on the training set. The observations of the experiments are presented below.

6.1.1 SenTree Model only

As a first step, the SenTree model alone is run on the corpus. A parameter sweep is performed on the value of λ_S, the smoothing parameter. The DET curve of the best performing value of λ_S (0.07) is shown in figure 5. It is clear from the plot that the unigram model outperforms the SenTree model since, the former has lower values of miss rate and false alarm in the entire range and a C_{link} that is also much lower. This suggests that the frequencies of occurrence of terms is a more important feature of relevance than sentence level dependencies. Notwithstanding this fact, it is noted that the SenTree model is still encouraging and it is hoped that the best performance can be improved by a mixture of both the models.

6.1.2 Mixture model

As described in section 4.6, a simple parametric linear combination of the unigram model and the SenTree model is run on the corpus. In the mixture model, we need to learn the values of the smoothing parameter for the unigram model λ_U, smoothing parameter for the SenTree model λ_S and the mixture parameter θ. Hence we performed a three dimensional parameter sweep on the entire

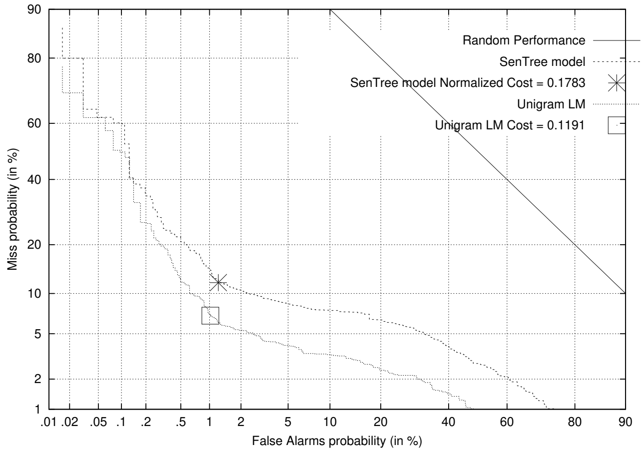

Figure 5: Best performing SenTree model on the training set

range of the values of the three parameters. The best performing values on the training data are found to be $\lambda_U = 0.05,\ \lambda_S = 0.45$ and $\theta = 0.85$. The performance of the system that uses these values is shown in comparison to the same baseline in figure 6.

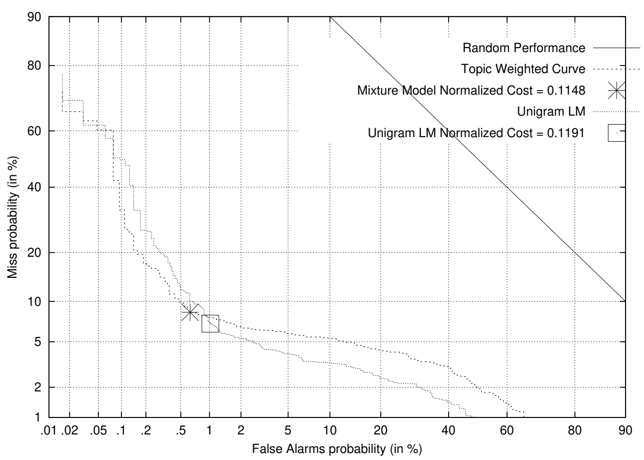

Figure 6: Best performing mixture model on the training set

We note that the mixture model performs better than the unigram model in the regions of low false alarm. For example, at 0.1% false alarm, the miss rate is reduced by around 15% compared to the baseline performance of the unigram model. However at low miss rates, the combination performs worse than the baseline. Hence the mixture model may be preferred to the unigram model if the application demands operation in the region of low false alarms.

6.2 Testing the system

Having found the best performing values of various parameters, we now run the system on the test set. As before, we run the sentence based language model alone as well as the mixture model separately. The results are summarized in the following subsections.

6.2.1 SenTree model only

The system is first run on the test set using the SenTree model only. The performance of the system is shown in 7. Once again, we notice that the SenTree model does not perform as well as the unigram model.

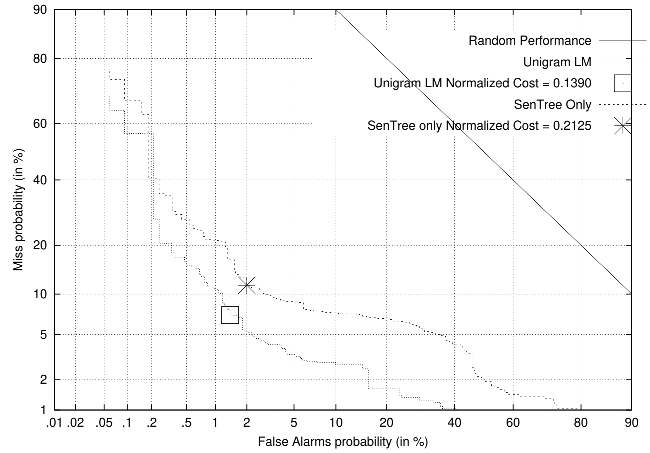

Figure 7: Best performing SenTree model on the test set

6.2.2 Mixture model

Now the mixture model is run on the test set with the parameters fixed at values presented in section *6.1.2*. We notice that the results are consistent with those from the training set. The mixture model outperforms the unigram model in regions of low false alarm. We also note that the mixture model has succeeded in lowering the normalized cost function from 0.1390 to 0.1179. The DET curve of this run is shown in figure 8.

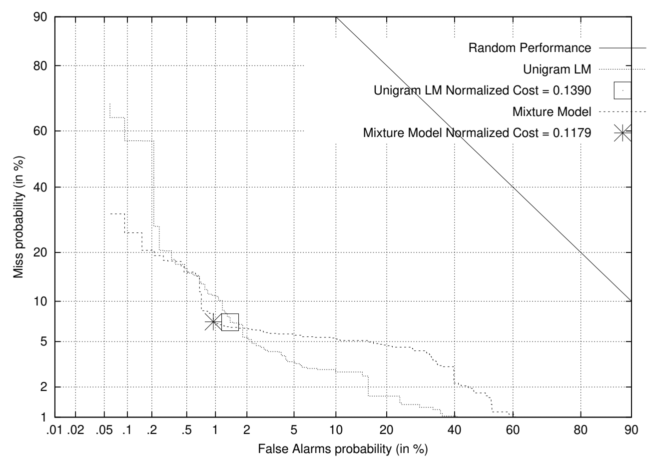

Figure 8: Best performing Mixture model on the test set

7. CONCLUSIONS

In this work, we have presented a new approach of modeling a document by exploiting the syntax of sentences. The approach captures within-sentence dependencies by modeling each sentence as a maximum spanning tree of dependence. Although this approach is built towards addressing a specific TDT task, we believe that the generality of the model permits one to apply it to any text classification task.

Our experiments indicate that sentence level dependency alone is not a better measure of relevance than simple unigram approach, but is still a good discriminator between on-topic and off-topic story pairs. We have also seen that the performance of the unigram models can be enhanced by supplementing the unigram model with the sentence model.

We believe that, apart from a slight improvement in performance, the most important contribution of this work is the evidence we provided that capturing the sentence level dependencies can be a good measure of relevance. We hope that this encouraging result paves the way towards building more sophisticated models that eventually achieve the ultimate goal of capturing the exact semantics of natural language.

Acknowledgments

We would like to thank Victor Lavrenko for his valuable comments and suggestions. This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-99-1-8912. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] J. M. Ponte and W. B. Croft, A Language Modeling Approach to Information Retrieval, *ACM SIGIR*, 275-281, 1998.

[2] B. De Finetti, *Theory of Probability*, 1:146-161, Wiley, London 1974.

[3] A. Martin, G. Doddington, T. Kamm and M. Ordowski, The DET curve in assessment of detection task performance, *EuroSpeech*, 1895–1898, 1997.

[4] The Topic Detection and Tracking evaluation phase 2 plan, *http://www.nist.gov/speech/tests/tdt/tdt98/doc/tdt2.eval .plan.98.v3.7.pdf*.

[5] Van Rijsbergen, *Information Retrieval*, Butterworths, 1979.

[6] S. E. Robertson and J. D. Bovey, Statistical Problems in the Application of Probabilistic Models to Information Retrieval, *Technical Report, Center for Information Science, City University*, 1982.

[7] H. R. Turtle and W. B. Croft, *Inference Networks for Document Retrieval*, ACM SIGIR, 1-24, 1990.

[8] R. M. Fung, S.L. Crawford, L.A. Appelbaum and R.M. Tong, An architecture for probabilistic concept-based information retrieval, *ACM SIGIR*, 455-467, 1990.

[9] S.K.M. Wong, W. Ziarko, and P.C.N. Wong, Generalized Vector Space Model in Information Retrieval, *ACM SIGIR* 18-25, 1985.

[10] J. G. Conrad and M. H. Utt, A System for Discovering Relationships by Feature Extraction from Text Databases, *ACM SIGIR*, 260-270, 1994.

[11] F. Song and W.B. Croft, A General Language Model for Information Retrieval, *Information and Knowledge Management*, 1999.

[12] J. Allan, V. Lavrenko and R. Swan, Explorations Within Topic Tracking and Detection, *Topic Detection and Tracking: Event-based Information Organization*, James Allan, Editor, Kluwer Academic Publishers, 197-224, 2002.

[13] Y. Yang, J. Carbonell, R. Brown, J. Lafferty, T. Pierce and T. Ault, Multi-strategy Learning for TDT, *Topic Detection and Tracking: Event-based Information Organization*, James Allan, Editor, Kluwer Academic Publishers, 85-114, 2002.

[14] V. Lavrenko, J. Allan, E. DeGuzman, D. LaFlamme, V. Pollard and S. Thomas, Relevance models for Topic Detection and Tracking, *Proceedings of the Conference on Human Language Technology (HLT)*, 2002, Forthcoming.

[15] C. D. Manning and H. Schutze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.

[16] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[17] M.F. Porter, An algorithm for suffix stripping, *Program*, 14(3):130-137, 1980.

[18] H. Jin, R. Schwartz, S. Sista and F. Walls, Topic Tracking for Radio, TV Broadcast, and Newswire, *DARPA Broadcast news Workshop*, 199-204, 1999.

[19] J. Yamron, I. Carp, L. Gillick, S. Lowe and P. van Mulbregt, Topic Tracking in a New Stream, *DARPA Broadcast news Workshop*, 133-138, 1999.

[20] W. Bruce Croft, Howard R. Turtle, and David D. Lewis, The use of phrases and structured queries in information retrieval, *ACM SIGIR*, 32-45, 1991.