UMass at TDT 2004

Margaret Connell, Ao Feng, Giridhar Kumaran, Hema Raghavan, Chirag Shah, James Allan
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{connell,aofeng,giridhar,hema,chirag,allan}@cs.umass.edu

1. INTRODUCTION

The Center for Intelligent Information Retrieval at UMass Amherst submitted runs for all four tasks, namely, Hierarchical Topic Detection, Topic Tracking, New Event Detection and Link Detection. In this paper, we describe our models, experiments during training and our results on all the four tasks.

2. HIERARCHICAL TOPIC DETECTION

This task replaces Topic Detection in previous TDT evaluations. Since it is the first year we have HTD, there are not any existing results to compare to. We used the vector space model as the baseline, did bounded clustering to reduce time complexity and had some simple parameter tuning.

2.1 Model description

Topic Detection classifies stories into different topics, but HTD requires more than that. Is there any other entities between a story and a topic? [10] views a topic as a structure of inter-related events, which gives us a good hint for this new task.

Experiments in [10] show that time locality is a very useful attribute in event organization, and it can also help to solve the complexity problem in TDT2004. The TDT-5 collection contains 407,503 stories in three different languages, and the running time for traditional clustering algorithms, which take $\Omega(n^2)$, is not acceptable for such a huge collection. Since we know that stories in the same event tend to be close in time, we only need to compare a story to its "local" stories instead of the whole collection.

The algorithm we use has two steps, bounded 1-NN for event formation and bounded agglomerative clustering for building the hierarchy. In the first step, all stories in the same original language and from the same source are taken out and time ordered. Stories are processed one by one and each incoming story is compared to a certain number of stories before it. This number is approximately the number of stories in a token file and the value is 100 for the baseline run. If the similarity (cosine similarity of tf-idf term vectors) of the current story and the most similar previous story is larger than a given threshold (0.3 in the baseline run), the current story will be assigned to the event that the most similar previous story belongs to. Otherwise, a new event is created.

After the first step, there is a list of events for each source/language class. The events within each class are sorted by time order according to the time stamp of the first story. Then we do a bounded agglomerative clustering for the events, which works as following.

Take a certain number of events (the number is called WSIZE, default value 120) from the sorted events list. At each iteration, find the closest event pair and combine the later event to the earlier one. This process continues for (BRANCH-1)WSIZE/BRANCH iterations so the number of clusters left is WSIZE/BRANCH. Take the first half out and get WSIZE/2 new events and agglomeratively cluster until WSIZE/BRANCH clusters are left. The process continues before all events are clustered. Then number of clusters left will be one BRANCH-th of the events number. [1] shows how to get the optimal branching factor. For the evaluation cost parameters this year, the optimal value is around 3, so we take BRANCH=3 in the baseline run.

Now we have a list of clusters, and they can be clustered again using the bounded agglomerative clustering algorithm shown above. This process continues until the number of clusters is smaller than a given value, which is proportional to the square root of the number of stories in that class. Then all clusters in the same language but from difference sources are combined, sorted by time order and a single round of clustering is performed. Finally clusters from all languages are mixed and clustered until only one cluster is left, which becomes the root node of the hierarchy.In the multi-lingual experiment, we used the machine translation for Arabic and Mandarin stories to simplify the similarity calculation.

There are some clusters each containing only one story. Since stories in non-leaf nodes are allowed in the evaluation, such a cluster is replaced by the corresponding story, which is beneficial for the travel cost.

2.2 Training

The TDT-4 collection is the closest to TDT-5 in time, so we use it as the training corpus. TDT-4 contains both newswire and broadcast stories, while TDT-5 has only newswire. In order to make them comparable, we take only the newswire stories from the TDT-4 corpus, which includes NYT, APW, ANN, ALH, AFP, ZBN and XIN.

A lot of parameters were tuned in the training set.

- BRANCH: the average branching factor in the bounded agglomerative clustering algorithm. It determines the height of the whole hierarchy.
- Threshold: the threshold in event formation to decide if a new event will be created.
- STOP: the agglomerative clustering algorithm in each source stops when the number of clusters is smaller than $STOP\sqrt{\#story}$.

It controls when to mix the clusters from different sources.

- WSIZE: the maximal window size in agglomerative clustering. It is used to reduce the time complexity or the agglomerative clustering will take forever.
- NSTORY: Each story will be compared to at most NSTORY stories before it in the 1-NN event clustering. This idea comes from the time locality in event threading.
- SIM: the similarity function when comparing two term vectors.

Preliminary experiments show that cluster sizes vary a lot using this algorithm. Among the clusters very close to the root node, some contains thousands of stories while a lot of clusters are singletons. Both 1-NN and agglomerative clustering algorithms favor large clusters. To make the hierarchy more balanced, we modified the similarity calculation to give smaller clusters more chances.

$$sim(cluster1, cluster2) = \frac{sim(v1, v2)}{|cluster1| + a} \quad (1)$$

Here $sim(v1, v2)$ is the similarity of the cluster centroids, $|cluster1|$ is the number of stories in the first cluster (events are time ordered according to the first story), and a is a constant to control how much favor smaller clusters can get. a is also one of the parameters tuned in the training set.

2.3 Results

This year we submitted three runs for each condition: UMASSv1, UMASSv12 and UMASSv19. UMASSv1 is our baseline run, and the parameters in this run are shown below.

KNN bound The number of previous stories compared to each incoming story in the KNN algorithm. Default value is 100.

SIM Similarity function. Cosine similarity of cluster centroid (or simply story vector for story comparison).

WEIGH Vector weighing scheme. Here we use idf, $IDF = log((N + 0.5)/DF)/log(N + 1)$, where N is the number of stories in the collection, and DF is the number of stories the term appears in.

THRESH Threshold for KNN. Default 0.30.

WSIZE Maximal number of clusters in agglomerative clustering. It is 120 in the baseline run.

BRANCH Average branching factor. We take 3, as it is the optimal value with the evaluation parameters.

STOP The constant which decides when clusters from different sources are mixed. It is 5 here.

UMASSv12 is very similar to UMASSv1 except that the similarity function is normalized by the cluster size(see equation 1, $a = 0$). And the only difference between UMASSv19 and UMASSv12 is that WSIZE increases to 240 in UMASS19. Table 1 shows the performance of the three runs in TDT-4, and Table 2 contains the results in the official TDT-2004 submission.

We do not use the normalized travel cost because there are two different versions of normalization schemes. The normalization factor of the old scheme is the expected travel cost to reach a leaf node in a minimal spanning tertiary tree. And the new scheme assumes a 1-level DAG.

system output	TDT-4 mul,eng C_{det}	C_{travel}
UMASSv1	0.6522	226.47
UMASSv12	0.5344	38.98
UMASSv19	0.4761	37.61

Table 1: UMASS HTD trained in TDT-4

system output	TDT-5 eng,nat C_{det}	C_{travel}	TDT-5 mul,eng C_{det}	C_{travel}
UMASSv1	0.4412	206.01	0.3287	166.35
UMASSv12	0.3822	59.51	0.3175	109.57
UMASSv19	0.3204	56.00	0.2910	176.27

Table 2: UMASS HTD in TDT-2004

The old one in the evaluation plan is a very strict scheme. If a system does not pay enough attention to the branching factor, it will be penalized a lot. Like CHUK and ICT, they will get normalized travel cost of 15 and 30 respectively in the mul,eng run, if we use the old normalization scheme. However, TNO did a very good job. Its normalization cost is still around 1 even with the old normalization.

The new one is much easier to achieve. But it is so easy that any normal hierarchy can beat it and the normalized travel cost does not make much sense. Even if you have very large travel cost like 2000, it is still below 0.1 after normalization. Since travel cost is much smaller than detection cost, it will encourage everybody to focus on detection cost. While overlapping clusters is allowed, it will probably lead to highly overlapping clusters. We will get the power set when coming to the extreme case. I do not think it is the result we want to see when starting HTD.

So, in my opinion, I prefer the old normalization scheme. But some changes are needed to make detection cost and travel cost comparable. First, the normalized travel cost (using the old scheme) is much larger than detection cost, so we can set a smaller weight for it, like 0.1 or even smaller. Second, binary tree wins this year with the maximal height over 40, but most real systems do have such a deep hierarchy. I suggest we increase CTITLE and decrease CBRANCH to push the optimal branching factor to about 5.

UMass did not do very well in this year's evaluation, mainly because of the high detection cost. These factors may be useful to improve the performance.

- Small branching factor can reduce both detection cost and travel cost. With smaller branching factor, we get more clusters with different granularities, and have more chance to match the topic. And the optimal branching factor of 3 shows the advantage of small branching factor in reducing travel cost. Experiments with limited branching factor proved this assumption.
- Assigning one story to multiple clusters can decrease miss rate dramatically. TNO observed it in their experiments.
- The assumption of temporal locality is useful in event threading [10], but it does not seem to work for topics. More experiments after the submission show larger window size can improve performance.
- The source-language-all clustering hierarchy assumes stories from the same source are more likely to be in the same

topic, but it makes the result worse. Removing this restriction shows some improvement over the submitted runs.

In this year's submission, TNO is much better than other participants on both detection cost and travel cost. And it is the only participant that allows one story to be assigned to more than one clusters, which may be very useful to reduce the detection cost. Its binary tree may be another reason for the small detection cost.

3. TOPIC TRACKING

3.1 Creating a training Newswire corpus

We created a subset of the TDT4 corpus that emulates the characteristics of the TDT5 corpus as close as possible. It was made known to the participants *a priori* that 50% of the topics in TDT5 had English only stories in the test set and the rest of the topics had multi-lingual stories. To emulate these statistics, we first sorted the TDT4 topics by the number of test stories from English sources. We collected top 15 topics that had the highest number of English test-stories as our English only topics. We removed all non-English stories from these topics. From the remaining topics in TDT4, we sampled 15 topics as follows. For each topic, we estimated the standard deviation in the number of stories from each language and chose the top 15 ones that had the least standard deviation and also that had a minimum number of stories from each language. These topics are expected to contain nearly equal proportion of test-stories from all languages and hence are expected to imitate the multi-lingual topics in the test corpus. Here, the idea was that given no information on the language specific distribution of stories in the multi-lingual topics of TDT5 corpus, it is safest to assume uniform distribution. In all, we formed 30 training topics, 15 of which are English-only topics and the rest, multi-lingual.

3.2 Unsupervised Tracking

In the topic tracking task a small number, Nt, of training stories is given for a particular topic. The topic is represented by a centroid, which is an average of the vector representatives of the training stories. Incoming stories are compared to the centroid for the topic. If the cosine similarity of the story to the centroid exceeds the YES/NO threshold, then the story is considered as "on-topic", otherwise "off-topic". As in the linking task, different thresholds were chosen for each condition. They were determined by experiments run on the newswire subsample of the TDT4 corpus as explained in the previous section or on the TDT-3 training data.

For the TDT 2004 tracking task, we submitted five runs for the required conditions ($NT = 1$). All tracking runs used a deferral of one. Our tracking system had five parameters: the model, the translation condition, the number of terms in the vector, the YES/NO threshold value and whether or not adaptation was carried out.

Like Story Link detection we experimented with two models – the traditional vector space model, and relevance models. Adaptation allows the addition of the incoming story to the topic representation and recomputes the topic centroid. Adaptation uses an additional threshold which determines whether a new story is similar enough to the centroid to be added to the topic. A new centroid is computed from the story vectors each time a new story is added to the topic. For adaptation the number of terms added to the model when it is updated is a variable parameter.

Since stories exist in English, Arabic and mandarin, and there are English translations for the English and mandarin, when we compare two Chinese stories or two Arabic stories, we have the means to compare them in their native language or in their translated form. The translation condition indicates whether the comparisons are made in English or in the native language.

We submitted the following runs:

1. **TF-IDF**(UMASS-4): This method uses a vector space model. Cross-language similarities were calculated using the provided English translations and the incrementally updated corpus statistics. The vector length was constrained to 1000 terms. Centroids were not adapted. The YES/NO threshold was 0.06. The threshold were trained on a specially sampled portion of the TDT4 corpus as explained in section 3.1.

2. **TFIDF + adaptation** (UMASS-1): This method was similar to the baseline case described above, but included adaptation. We have seen in experiments with TDT-3 that adaptation can degrade performance when the vector has as many as 1000 terms. This happens because adaptation may allow the inclusion of new stories that stray from the topic and include confusing terms. We therefore limited the number of terms in the centroid to the top 100 highest weighted terms rather than 1000 terms. At the maximum, 100 stories could be added to the centroid vector. For the required conditions (manual transcription and $Nt = 1$) the adapting threshold was 0.4 and the YES/NO threshold was 0.06. These numbers were obtained by training on the training data obtained as explained in section 3.1.

3. **TF-IDF + Language specific comparisons + adaptation:** (UMASS-2)

   In this run we chose to make comparisons of stories in their native language whenever possible. However as the training stories were English, at the start, as stories arrived they had to be compared with the English training stories. When a Chinese story or an Arabic story that was highly similar to the centroid vector (English) arrives it was used to seed a Chinese or Arabic centroid vector as is explained below.

   For a given training story (i.e. $Nt = 1$) the system created two centroid vectors, one centroid for the whole collection, called a global centroid and another for the native English stories. If an incoming story was native English then the system calculated the similarity with the English centroid, otherwise the system calculated the similarity with the global centroid. Initially there was no centroid for native Arabic or Mandarin stories. To determine whether to start a new Arabic or Mandarin centroid for a topic, we use the the global threshold for that topic. That threshold is the average similarity over all pairs of stories in the global centroid. If the global centroid only has one story, the threshold is set to 0.5, based on experiments with TDT-3 data. If the similarity between an incoming story and the global centroid is greater than this threshold then the incoming story becomes the new native centroid. Once a native topic centroids have been established, the adaptation threshold is computed to be the average similarity over all pairs of stories in the native centroid, or .5 if there is only one. The threshold thus changes whenever a story is added to a centroid.

   The incoming native English stories were compared in English and the $tf \times IDF$ statistics were incrementally updated using stories taken from native English sources. In this case the last story considered in the update was the incoming story. Only the top weighted 100 terms of a vector were considered and a maximum of 100 stories could be added to the centroid vector.

For the required conditions the YES/NO threshold was 0.06 and the adaptation threshold was initialized to 0.5. These thresholds were trained on the specially sampled portion of the TDT4 corpus (refer section 3.1. We also submitted a run with a YES/NO threshold of 0.084, which was our system from last year.

4. **Relevance Models + adaptation** (UMASS-5)

   Our application of Relevance Models to topic tracking was somewhat different. We wanted to take advantage of the streaming nature of the tracking task and re-cast Relevance Models in terms of unsupervised topic adaptation. Let T denote the topic being tracked and let S be a subsequent story in the stream. The similarity of of S to the topic is measured as the asymmetric clarity-adjusted divergence of their language models:

$$S_{trk}(T,S) = \sum_{w} P(w|T) \log \frac{P(w|S)}{P(w|GE)} \qquad (2)$$

   Here $P(w|GE)$ is the background probability of w, $P(w|S)$ is the language model of the story and $P(w|T)$ refers to the Relevance Model of the topic T. Because of computational constraints we chose not to estimate a Relevance Model for $P(w|S)$, instead using smoothed maximum-likelihood (ML) estimates. $P(w|T)$ on the other hand was constructed incrementally in the following fashion. We start by setting $P_0(M_d) = \frac{1}{N_t}$ if d is one of the N_t training documents and zero otherwise. We initialize $P(w|T)$ according to equation (8) but using $P_0(M_d)$ instead of $P(M_d|Q)$. Then, we start tracking, and if the similarity $S_{trk}(T,S)$ of some story S exceeds a pre-defined threshold θ_{ad}, we update the topic model. The update involves setting $P_{n_{ad}}(M_d) = \frac{1}{N_t+n_{ad}}$ if d is one of the original N_t training stories, or one of the n_{ad} stories which scored above the adaptation threshold θ_{ad}. We keep $P_{n_{ad}}(M_d)$ at zero for all other documents and recompute $P(w|T)$ using equation (8) with $P_{n_{ad}}(M_d)$. We do not allow more than $n_{ad} = 100$ adaptations.

The above systems are enhancements on several years of TDT tracking research ([9],[2])

3.2.1 Results

The graph in figure 1 shows the det curves for UMASS-2, UMASS-4 and UMASS-5 on the training corpus which consists of Newswire stories sampled as in section 3.1. We find little improvement due to adaptation in the TF-IDF model. The det curve for relevance model is in general better, but this is not reflected much in the min cost.

3.3 Supervised Tracking

In supervised tracking, the assumption is that you are given N_t stories as training stories and the test stories arrive in a stream. For each test story the system has to output a confidence score and a decision just as in unsupervised tracking. However, for every story that is delivered the assumption is that the user reads it immediately and provides a relevance judgment immediately.

For evaluation, in addition to being evaluated on Min-cost we were to be evaluated on the T11SU metric from the TREC filtering runs. For supervised tracking runs we used a lot of approaches from unsupervised tracking, with certain modifications based on the fact that relevance judgments are obtained for every document that is delivered.

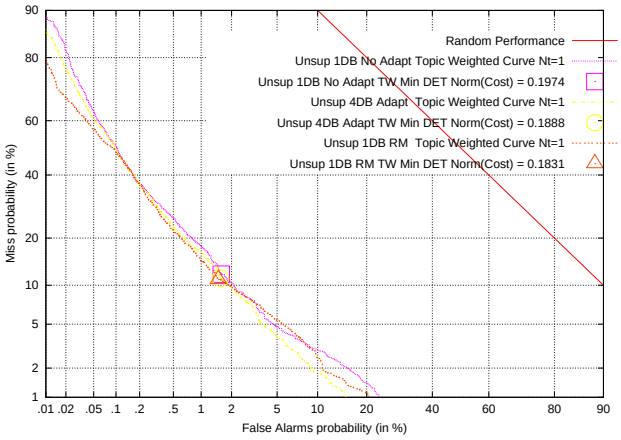

Figure 1: Det curve: Unsupervised tracking results

3.3.1 Adaptation

Unlike unsupervised tracking in supervised tracking there is no danger of adapting to documents that are false alarms. In supervised tracking we adapt if it is above the YES/NO threshold (that is, the document is delivered) and the document is relevant to the topic. Here also, we adapt using the top 100 terms.

3.3.2 Evidence of the advantages of supervision

1. **Impact of Supervision on Min-cost** As a first experiment we constructed a supervised version of UMASS-1 (TF-IDF + adaptation), called UMASS-1' and compared this to our best performing unsupervised system. We get DET curves corresponding to figure 2, which clearly demonstrates the impact of supervision.

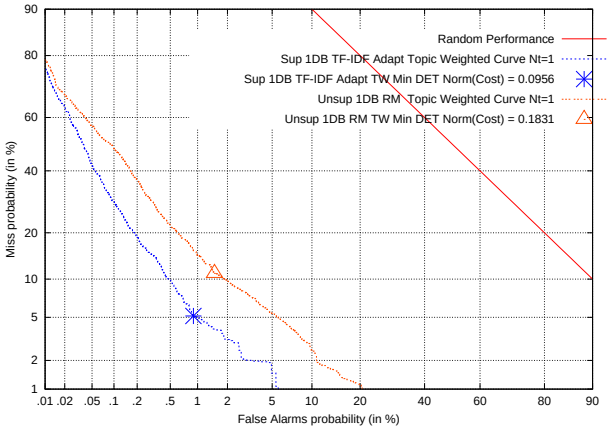

Figure 2: Det curve: From supervised to Unsupervised

2. **Impact of Language Specific comparisons in supervised learning** We now study the effect of the native language hypothesis by creating a supervised version of UMASS-2 (TF-IDF + native models+ adaptation), called UMASS-2'. The det curve comparing UMASS-1' and UMASS-2' is shown in figure 3.

3.3.3 Negative Feedback

In supervised tracking we obtain judgments for every document delivered. Thus, we get judgments for both hits and false alarms. We can thus adapt to negative feedback in a scheme like Rocchio.

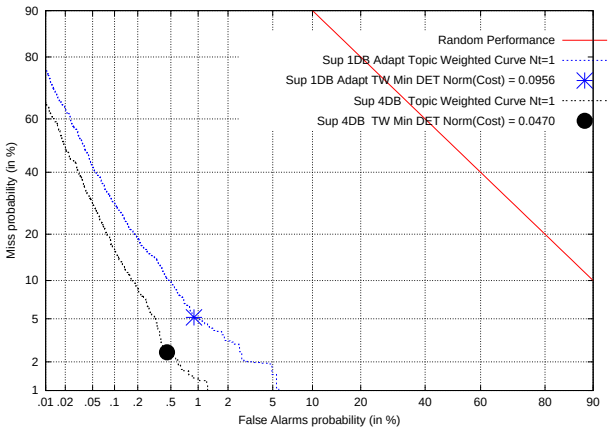

Figure 3: Det curve: Native Language comparisons

$$C = C + \alpha D \tag{3}$$

where C is the centroid vector, D is the document that has been delivered. $\alpha = 1$ if the document is on topic and $\alpha = -0.5$ if the document is off topic.

We found negative feedback useful for improving on the T11SU metric.

3.3.4 *Incrementing the threshold*

For some of our runs we played with the idea of incrementing the threshold when the system was doing badly. Basically, if the system was delivering a large number of non-relevant documents to the user we incremented the threshold. More specifically, if the ratio of relevant to non-relevant documents delivered fell below 0.1 we incremented the threshold. We found that incrementing the threshold helped improve the T11SU cost.

3.3.5 *Systems submitted*

In all, we submitted 5 runs for the official evaluation. In all cases, training was done using the specially sampled newswire portion of the TDT4 corpus as described.

1. **Relevance Models** (UMASS-2): This is the identical system as UMASS-5 for Unsupervised tracking, except that adaptation is only on the basis of relevant documents.

2. **Relevance Models + Increase thresholds** (UMASS-1): This is the identical system as the one described above. In addition, the YES/NO threshold is incremented by 0.1 as described above when $P(false - alarm) > 0.1$.

3. **TF-IDF + adaptation + negative feedback + increase threshold** (UMASS-3) This is the vector space system with 1 database as in UMASS-1 for unsupervised tracking. In addition, negative feedback and increase of thresholds as described above are implemented.

4. **TF-IDF + adaptation + language specific comparisons** (UMASS-4) This is identical to UMASS-2 for unsupervised tracking (that is, a vector space model with 4 databases is used), except that adaptation is done only on relevant stories.

5. **TF-IDF + adaptation + language specific comparisons + negative feedback + increase threshold** (UMASS-7) This is identical to UMASS-4 described above, except that negative feedback and increase of YES/NO thresholds was implemented.

On the training set it seemed like UMASS-1 and UMASS-2 performed reasonably well on both metrics (T11SU and MinCost). Results are discussed in the next section.

3.3.6 *Results and Discussion*

The following table shows the MinCost and T11SU for the development set (3.1) and for the test set, i.e the TDT5 data. We find that systems that perform well on one metric do not perform well on the other.

System	T11SU Dev	Mincost Dev	T11SU Test	MinCost Test
UMASS1	0.722	0.097	0.610	0.052
UMASS2	0.705	0.047	0.192	0.042
UMASS3	0.534	0.366	0.667	0.183
UMASS4	0.684	0.366	0.626	0.244
UMASS7	0.524	0.466	0.642	0.245

The highest T11SU on the TDT5 set is obtained using UMASS-3. Four of five of our systems obtained reasonably high T11SU utilities and 2 of our systems have quite low Min costs. UMASS-1 (although not the best system on either one of the metrics) seems to have done well on T11SU utility and on Mincost as well.

3.4 Baseline runs

We represent a story as a vector in term-space, where a coordinate represents the weight of a particular term in the story [4]. This weight is calculated using the usual $TF.IDF$ product [3], where TF is defined as

$$TF = tf/(tf + 0.5 + 1.5 * \frac{DL}{avg_D L}) \tag{4}$$

where tf is the number of times a given term occurs in a story (row term frequency), DL is the document length, and $avg_D L$ is the average length of the documents. IDF is found using the following formula.

$$IDF = \frac{log((N + 0.5)/DF)}{log(N + 1)} \tag{5}$$

where DF is the number of stories in the collection that contain one or more occurrences of the terms, and N is the total number of stories in the collection. Both these parameters N and DF are calculated incrementally with a deferral limit, i.e., N and DF are computed from the total number of documents seen so far up to the time of the deferral period. For all our runs, we used deferral one.

We start the tracking task by creating a cluster of Nt training stories given for a particular topic. All our runs are based on $Nt = 1$, i.e., considering only one story for the training. Therefore, the centroid of this cluster will be the same as the vector representation of the training story. In case of our baseline, which does no adaptation, this cluster will never change.

Incoming stories are compared with the centroid of the training cluster for measuring the similarity. We use cosine as the similarity function, which simply measures the inner product of the two vectors. Each vector is normalized to unit length, and has an identical number of terms. The terms are decrementally ranked by their weights. If the number of unique terms exceeds a given vector length, then the system will select the top number of terms for the centroid and the incoming story vectors.

For our baseline run, we used single database, where all the stories were in English. The original Arabic and Mandarin stories

were translated by provided machine translators. All the stories were lowercased and reduced to a root form by Krovetz's stemmer [8], which is a dictionary-based stemmer. Stop-words were removed using the standard InQuery stop-words list.

4. NEW EVENT DETECTION

4.1 Basic Model

We used the cosine similarity (Equation 6) metric to determine the similarity of a story with each story that arrived earlier. The belief in novelty of the story was taken to be its similarity with its closest neighbor in the past.

$$Sim(d,d') = \frac{\sum_w weight(w,d) * weight(w,d')}{\sqrt{\sum_w weight(w,d)^2}\sqrt{\sum_w weight(w,d')^2}} \quad (6)$$

where

$$weight(w,d) = tf * idf$$

$$tf = \log(termfrequecy + 1.0)$$

$$idf = \log((docCount + 1)/(documentfreq + 0.5))$$

4.2 Preprocessing

We used version 1.9 of the open source Lemur system[1] to tokenize the data, remove stop words, stem and create document vectors. We used the 418 stop-words included in the stop list used by InQuery [7], and the K-stem stemming algorithm [8] implementation provided as part of Lemur. Named entities were extracted using BBN Identifinder[5].

All collection-wide statistics are incremental [6]: we start with an empty collection and update statistics as the stories arrive.

A maximum decision deferral period of 10 source files was used, and the threshold for YES/NO decisions was set to 0.2. Each model run on a TDT collection was trained using English newswire stories from previous TDT collections. For example, we used English newswire stories from TDT2, TDT3, and TDT4 for training the model used for runs on TDT5.

4.3 Systems fielded

4.3.1 UMass1

In this model, for each story *S* the overlap of named entities and non named-entities with the six closest matching stories (Table 3) was measured. Using these scores as further evidence,the original confidence score *AllSimS1* assigned to the story was modified.

The final confidence score for story *S* was calculated using the formula given below.

1. $if \quad (AllSimS1 \leq 0.3)\{$
2. $\quad if \quad (NESimS1 \leq noNESimS1)\{$
3. $\quad\quad cnt \leftarrow 10$
4. $\quad\quad if \quad (NESimS2 < 0.1) \quad cnt \leftarrow cnt - 1$
5. $\quad\quad if \quad (NESimS3 < 0.1) \quad cnt \leftarrow cnt - 1$
6. $\quad\quad if \quad (NESimS4 < 0.1) \quad cnt \leftarrow cnt - 1$
7. $\quad\quad if \quad (NESimS5 < 0.1) \quad cnt \leftarrow cnt - 1$
8. $\quad\quad if \quad (NESimS6 < 0.1) \quad cnt \leftarrow cnt - 1$
9. $\quad\quad return((cnt/5.0) * AllSimS1)$
10. $\quad\quad \}else\{$
11. $\quad\quad\quad cnt \leftarrow 10$
12. $\quad\quad\quad if \quad (noNESimS2 < 0.1) \quad cnt \leftarrow cnt - 1$
13. $\quad\quad\quad if \quad (noNESimS3 < 0.1) \quad cnt \leftarrow cnt - 1$
14. $\quad\quad\quad if \quad (noNESimS4 < 0.1) \quad cnt \leftarrow cnt - 1$
15. $\quad\quad\quad if \quad (noNESimS5 < 0.1) \quad cnt \leftarrow cnt - 1$
16. $\quad\quad\quad if \quad (noNESimS6 < 0.1) \quad cnt \leftarrow cnt - 1$
17. $\quad\quad\quad return((cnt/5.0) * AllSimS1)$
18. $\quad\quad \}$
19. $else\{$
20. $\quad\quad return(AllSimS1)$
21. $\}$

[1] http://www.cs.cmu.edu/~lemur

Stories most similar to Story S	*Cosine similarity computed using*		
	All terms in document	*Only named entities*	*Terms excluding named entities*
S1	AllSimS1	NESimS1	noNESimS1
S2	AllSimS2	NESimS2	noNESimS2
S3	AllSimS3	NESimS3	noNESimS3
S4	AllSimS4	NESimS4	noNESimS4
S5	AllSimS5	NESimS5	noNESimS5
S6	AllSimS6	NESimS6	noNESimS6

Table 3: Features used to determine the final confidence score for a story *S*. The stories *S1* to *S6* are ordered in terms of decreasing cosine similarity to *S*.

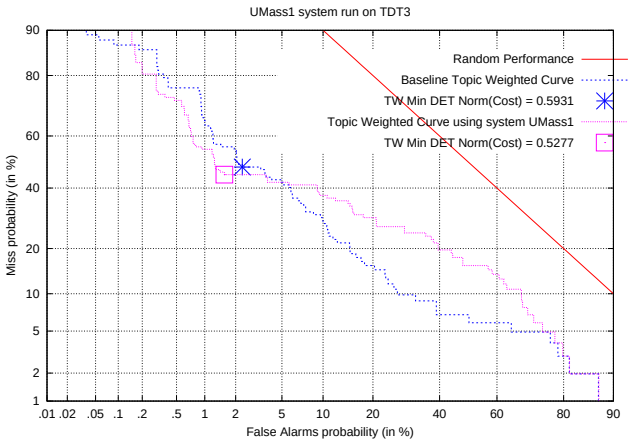

Figure 4: DET curves comparing the performance of baseline cosine system and UMass1.

The formula is designed to boost the confidence scores of *old* stories, and leave the *new* story confidence scores untouched. A large number of old stories have low confidence scores due to reasons like documents being on multiple topics, the topic being diffuse, and so on. Step 1. first checks if the original confidence score is low enough to warrant further investigation. Since it is possible that stories on a topic are linked together by named entities or non-named entities (or both), Step 2. is designed to detect which of the two groups of terms form the link between stories. Once this is done, the original confidence score is doubled. Steps 4. to 9. and 12. to 16. check to see if there are more stories that share a very high degree of similarity either due to named entities, or non-named entities. If this is not the case, then the confidence scores are slowly reduced back to the original value. All the numerical values

in the formula were obtained from training data.

Figure 4 shows the performance of UMass1 on TDT3. Table 4 summarizes the performance of the four systems on the TDT5 collection.

Table 4: Summary of the performance of UMass systems on the TDT5 collection.

	UMass1	UMass2	UMass3	UMass4
Topic Weighted Minimum Cost	0.8790	0.8387	0.8479	0.9213

4.3.2 UMass2

Each story was compared with all the stories preceding it. The highest cosine similarity was returned as the confidence score that a story was old.

4.3.3 UMass3

Each story was compared with a maximum of 25000 preceding stories that had the highest coordination match. The highest cosine similarity was returned as the confidence score that a story was old.

4.3.4 UMass4

UMass4 is similar to UMass1, except that it considers the five closest matches, and uses a different formula. The rationale is however the same.

$$
\begin{aligned}
&if \quad (AllSimS1 \leq 0.4)\{\\
&\quad cnt \leftarrow 8\\
&\quad if \ (NESimS2 < 0.1) OR (noNESimS2 < 0.05)\, cnt \leftarrow cnt-1\\
&\quad if \ (NESimS3 < 0.1) OR (noNESimS3 < 0.05)\, cnt \leftarrow cnt-1\\
&\quad if \ (NESimS4 < 0.1) OR (noNESimS4 < 0.05)\, cnt \leftarrow cnt-1\\
&\quad if \ (NESimS5 < 0.1) OR (noNESimS5 < 0.05)\, cnt \leftarrow cnt-1\\
&\quad return((cnt/4.0) * AllSimS1)\\
&else\{\\
&\quad return(AllSimS1)\\
&\}
\end{aligned}
$$

5. LINK DETECTION

For the TDT 2004 link detection cross-lingual task we used our existing TDT system to see how it performed on the new TDT5 data. Two methods of story representation and similarity measure were run. The first was the vector-space model with cosine similarity and the second relevance modeling. These models and further references are fully documented in [2] and [9].

To compare two stories for link detection with cosine similarity, we used the same vector-space TDT system that we have used in previous years. Each story is represented as a vector of terms with $tf \times idf$ term weights:

$$idf = \frac{log((N + 0.5)/df)}{log(N + 1)} \tag{7}$$

where tf is the number of occurrences of the term in the story, N is the total number of documents in the collection, and df is the number of documents containing the term. Collection statistics N and df were computed incrementally, based on the documents already in the stream within a deferral period after the test story arrives. The deferral period for link detection was 10.

Relevance Models is a statistical technique for estimating language models from small samples of text. Suppose Q is a short string of text and C is a large collection of documents, the language model for Q is estimated as:

$$P(w|Q) = \sum_{d \in C} P(w|M_d) P(M_d|Q) \tag{8}$$

A Relevance Model is a mixture of language models M_d of every document d in the collection. The document models M_d are *weighted* by the posterior probability of producing the query $P(M_d|Q)$. The posterior is computed as follows:

$$P(M_d|Q) = \frac{P(d) \prod_{q \in Q} P(q|M_d)}{\sum_{d' \in C} P(d') \prod_{q \in Q} P(q|M_{d'})} \tag{9}$$

The effect of equation (9) is that it assigns high weights to documents d that are most likely to have generated Q. A Relevance Model can be interpreted as a massive query expansion technique. To apply relevance modeling to story link detection we represent each story with a small number of terms, find the relevance model for each story and measure the similarity of the two models. Every story was pruned to 10 terms. The terms are those with the lowest probability of occurring by chance when picking 10 words randomly from the collection, C. To do this we use the *hypergeometric* distribution.

$$P_{chance}(A_w) = \frac{\begin{pmatrix} C_w \\ A_w \end{pmatrix} \begin{pmatrix} |C| - C_w \\ |A| - A_w \end{pmatrix}}{\begin{pmatrix} |C| \\ |A| \end{pmatrix}} \tag{10}$$

Where $|A|$ is the length of story A, A_w is number of times the word w occurs in A ,$|C|$ is the size of collection C and C_w is the total number of times w occurs in C.

Collection statistics C and C_w are computed incrementally, based on the documents already in the stream within a deferral period after the test story arrives. The deferral period for link detection was 10.

Once we have the estimated Relevance Models for stories A and B, similarity between the models is measured by symmetrized clarity-adjusted divergence.

$$\begin{aligned} S_{lnk}(A, B) &= \sum_w P(w|Q_A) \log \frac{P(w|Q_B)}{P(w|GE)} \\ &+ \sum_w P(w|Q_B) \log \frac{P(w|Q_A)}{P(w|GE)} \end{aligned} \tag{11}$$

$P(w|Q_A)$ refers to the relevance model estimated for story A, and $P(w|GE)$ is the background (General English) probability of w, computed from the entire collection C, within the deferral period.

The ROI post-processing experiments tried in 2003 were omitted in 2004.

Another important issue in multi-language link detection is how best to compare stories. Comparisons can be made with all stories in English, using the provided machine-translated versions of Mandarin and Arabic stories. An alternative is to compare two stories in the original language whenever possible. This choice was based on the belief that a similarity measure is more accurate when possible errors in translation are avoided. When both stories in the pair were originally in English the similarity comparison was done in English. The comparison was done in Chinese when both were

in Chinese and the comparison was done in Arabic the when both stories were Arabic. Otherwise cross-language comparisons were done in English. Corpus statistics were incrementally updated separately for each language.

English stories were lower-cased and stemmed using the kstem stemmer. Stop words were removed. Native Arabic stories were converted from Unicode UTF-8 to windows (CP1256) encoding, then normalized and stemmed with a light stemmer. Stop words were removed. Native Mandarin stories were converted from Unicode UTF-8 to gb. They were stopped with a list of Chinese stop words and a set of stopping rules. Bigrams were then made of character pairs.

The submissions were:

1. *baseline (umass/1D-costfidf)*: This run used the vector-space model with all stories machine-translated into English. Each vector was constrained to the 1000 highest terms. Both story vectors were weighted using the tf*idf weighting scheme. The YES/NO threshold was determined by training on the newswire portion of the TDT4 corpus and was set at 0.0929. The deferral period was 10.

2. *relevance model English (umass/1D-rm)*: This run used the language model approach with all stories machine-translated into English. The number of words in each story was pruned using a hypergeometric distribution. The YES/NO threshold was determined by training on the newswire portion of the TDT4 corpus and was set to 0.1096 . The deferral period was 10.

3. *native language comparisons (umass/4D-costfidf)*: This run used the vector space model, but stories were compared in their native language whenever possible. The YES/NO threshold was determined by training on the newswire portion of the TDT4 corpus and was set at 0.0840. The deferral period was 10.

4. *native language comparisons, relevance model (umass/4D-rm)*: This run used the language model approach, but stories were compared in their native language whenever possible. The number of words in each story was pruned using a hypergeometric distribution with the collection in the appropriate language. The YES/NO threshold was determined by training on the newswire portion of the TDT4 corpus and was set at 0.0910. The deferral period was 10.

Figure 5 shows the comparative performance of the four submitted conditions on the TDT-4 newswire data. Relevance modeling improves performance over the vector-space model and comparisons with native language stories, when possible, improves performance over comparisons with all stories in English. There is a clear picture that both relevance modeling and the use of native language comparisons improves performance, reinforcing our findings on the TDT-3 data.

	Minimum Cost		Cost
Detection Algorithm	TDT-4	TDT-5	TDT-5
Cosine tfidf (1D)	0.2260	0.1083	0.1667
Relevance Model (1D)	0.2066	0.1143	0.1732
Cosine tfidf (4D)	0.1890	0.1155	0.1318
Relevance Model (4D)	0.1743	0.0957	0.1270

Table 5: Performance of different algorithms on the Link Detection. Adjudicated results

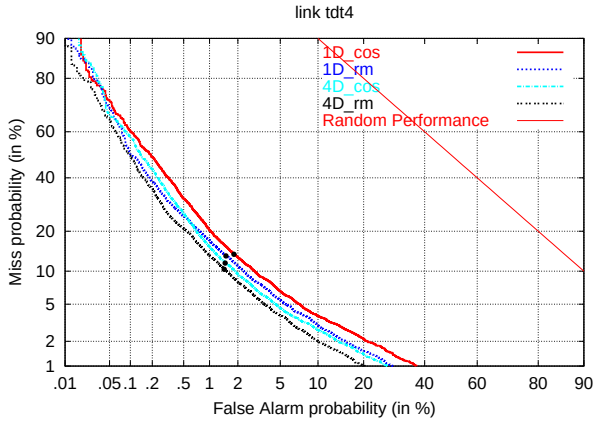

Figure 5: Det curve: for story Link Detection TDT-4 newswire training data

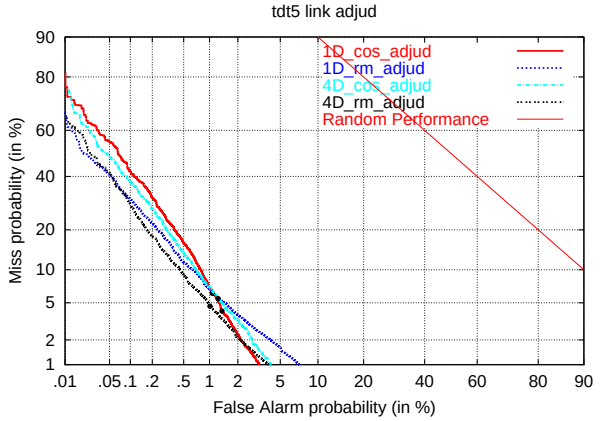

Figure 6: Det curve: for story Link Detection adjudicated TDT-5 data

Both the table 5 and the figure 6 show a somewhat different pattern on TDT-5 than on TDT-4. The Minimum cost is lower overall in TDT-5. The Relevance Model (4D) with native languages outperforms the others as in TDT-4. Because it is not clear from the graph that using native languages with the vector-space model was a win, we were suspicious of the results. Upon further investigation we found that when we used a Chinese segmenter on the native Mandarin stories prior to stopping and transforming them to bigrams, native language comparisons do improve performance. It is also not clear that relevance modeling in general is a win. It looks like at low thresholds it is not.

Our choices for TDT-5 YES/NO thresholds , which were based on TDT-4 training runs, were too low. Table 6 shows the minimum cost and the corresponding threshold for each of the four runs on the revised data. A graph of the four submitted conditions with the native Mandarin stories revised can be seen in Figure 7. One can see native language comparisons give better results than English-only comparisons, both in the vector-space model and the relevance model case. However on TDT-5, the relevance model does not outperform the vector-space model at high recall where the probability of miss becomes low.

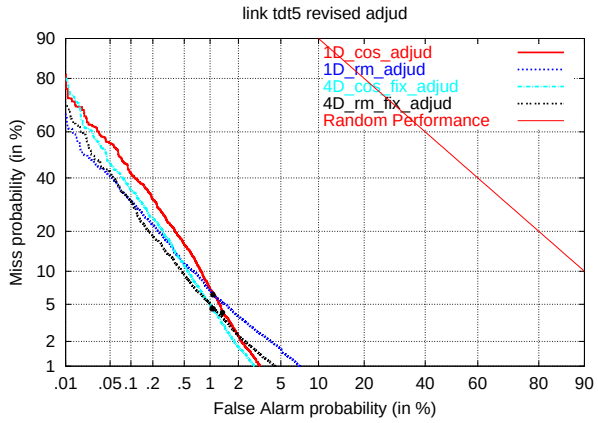

Figure 7: Det curve revised: for story Link Detection adjudicated TDT-5 data

Detection Algorithm	Minimum Cost TDT-5	Threshold TDT-5
Cosine tfidf (1D)	0.1083	0.1486
Relevance Model (1D)	0.1143	0.2058
Cosine tfidf (4D)	0.0965	0.1284
Relevance Model (4D)	0.0982	0.1590

Table 6: Revised performance of different algorithms on the Link Detection. Adjudicated results

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by SPAWARSYSCEN-SD grant number N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

6. REFERENCES

[1] J. Allan, A. Feng, and A. Bolivar. Flexible intrinsic evaluation of hierarchical clustering for tdt. In *In the Proc. of the ACM Twelfth International Conference on Information and Knowledge Management*, pages 263–270, Nov 2003.

[2] James Allan, Alvaro Bolivar, Margaret Connell, Steve Cronen-Townsend, Ao Feng, Fangfang Feng, Giridhar Kumaran, Leah Larkey, Victor Lavrenko, and Hema Raghavan. Umass tdt 2003 research summary. In *Proceedings of TDT2003 evaluation, unpublished*, 2003.

[3] James Allan, Jamie Callan, Fangfang Feng, and D. Malin. Inquery and trec-8. In *Proceedings of TREC-8*, 1999.

[4] James Allan, H. Jin, M. Rajman, C. Wayne, D. Gildea, V. Lavrenko, R. Hoberman, and D. Caputo. Topic-based novelty detection. In *summer workshop at CLSP*, 1999.

[5] Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

[6] Thorsten Brants, Francine Chen, and Ayaman Farahat. A system for new event detection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 330–337. ACM Press, 2003.

[7] James P. Callan, W. Bruce Croft, and Stephen M. Harding. The INQUERY retrieval system. In *Proceedings of DEXA-92, 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[8] Robert Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 191–202. ACM Press, 1993.

[9] Leah Larkey, Fangfang Feng, Margaret Connell, and Victor Lavrenko. Language-specific models in multilingual topic tracking. In *Proceedings of SIGIR 2004, pp. 402-409*, Sheffield England, 2004.

[10] Ramesh Nallapati, Ao Feng, Fuchun Peng, and James Allan. Event threading within news topics. In *ACM Thirteenth International Conference on Information and Knowledge Management*, 2004.