An Online Conceptual Model for Link Detection

1st Author
1st author's affiliation
1st line of address
2nd line of address

2nd Author
2nd author's affiliation
1st line of address
2nd line of address

3rd Author
3rd author's affiliation
1st line of address
2nd line of address

ABSTRACT
The underlying problem of most tasks in TDT is to measure the relevance of two stories. Similarly the basic manipulation of Information Retrieval and Filtering is to compare information stream with query. Therefore it is significant that TDT provides a task of story link detection to recognize the correlation and divergence between each pair of stories. The core of previous researches to link detection is comparing overlapping words of two stories or expanding the set of words associated with the story being processed. These researches have achieved great success in link detection task based on statistical theory. However they overlook the significance of concepts of words, and think of a story as a “words bag”. In this paper we construct an online conceptual tree to describe all online concepts of a word and their coherence with the story being processed. Additionally we compare a pair of stories using their online conceptual models generated from the online conceptual trees. We demonstrate that online conceptual model can result in substantial improvement over the relevance modeling baseline.

Categories and Subject Descriptors
H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing – Indexing methods, Linguistic processing.

General Terms
Algorithms, Experimentation.

Keywords
TDT, link detection, online conceptual tree, online conceptual model

1. INTRODUCTION
Topic detection and tracking (TDT) is a research area concerned with automatically organizing a multilingual stream of news broadcasts as it arrives over time [1]. As an important task of TDT, Link Detection (LDT) requires determining whether or not two randomly selected stories discuss the same topic. Unlike the other tasks in TDT which have directly practical value in and of themselves, such as topic tracking used for collecting related stories and topic detection used for detecting new topic, LDT is a fundamental technology. It is usually used as an underlying model of 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.

Additionally, LDT can be used as the basic research in the field of Information Retrieval (IR), Information Filtering (IF) and Data Mining. Therefore it is significant for us to further improve the performance of LDT.

2. RELATED WORK
Most previous approaches to LDT are comparing the overlapping words of two stories. That is to say, if a pair of stories have more words in common, they are more likely on same topic. Based on this theory, the earliest research in LDT focused on the statistical model such as VSM [2] which has been widely adopted in the field of IR. By extracting the terms highest weighted, VSM builds vector space to describe the main idea of stories. And it is decided whether two stories is related by comparing their vector space. Another early research of statistical model in LDT is Language Model (LM) [3][4][5], by which each story can builds its topic model. And the degree of correlation between two stories can be estimated by comparing their topic model directly. The deficiency of VSM and LM is the sparsity of contexts which make it difficult to identify the meaning of terms in current story. Therefore several works lately focused on expanding the words to include other strongly related words. The purpose of this method is to increase the likelihood that two related stories will have more important overlapping words. In this field, query expansion techniques such as Local Context Analysis [6] can be used in VSM to extend the space of terms [7]. And statistical LM can implicitly includes related words from a general background source as part of the smoothing process [4][5].

Although the statistical methods above have achieved successful result, they only represented documents using a “bag of words” approach where words are weighted according to their frequency within a document. Therefore Some researches attempt to improve these models by using lexical chains which more like to mine the semantic information rather than individual words. One research about lexical chains is SYN [8], which represents documents by solving the problem of synonym. Thus documents are represented by sets of WordNet sense identifiers, i.e. numbers which represent meanings. However, SYN provides no facility for sense disambiguation and hence actually performs worse than VSM. The other method using lexical chains is to represent the lexical coherence based on lexical resource [9][10][11][12]. It believes that a lexical chain is a succession of semantically related words in a text that creates a context and contributes to the continuity of meaning. And these methods can improve LDT by getting text to hang together as a whole [13]. Additionally, some research incorporate the semantic information of words into statistical
model by using Name Entity [14]. This method adjusts the weight of words based on their name-entity type.

Words expansion and semantic mining for traditional statistical model have achieved successful improvement for LDT. But they are usually based on general background sources, stationary dictionary or semantic classes, and ignore the truth that a word can’t estrange itself form its contexts in current story. Therefore some recent researches attempt to build topic model of stories by Relevance Model (RM) [15][16][17]. RM uses a story as a query to retrieval high relevant stories, and builds its topic model using the techniques of LM based on these stories. RM inherits the benefits of LM to expand the contexts of stories implicitly. More important it is that the contexts expanded will not come from general background but a field of related information. But RM still treats a story as a “words bag”, and the procedure of retrieval mainly focused on the overlapping words. That is to say, the stories which have more words in common with the story being processed will be selected to construct its topic model. Therefore the effectiveness of words expansion is limited. Additionally RM also overlooks the concept of words, which can’t constrain the trend that noises as related words will be combined into topic model.

In this paper we build an Online Conceptual Tree (OCT) to explore the concepts of each word and the correlation of these concepts to the stories. Then we describe the topic of a story using an Online Conceptual Model (OCM) generated from OCT. In the next section we outline the basic ideas of OCM. In Section 4 we present how OCM were adapted to build topic model, and how stories are compared based on the probability distribution of the concepts in their OCM. Section 5 describes TDT4 corpus, the plan of evaluation for LDT and experiments performed. Section 6 presents the results obtained and analyzes the performance of OCM. Section 7 draws conclusions and speculates on future work.

### 3. ONLINE CONCEPTUAL MODEL

The basic thought of online conceptual model, which is simply named OCM, is conceptual dependency among words. This dependency means that the practical meaning of a word can’t come apart from its context, but can be interpreted and described by other terms appearing around it. For example, the term “terrorism” combined with “World Trade Center” mainly deals with the topic of “9/11 Attack”, while with “Bali” describes the topic of “Bali Massacre”. Therefore online concept of a word formed with its context usually works better than its isolated semantic in dictionary.

More important reason of building OCT is dependency of comprehension between words and their contexts. This dependency believes that the concept of a term can only be comprehended on the premise that its context has been understood firstly. See the example given above, before comprehending the meaning of “9/11”, we must understand what the meaning of “WTC” is in advance. Only when we know “WTC” is a building in New York, we can distinguish the “9/11” from the terrorism “Bali Massacre” which happened in Bali of Indonesia. Therefore, there is two key problems to describe the concept of a word, one of which is to collect enough contexts of the word, the other is to comprehend the meaning of its contexts in advance.

Based on the discussion above, we build OCM to describe the dependency of concept and comprehension between words and their contexts:

$$ONM(t_i, context_j(t_i)), i \in 1 \cdots n, j \in 1 \cdots n$$

Here, $t_i$ is the target to be comprehended; $context_j(t_i)$ is the $j$-th term of contexts appearing with $t_i$ together in current story. Based on the hypothesis about dependency of comprehension, it is necessary to build the conceptual space of each $context_j(t_i)$ firstly. Additionally, it is helpful to calculate the degree of conceptual dependency between each concept of the space and the target $t_i$. Serving the two purposes above, we design a structure named online conceptual tree which will be elaborated in next section.

#### 3.1 Online Conceptual Tree

Online Notional Tree simply named OCT is a tree constructed by all of the paths between two words. The purpose to build OCT is to search all concepts of a word and to evaluate the strength of the correlation between its concepts and another word. Figure 1 shows an OCT of 2-tuple words $(t_i,t_j)$, where the root indicates the word $t_i$ and each leave indicates the word $t_j$. In the OCT, each path from root to leave consists of several stories and intermediate nodes, the quantity of which is decided by the whole depth of the tree. Additionally, each story in OCT isn’t described by all its words but those highest weighted. The most crucial rule to construct OCT is each path must reach the leave $t_j$ and the topology of intermediate node must include at least one leave, otherwise the path will be cut.

![Figure 1. An example of OCT of 2-tuple words $(t_i,t_j)$. The red circle indicates the term $t_i$, the blue circle indicates the term $t_j$, and the hollow circle indicates the intermediate circle. The yellow icon of document indicates the story](image-url)

The topology of OCT has $n$ layers, and each layer has at least one subtree. The number of layer $n$ specify the maximum range from concept of word $t_i$ to another word $t_j$, who as a parameter will be discussed for its impact to link detection in experiment. In each layer except the first one, the root of any subtree is an intermediate node coming from the topology of the next higher layer, and mediated by stories it products more new leaves and intermediate node. In the procedure to build topology, a story including the root of subtree can be used as a path to search leaves, and if it includes term $t_i$ the subtree creates a leaf accompanied with several intermediate nodes which will be the roots in next layer. If there isn’t any leave in the topology of the subtree when the layer reaches maximum range, we cut the whole subtree. Additionally the procedure of building topology stops at any leave and product new subtrees in next layer at any intermediate node.
All stories in first layer of OCT form various contexts for word \( t_j \) and each context describes a concept of \( t_j \) in specific situation. Therefore we can build a repository of concept for term \( t_j \) based on OCT. But how to decide which concept is suitable for the meaning of term \( t_j \) in current story most? An empirical hypothesis is the more paths a concept has to leaf \( t_j \) and the shorter the depth of each path is, the concept is more reasonable for the meaning of \( t_j \) in current story.

Additionally, we prefer to make a difference between the subtopics including a leaf and those without leaves in first layer. The reason is the dependency of comprehension which more likes to grasp the meaning of word \( t_j \) firstly and then use its concept to describe the meaning of word \( t_j \). While the latter concept doesn’t involve the word \( t_j \) as a leaf in first layer, but reaches the leaves through several intermediate nodes and stories. This option more corresponds with the dependency of comprehension.

### 3.2 Online Concept Representation

As discussed above, OCT builds a structured tree for comprehension of concept. Each layer in the tree is the deeper comprehension for next higher layer, and the concept along the topology of OCT is always correlated with a clear objective (in figure 1, the objective is the term \( t_j \)). Therefore OCT can simulate the way of human thinking. Online conceptual model simply named OCNM is a statistical model to describe the probability distribution of concept. The space of concepts is constructed by the stories in first layer of OCT, and the weight of each word in OCM can be approximated as follow:

\[
P(w|N_{t_j^{-tcore}}) = \sum_{S\in N_{t_j^{-tcore}}} P(w|S)P(S|ONT) \tag{1}
\]

Here, \( t_j^{-tcore} \) represent the root of OCT and word \( t_j \) and each leaf is \( t_{core} \). \( N \) indicates the set of stories in first layer of OCT and \( S \) indicates a story in \( N \). And \( P(w|S) \) represents the weight of word \( w \) in story \( S \):

\[
P(w|S) = \lambda \frac{tf_wS}{|S|} + (1-\lambda) \frac{cf_w}{coll.size} \tag{2}
\]

Here, \( tf_wS \) is the frequency the word \( w \) occurs in the story \( S \), \( cf_w \) is the total number of times \( w \) occurs in a large background collection, and the \( coll.size \) is the total number of words in that background collection. The value of \( \lambda \) as well as the number of stories to include in first layer of OCT are trained in TDT3, which is reported in the section of experiment.

\[
P(S|ONT) = \sum_{w\in S} \left[ \prod_{i=2}^{n} \frac{\log \left( link_{i}(w) + 1 \right)}{\log \left( link_{i.size} \right) \cdot \text{depth}_i} \right]^{n} \tag{3}
\]

Additionally, \( P(S|ONT) \) in (1) describes the degree of correctness that story \( S \) serves as the concept of word \( w \) in \( S_{core} \) as (3).

Here, \( link(w) \) is the number of leaves in the \( i \)-th layer of subtree defined \( w \) as its root, \( link.size \) is the total number of leaves in the \( i \)-th layer of OCT, \( n \) is the whole depth of OCT, and \( \text{depth}_i \) is depth of the \( i \)-th layer:

\[
\text{depth}_i = \text{layer}_i - 1 \tag{4}
\]

\( \text{Depth.size} \) describes the most distant of topology from \( t_j \) to \( t_{core} \). We trained its value in TDT3 whose values are reported below.

### 4. OCM FOR LINK DETECTION

OCM extracts all contexts from corpora for each word, and use these contexts to explain the concepts of terms in the story being processed. Additionally OCM describes the correlation between concepts and the story being processed by depth and extent of leaves in topology of OCT. In TDT’s link detection task two stories are being compared to decide whether they are on the same topic. Therefore we build topic model for each story based on OCM, and measure the similarity between the two stories directly through their topic models. There are three steps to build topic model:

- Select core word \( t_{core} \) from current story and combine \( t_{core} \) with each context in this story into 2-tuple \( (t_j, t_{core}) \) respectively.
- Build OCT for each 2-tuple \( (t_j, t_{core}) \) and calculate its OCM.
- Combine all the OCM of 2-tuple \( (t_j, t_{core}) \) to build topic model of current story.

Once the two topic models are created, we decide whether they discuss the same topic by comparing the models directly. We found it helpful to use a modified form of the Kullback-Leibler divergence.

### 4.1 Core Word Selection

As the discussion above, a topic model of a story is built by combining all the OCM of 2-tuple \( (t_j, t_{core}) \). In this procedure we extract a word as the core word of the story and select some words whose weight are highest ranked as its contexts. Each 2-tuple \( (t_j, t_{core}) \) is composed of the core word and one of its contexts. This way to create 2-tuple terms can reduce the complexity of space and time greatly. And more important reason is that the main content of a story usually can be summarized by the core word, and every component of the story can be described around it. For example, the two stories whose topic respectively is “9/11 Attack” and “Bali Massacre” both can be summarized by a core word “terrorism”.

In TDT, the corpus mainly consists of textual news stories. And most topics of the stories usually are abrupt. Therefore we hope the core word selected not only can represent the core content adequately, but can be novel relative to other words. So we select the core word based on follow two conditions:

- Core word should occur in current story frequently.
- Core word occurs frequently during a specific period and rarely outside of this period.

As the discussion above, we select a word highest weighted by \( \text{weight}(t|S) \) as the core word, which formula is (5). In TDT4, a \textit{window} of time is allowed for Link Detection system to estimate parameters. And the \textit{window} starts from Oct 2000 and terminates...
ten files after the file the story being processed locates in. Additionally we set a period shorter than the window, over which a topic occurs frequently. Here, win.size is the total number of stories over the period, \(t_{in}\) is the frequency of term \(t\) occurring over that period, \(L\) is the total number of stories in the window, and \(t_{out}\) is the frequency of term \(t\) occurring outside of the period. We trained the period in TDT4, whose value will be reported in section 5.

\[
\text{weight}(t \mid S) = \frac{tf_w \cdot \log L - \text{win.size}}{tf_{out} \cdot \text{win.size}} \quad (5)
\]

### 4.2 Building Topic Model

In order to build a topic model for a story, we start with the process described above. Firstly, we select a core word and its contexts from current story. Secondly we build OCT for each 2-tuple term \((t_j, t_{core})\), and calculate its OCM based on the tree. At last, we combine all the OCM of 2-tuple terms \((t_j, t_{core})\) as the topic model of current story:

\[
P(w \mid ONM) = \sum_j P(w \mid N_{t_j-t_{core}})P(t_j \mid S_{cur}) \quad (6)
\]

Here, \(N_{t_j-t_{core}}\) represent the first layer of OCT, \(t_1\) is one context of the core word \(t_{core}\). \(P(w \mid N_{t_j-t_{core}})\) describes the probability distribution of \(w\) in \(t_j\) calculated by (1). \(P(t_j \mid S_{cur})\) represents the weight of \(t_j\) in current story. In this paper, we only select 30 words whose weight is highest as the contexts of \(t_{core}\).

### 4.3 Measuring Topic Similarity

Once we have built the topic models for each story, we need to compare the two models to determine the chance that they discuss the same topic. Given two stories, \(S_1\) and \(S_2\), assume that their online conceptual models are \(O_1\) and \(O_2\), respectively. If we were measuring the similarity between query and information in IR, or profile and information in IF, we might use OCM to calculate either \(P(S_1 \mid O_2)\) or \(P(S_2 \mid O_1)\) or possibly the average of the two according to the theory of Bayes.

However, given that the two models were estimated from similar amounts of data \((S_1\) or \(S_2)\), we can instead compare the models directly. The Kullback-Leibler (KL) divergence is a standard way to compare two probability distributions, defined as:

\[
D(O_1 \parallel O_2) = \sum_w P(w \mid O_1) \log \frac{P(w \mid O_1)}{P(w \mid O_2)} \quad (7)
\]

KL divergence is asymmetric, which is unacceptable as a link detection metric. We counteract the asymmetry by summing the divergence in both directions: \(D(O_1 \parallel O_2) + D(O_2 \parallel O_1)\). Additionally KL divergence is a measure of dissimilarity of two distribution, we use the negation of the above quantity to measure similarity. This yields a reasonable approach, but has the problem that if the models are very ambiguous-e.g., if OCM created a model in which the distribution of terms looks too much like general language of source data, their matching will have very little significance. To address this problem, we leverage a notion of story clarity [3], the KL divergence between a distribution and general language. A distribution is clear if it is very different with general language, otherwise unclear. Therefore, the non-symmetric negation formula

\[-D(O_1 \parallel O_2)\] is changed to \(-D(O_1 \parallel O_2) + \text{Clarity}(O_1)\). That is, the degree to which \(O_1\) and \(O_2\) are similar, increased to the extent that \(O_1\) is a clear model that differs from general language. After a simple algebraic manipulation, the formula can be described as:

\[
D(O_1 \parallel O_2)_{\text{Clarity}} = \sum w P(w \mid O_1) \log \frac{P(w \mid O_1)}{P(w \mid GE)} \quad (8)
\]

It has been demonstrated that the clarity-adjusted symmetric KL divergence achieved better performance than other version of KL divergence. Therefore we compare topic model of two stories in our experiment by calculating the same quantity for \(O_1\) and \(O_2\) respectively and adding them together:

\[
\text{KL}(O_1, O_2) = D(O_1 \parallel O_2)_{\text{Clarity}} + D(O_2 \parallel O_1)_{\text{Clarity}} \quad (9)
\]

### 5. EXPERIMENT

In this section, we introduce our experiments built for LDT. Firstly we introduce the corpus of TDT4 and the subset of it used to evaluate LDT system. Then we describe the evaluation methodology of TDT2003. Finally we introduce the experimental setup.

#### 5.1 Corpus

The TDT4 corpus spans Oct 2000-Jan 2001 and consists of English, Mandarin and Arabic data. There are 40 topics annotated for the corpus. Each story TDT4 corpora is taggged according to whether it discusses each of the defined topics. These story-topic tags assume a value of YES if the story discusses the target topic, BRIEF if that discussion comprises less than 10% of the stories, or otherwise NO if the story does not discuss the topic.

<table>
<thead>
<tr>
<th>Table 1. Input data alternatives of types for LDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Data Types</td>
</tr>
<tr>
<td>text sources and manual transcription of the audio source</td>
</tr>
<tr>
<td>text sources and ASR transcription of the audio sources</td>
</tr>
<tr>
<td>text sources and the sampled data signal for audio sources</td>
</tr>
</tbody>
</table>

There will be three alternative choices, shown in Table 1, for the form of the source data to be processed. In our experiments, we use the first type of data in the table as the input of LDT system. Another important issue of experiment is the choice of languages to be processed. As mentioned above, TDT4 provides three source data of English, Mandarin and Arabic for LDT. And evaluation of LDT can be conditioned on the source language: cross-language and same-language story pairs. We will evaluate our system only based on the condition of mandarin story pairs. For the evaluation of LDT in TDT2003, LDC provides the index for the stream of mandarin data, which includes 26,065 story pairs from the text sources and ASR transcription of the audio sources. And in this corpus there are 3,075 story pairs are assumed the YES, others assumed NO. In our experiment, we use 1,000 story pairs assumed YES and 6,000 story pairs assumed NO to build training corpus, and the others to build testing corpus.

#### 5.2 Evaluation

TDT tasks are evaluated as detection tasks. For each test trial, the system attempts to make a yes/no decision. In story link detection, the decision is whether the two members of a story pair belong to the same topic. In all tasks, performance is summarized in two
ways: a detection cost function \( C_{\text{Det}} \) and a decision error tradeoff curve [18]. Both are based on the rates of two kinds of errors: misses, in which the system gives a no answer where the correct answer is yes, and false alarms, in which the system gives a yes answer where the correct answer is no.

\( C_{\text{Det}} \) combines the two error probabilities into a single detection cost by assigning costs to each of them:

\[
C_{\text{Det}} = C_{\text{Miss}} P_{\text{Miss}} P_{\text{Target}} + C_{\text{FA}} P_{\text{FA}} P_{\text{non-target}} \tag{10}
\]

Here, \( C_{\text{Miss}} \) and \( C_{\text{FA}} \) are the costs of a Miss and a False Alarm, respectively, \( P_{\text{Miss}} \) and \( P_{\text{FA}} \) are the conditional probabilities of a Miss and a False Alarm, respectively, and \( P_{\text{Target}} \) and \( P_{\text{non-target}} \) are the prior target probabilities (\( P_{\text{non-target}} = 1 - P_{\text{Target}} \)). The evaluation cost parameters to be used for the TDT2003 evaluation are given in Table 2. Additionally \( C_{\text{Det}} \) can be normalized and averaged over topics:

\[
(C_{\text{Det}})_{\text{Norm}} = \min(C_{\text{Miss}} P_{\text{Target}}, C_{\text{FA}} P_{\text{non-target}}) \quad \tag{11}
\]

Thus the absolute value of \( (C_{\text{Det}})_{\text{Norm}} \) is a direct measure of the value (i.e., relative cost) of the TDT system. This measure can help us separate the performance on the task from the choice of yes/no threshold.

**Table 2. Evaluation cost parameters used in TDT2003**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_{\text{Target}} )</td>
<td>0.02</td>
</tr>
<tr>
<td>( C_{\text{Det}} )</td>
<td>1.0</td>
</tr>
<tr>
<td>( C_{\text{FA}} )</td>
<td>0.1</td>
</tr>
</tbody>
</table>

The detection error tradeoff curve is drawn by connecting all the point of \( C_{\text{Det}} \) calculated based on different threshold. Because \( C_{\text{Det}} \) measures the rates of misses and false alarms which is lower the performance of system is better, the tradeoff curve nearer lower sinister corner of the coordinate plane will illustrate better performance for a system at all threshold. The minimum value found on the curve is known as the \( \min (C_{\text{Det}}) \).

While link detection systems are expected to be capable of making link decisions for all pairs of stories, evaluation of all such decisions is neither practical nor necessary. Therefore evaluation can be limited to a subset of story pairs sufficient to provide reliable estimates of \( P_{\text{Miss}} \) and \( P_{\text{FA}} \). This will keep system output files to a manageable level.

### 5.3 Experiment Setup

Our LDT system processes the input source data in chronological order. For each pair of stories, we build OCT for each of them and estimate their maximum likelihood OCM based on the OCT. Then we use symmetric clarity-adjusted KL to measure the divergence between their topics. After calculating the degree of relevance for all source data, we determine which story pairs is related to a topic using different thresholds and evaluate the performance of our system using \( C_{\text{Det}} \) and \( (C_{\text{Det}})_{\text{Norm}} \). However, the link detection system may defer its identification of story links until a limited amount of subsequent source data is processed. This deferral period is a primary task parameter and is the number of source files, including the source file being processed, for which processing may be completed before making story-story link decisions. (The greater the deferral, presumably the better will be the link decisions.) The deferral parameter values to be used in our experiment is 10 source files, included the one being processed.

Before evaluating our LDT system, we need train three main parameters: whole depth \( n \) of OCT in (3), the number of stories \( U \) in first layer to be used in (1) and period in (5). The whole depth \( n \) indicates the maximum steps through which a concept of root in OCT correlates the core word of the story being processed. We evaluate the performances of \( n \) from 1 to 15 to search the best choice. The set of stories \( N \) indicates how many stories in first layer of OCT can be used to construct OCM. In the procedure of training, we rank the stories using their \( P(S|OCT) \) and select the highest \( N \) stories to construct OCM. And we measure the performance for a system at all threshold. The minimum value found on the curve is known as the \( \min (C_{\text{Det}}) \).

In TDT4 the story pairs are listed in order of time, and saved in different files. And the window given by TDT2003 Evaluation Plan stars from Oct 2000 and terminates ten files after the file the story being processed locates in. Additionally the topic of news is abrupt, and the time the topic reported concentrate in a short period densely [19]. Therefore we trained the period from 1 file to 10 files around the file current story located in. Through training, we set the period equal to 2 files. Additionally we refer to the value of smoothing parameter \( \lambda = 0.9 \) by which LM obtains its best performance [16].

### 6. RESULT AND DISCUSSION

The most important aspect of our experiment is to train the parameters of \( U \) and \( n \). As mentioned above, \( U \) is the number of stories in first layer used to build OCM for the root of OCT, and \( n \) is the whole depth of the topology of OCT. In experiment, we select \( U \) stories highest ranked by \( P(S|OCT) \) from first layer to describe the online concept of the root in current story being processed. We hope these stories can expand the term space of the story being processed, at same time restrained by \( U \) have low possibility that noises are inserted into the space. Therefore we train the value of \( U \) from 1 to 30 to explore how many stories in first layer can make LDT obtain the best performance. Besides, the whole depth \( n \) of the topology of OCT is another factor influencing performance of LDT. As the discussion in section 3, the depth of topology indicates the strength a concept of the root in OCT correlates with the core word of the story being processed: the lower the depth is the more the concept is related to the topic of the story being processed. But if the whole depth \( n \) is too low, there will be not enough paths between a concept and the leave (core word in the story being processed) in OCT can be used to describe their strength correctly. We evaluate the performance of the whole depth \( n \) from 1 to 15 in the stage of training.

We present the influence of \( U \) and \( n \) to the performance of LDT in Figure 2 and Figure 3. The former shows the minimum DET cost obtained on different \( n \) from 1 to 12, and the latter from 12 to 15. Both of them shows the performance obtained based on different \( U \) from 1 to 30. The result in the figures demonstrates our hypothesis that performance of OCM is greatly influenced by the parameters of \( U \) and \( n \). When the whole depth \( n \) changes among lower values from 1 to 8, the performance is worst and changes very little. As discussion above, at this moment the
topology still doesn’t include enough connection between concept and core word. Therefore the OCM constructed by this topology can’t describe the relevance between each concept and the topic of the story being processed faithfully. Along with the increase of \( n \), the performance of OCM gets better progressively. But when \( n \) exceeds 12, the performance gets worse again. The cause of this phenomenon is general distribution of paths between root and leaves in high level topology of OCT. In fact, there is a great number of overlapping intermediate nodes to same layer of different concept topology in an OCT, and the number grows rapidly with the level of layer. Additionally, the topology of an intermediate will not change wherever it is. Therefore the topology of high level layer in OCT becomes very similar, which result in general distribution of paths among different concepts. A more intuitive demonstration of this phenomenon is shown in Figure 4, which represents the trend of the minimum DET cost our system obtained among different layer. We can see the performance is best at layer 12.

Figure 2. Performance changes accompanying changes in \( U \) among different \( n \), evaluated from layer 2 to layer 12

Figure 3. Performance changes accompanying changes in \( U \) among \( n \), evaluated from layer 12 to 15

Figure 2 and Figure 3 also represent the influence of \( U \) to the performance of OCM. We found the influence isn’t obvious at low level layers. As the discussion above, the reason is topology used to estimate the strength of connection in low level layer is uncompleted. When \( n \) increases to about 11, relative few stories to be used in first level can make OCM achieve better performance (\( U \) is about 4). It means the concept of root described by about 4 highest ranked stories can efficiently represent the meaning of it in current story being processed. And if any more stories are used to describe the concept, the number of noises in it will be increased. But with the increase of \( n \) from 12, it is better to use more stories to construct OCM of the story being processed (\( U \) is more than 30). The reason of this phenomenon is the general distribution of paths in high level topology of OCT mentioned above, which make the likelihood very similar that each story in first layer correlates to the core word. Therefore OCM treats all of these stories as one concept, although some of them describe other concepts. It will result in OCM persistently expand the scale of the stories in the highest ranked concept to counteract the effectiveness of general distribution of paths. Figure 5 represents the average number of stories in highest ranked concept changes with \( n \).

Figure 4. Comparison of Minimum Cost of DET for LDT among different layer

Figure 5. Number of stories in highest ranked concept

Figure 6 and Figure 7 represent the performance of each \( n \) with best \( U \). As discussion above, the number of stories in highest ranked concept usually exceeds 30 when \( n \) is bigger than 12. But in the experiment we set the scale of stories in concept less than 30. Therefore the DET curve for each \( n \) bigger than 12 is generated based on \( U=30 \). From the two figure, we can found the best performance is obtained when \( n \) is equal to 12 and \( U \) is equal to 30. But we think it is not a good choice, because the number of stories in highest ranked concept is so many which is equal to 30. And from figure 2 and figure 3, we have found the truth that \( n=12 \) represented the trend that OCM counteract the effectiveness of general distribution of paths through expanding the scale of concept space persistently. Therefore we select \( n=11 \) with \( U=5 \) as
the best performance whose DET curve is enough similar with n=12, and it didn’t represent the trend of counteraction in figure 2.

Figure 6. DET curve for TDT4 LDT with best \( U \) among different layers, evaluated from layer 2 to layer 12

Figure 7. DET curve for TDT4 LDT with best \( U \) among different layers, evaluated from layer 12 to layer 15

Based on the value of parameters trained above, we compare the performance between RM and OCM. Figure 8 represents the result of evaluation. We found OCM achieves great improvement: the minimum DET cost is 0.0099 which exceed the performance of RM 0.0187 about 50 percent. Then we using same parameters evaluate the performance of them in greater test corpus. Figure 9 represents the result of testing. As we see, the improvement is great, although a little below that in training, which is 0.0177 better than the performance of RM 0.0193.

Figure 8. DET curve for TDT4 LDT with training story pairs, compared between best relevance model and best OCM

However we found that the performance of RM and OCM both are worse comparing with their performance reported in previous papers. The main reason lies in the selection of corpus and the defect of our word segmentation system (WSS). As mentioned in section 5, we only select mandarin data in TDT4 as our corpus, while unlike the English data, our WSS is not enough good for
mandarin data. Many words obtained by this system can’t catch the sense in current stories being processed. It results in great loss to the performance of LDT system in an early stage of our experiment. Additionally a characteristic of TDT corpus is the data mostly comes from news, which includes lots of novel words. But most WSS deal with data based on dictionary or large scale of obsolete corpus. Therefore the novel word in realtime stories of news usually can’t be organized, although they could be very significant. Towards these defects, our next work will focus on the detection of novel words. As we know, the ratio is very high that novel word accruing in the stories about new events, and the frequency of novel word changes abruptly.

7. CONCLUSION

LDT is widely used in most task of TDT, which requires determining whether a pair of stories discuss the same topic or not. LDT also can be used as the underlying research of information retrieval, information filtering and data mining, because it is an effective tool to measure the relevance of information. Additionally LDT is a platform to evaluate the performance of NLP technologies. Therefore it is significant to further improve the performance of present LDT techniques.

In this paper, we introduce a model named OCM to explore concepts of words and estimate their correlation with topics. OCM describes the strength of connection between concept and core word based on the topology in OCT and build topic model for each story by combining the distribution of concepts among different words. The relevance of two stories is calculated through comparing their topic models directly. By experiment we confirm that OCM achieves a great improvement relative to RM. And it demonstrates that comparison of concepts do a better job than measure of overlapping words between two stories. But because of the defect of our word segmentation system, especially segmenting words without consideration of novel words, the performances of RM and OCM both are worse than those reported in previous research. In future work, we will attempt to combine OCM with the underlying techniques of New Event Detection (NET), especially the detection of novel words. By using the former to improve the ability of recognizing new events and using the latter to improve the performance of comparison among stories, we will build an adaptive learning model to further improve the intelligence of our detection system.

8. REFERENCES