

UMass TDT 2003 Research Summary

Margaret Connell, Stephen Cronen-Townsend, Ao Feng,
Fangfang Feng, Giridhar Kumaran, Hema Raghavan
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{connell,crotown,aofeng,feng,giridhar,hema}@cs.umass.edu

1. ROI CLASSIFICATION (GIRI AND STEVE) (ROI).

As part of an effort to improve the performance of current vector-similarity-based Topic Detection and Tracking (TDT) systems, we decided to analyze the mistakes committed by them. A annotation system was developed using Java Swing for this purpose. We hoped that annotators using the system would be able to identify patterns in the types of mistakes made - information we could later use to improve TDT algorithms.

Among other information that was obtained from the annotations, it was observed that the vector space similarity measures left much to be desired. The similarity measures were easily thrown off-gear when there was a big difference in the length of documents. Also, a look at the contribution of individual terms to the overall similarity scores revealed that a bad job of assigning term weights was being done. For example, while comparing two stories on different topics in health care, terms like *drugs*, *cost*, *coverage*, *plan*, *prescription* etc. contributed most to the overall similarity score. While in a way this was a good thing to happen, it only helped in identifying that the two stories were on a similar issue. The two stories actually involved completely different locations and individuals! While tf-idf weighting should have suppressed the contributions of the most similar terms and revealed the difference in the stories in the form of an overall low similarity score, it apparently failed to do so.

While one solution was to develop a completely new similarity measure, we instead decided to improve upon the existing one. We believed that the problem of weight assignment to terms could be resolved by first placing stories into broad categories, and then computing term weights using the statistics within those categories. The next step was to determine what those broad categories would be. Since we were in essence trying to capture the LDC's methodology of determining topics (and hence new/old stories), a natural choice for the categories were the rules of interpretation

We investigated two approaches to developing a classifier to classify stories according to the ROIs.

1.1 Training

For our initial experiments we trained on TDT-2 judged documents and tested on TDT-3 documents. For our classification of TDT-4 (used in submissions) we trained on judged documents from TDT-2 plus TDT-3. In both cases, the training data was prepared in the same way.

First, the IDs of all judged stories with topic level "YES" were stored along with the rule of interpretation (ROI) number (1-13) of the topic they were relevant to. Any duplicates due to a story belonging to more than one TDT topic on the same rule of interpretation were removed. Finally, stories that belonged to more than one rule of interpretation were removed, leaving best exemplars we had for the 13 classes as the training data. Table 1 shows how the interesting cases in preparing the training data.

The most extreme case of a judged story that was omitted from the training data is a story in TDT-3 that is relevant to the topic "Chinese Labor Activists" with the "Legal/Criminal Cases" ROI, the topic "Blair Visits China in October" with the "Political/Diplomatic Meetings" ROI, and the topic "China will not Allow Opposition Parties" with the "Miscellaneous" ROI. Cases like this provide a cautionary note about our technique: some stories can clearly belong to several ROIs, though we try to estimate a single correct ROI for a given story.

Since we wished to build a general-purpose classifier that would work on data sets obtained from any time frame (new people, places etc. keep coming up in the news), we also experimented with training and testing sets in which the named entities were removed.

1.2 Naive Bayes Classification

One method we tried to use to estimate the correct classification of a documents was Naive Bayes classification[7]. We used a probability ratio formulation on smoothed multinomial language models of the documents[8, 6] This was tried on both the straight SGML (using ASR transcripts for broadcast sources and all non-english documents in translation) and to text with the named entities removed by Identifinder.

In particular, for each class we make a linearly smoothed language model which is a straight average of document models for all training stories. For each of the 13 classes, each document, D , in the testing set is ranked by the prob-

Collection	In Multiple Topics, same ROI	In multiple topics, different ROIs	Training Stories Used
TDT-2	26	85	9940
TDT-3	248	315	9758

Table 1: The number of stories in various cases used in creating the training data. Training stories judged on topic for topics with different rules of interpretation were not included in the training data.

ability ratio $P(D|Class)/P(D/Coll)$ which is the probability of the document being generated by the particular class model divided by the probability of the document being generated by the collection model. In this way were able to rank all test documents within each class and compute recall-precision graphs and other standard IR measures of the individual classifiers. We used the mean average precision over the 13 classifiers to tune the smoothing parameter and make other tests and optimizations when training on TDT-2 judged documents and testing on TDT-3 judged documents.

For classification, we use Bayesian inversion to compute $P(Class|D) = P(D|Class) * P(Class)/P(D)$. In the case of uniform priors, which was what we used for our standard runs, this is trivial and the system’s classification is taken to be the highest probability class. We also explored using the 13 $P(D|Class)$ scores for each document to be a fuzzy classification of each document. Though we have yet to find a way to use this fuzzy classification to lower TDT cost on various tasks, our intuition still says there will be a way to incorporate it profitably.

To give a sense of the system we used to make classification that were tried in our submissions, Table 2 gives the confusion matrix for our classification of TDT-3 judged stories with training on TDT-2 judged stories. The smoothing used in this case was 0.4 times the document relative frequency of a term plus 0.6 times the collection frequency of the term. Tremendous amounts of information about the performance of our naive Bayes classifiers are contained in this table. For example, we see that many “Legal/Criminal Cases” stories were misclassified as “Acts of Violence or War.” Of course, “mistakes” of the classifiers are convoluted with the fact that, in reality, a story does not have a single correct ROI classification.

1.3 Classification Using BoosTexter

BoosTexter is a general purpose machine-learning program based on boosting for building a classifier from text and/or attribute-value data[9]. Given training data, BoosTexter creates a series of simple rules that are used to build the classifier.

In our experiments, the terms in a document, weighted by their frequency of occurrence in it, were used as features. We compared the performance of Boostexter with Naive Bayes Classification by comparing their confusion matrices shown in Table 2 and Table 3 respectively. Trial runs of the New Event Detection task on TDT3 were done using the ROIs provided by BoosTexter and the Naive Bayes Classifier. The runs using BoosTexter ROIs proved to be more successful.

2. LINK DETECTION AND TRACKING (AO)

In tracking and story link detection, rule of interpretation (ROI) has been applied to most of the results. It comes from the idea that two stories within the same topic should

be in the same subject. Annotation experiments proved the conclusion. The list of subjects have been changed several times, at last we decided to use NIST’s rules of interpretation as the subjects [12].

There are two sets of classifiers for ROI. One is BoostTexter (by Giri), the other is Nave Bayes (by Steve). Experiments in TDT3 show that BoostTexter with name entities performs the best and improved the SLD results a lot.

Here is how ROI works in TDT2003.

2.1 Link detection

Post-process the system output. If the two stories in the pair belong to the same ROI, then keep the score unchanged. Otherwise, divide the score by 3 (1/3 is the optimal parameter in TDT3).

Experiments in TDT3 (trained in TDT2) shows that BoostTexter with name entities can improve the performance of story link detection and tracking obviously, so we expected a similar result in TDT2003. Pre-adjudicated results show a negative effect, which may be caused by the different of name entities between TDT3 and TDT4.

For most link detection submission, 4 different sets of ROI have been applied. The names ends with RI1, RI2, RI3 and RI4 respectively and the notation is explained below:

- RI1: ROI generated by BoostTexter with name entities included
- RI2: BoostTexter without name entities
- RI3: Nave Bayes with name entities
- RI4: Nave Bayes without name entities

The results are listed in table 4:

2.2 Tracking

Suppose there are n training stories in the model (if the model is non-adaptive, n=Nt; if the model is adaptive, n also includes the stories that are added to the model). If a new story shares the same ROI with m training stories, then we define the new score as

$$new_score = old_score \left(\frac{1}{3} + \left(1 - \frac{1}{3}\right) \frac{m}{n} \right) \quad (1)$$

For each tracking submission, there is a corresponding submission after applying ROI names. The changes made to the original output file are to change the score using the profile above and modify the system name as well. All other parameters keep unchanged.

Minimal costs for ROI runs are in table 5

3. VECTOR PRUNING AND NEW EVENT DETECTION (GIRI)

Using the ROI approach the same way it was used in link detection led to a miniscule improvement on the New Event

True ROI	1	2	3	4	5	6	7	8	9	10	11	12	13	totals
1. Elections	624	169	11	2	3	37	4	18	53	7	9	3	147	1087
2. Scandals/Hearings	19	63	15	2	10	2	1	6	0	63	21	2	13	217
3. Legal/Criminal Cases	111	209	688	11	50	443	7	134	7	29	76	51	31	1847
4. Natural Disasters	25	0	1	492	15	6	9	41	2	4	11	0	1	607
5. Accidents	8	0	9	8	134	3	0	1	0	1	0	3	0	167
6. Acts of Violence or War	44	2	23	3	21	348	1	6	1	0	12	51	7	519
7. Science and Discovery	20	0	14	2	1	1	296	9	0	1	2	18	5	369
8. Financial	110	11	1	5	2	70	10	804	0	4	16	14	23	1070
9. New Laws	3	0	0	0	0	0	0	0	0	14	0	0	0	17
10. Sports	69	10	16	25	11	18	6	52	1	678	9	3	35	933
11. Political & Dilomatic Mtgs	62	17	4	4	14	905	1	243	2	4	212	4	4	1476
12. Celebrity/Human Interest	17	5	27	3	0	140	56	21	1	14	44	46	2	376
13. Miscellaneous	22	45	0	17	24	77	3	108	48	2	44	0	51	441

Table 2: The confusion matrix for classifying TDT-3 single ROI stories based on training with TDT-2 single ROI stories. The labels down the left hand side are the true ROIs and the numbers across the top are the system classifications. The entry 905 in column 6 and row “11. Political & Diplomatic Mtgs” means that 905 stories that actually were “12. Political & Diplomatic Mtgs” stories were misclassified as “6. Acts of Violence or War” stories.

True ROI	1	2	3	4	5	6	7	8	9	10	11	12	13	totals
1. Elections	426	169	23	1	1	243	15	64	0	33	25	11	76	1087
2. Scandals/Hearings	3	45	21	0	6	55	1	10	0	53	3	9	11	217
3. Legal Criminal Cases	10	165	67	0	24	688	5	132	0	38	31	30	50	1848
4. Natural Disasters	2	11	20	163	76	121	12	94	1	31	26	12	38	607
5. Accidents	1	6	12	1	112	25	1	2	0	2	0	1	4	167
6. Acts of violence or war	9	2	36	3	6	418	0	13	0	4	15	9	4	519
7. Science & Discovery	0	1	13	1	5	33	237	23	1	11	11	24	9	369
8. Financial	6	18	5	0	3	101	4	893	0	14	16	0	10	1070
9. New Laws	0	0	0	0	0	1	0	2	0	14	0	0	0	17
10. Sports	0	10	15	6	11	82	2	70	0	699	10	6	22	933
11. Political & Diplomatic Mtgs.	7	16	4	1	1	1066	1	177	1	3	186	2	11	1476
12. Celebrity & Human Interest	3	13	17	1	1	130	13	48	0	31	50	39	30	376
13. Miscellaneous	9	33	13	4	9	167	5	101	15	5	32	3	45	441

Table 3: The confusion matrix for classifying TDT-3 single ROI stories based on training with TDT-2 single ROI stories using BoosTexter

Run	Result	Run	Result	Run	Result	Run	Result
1DcosRI1	0.3090	1DcosRI2	0.3220	1DcosRI3	0.2867	1DcosRI4	0.2937
4DcosRI1	0.2587	4DcosRI2	0.2649	4DcosRI3	0.2407	4DcosRI4	0.2463
UDcosRI1	0.2938	UDcosRI2	0.3020	UDcosRI3	0.2697	UDcosRI4	0.2738

Table 4: SLD runs and Results.

Nt	Run	Result	Run	Result	Run	Result	Run	Result
1	1DcosROI	0.2881	4DcosROI	0.2628	ADcosROI	0.2466	UDcosROI	0.2806
4	1DcosROI	0.2124	4DcosROI	0.1727	ADcosROI	0.1684	UDcosROI	0.2022

Table 5: Tracking runs and Results.

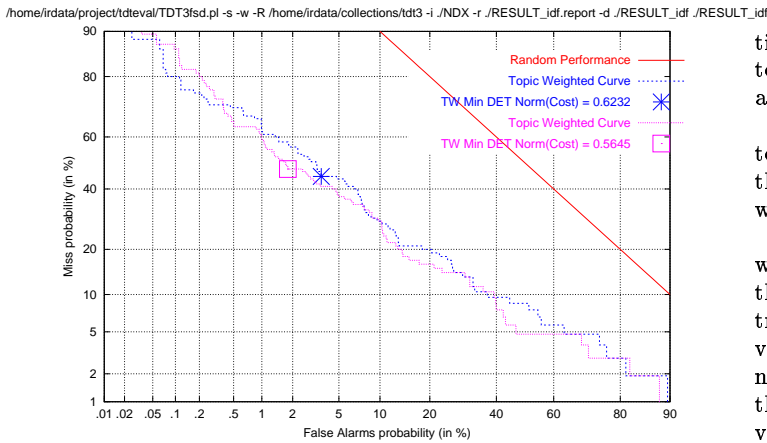


Figure 1: DET curve for New Event Detection task on the TDT3 collection.

Detection (NED) task. An alternative approach tried was to use the ROIs to cluster stories and then compute the term weights based on the statistics of the group. If anything, this approach hurt. One possible reason was that the strict classification of the story into a single ROI resulted in aggravating the mistakes made by the classifier. A probabilistic approach would have probably fared better.

Going back to the problem discussed in Section 1 of important terms (like named entities) not getting weighted higher than other terms in a document, the availability of a ROI for each story brought in a fresh perspective. Rather than working hard to somehow boost the weights of *important* terms, we could instead consider reducing (or setting to zero) the weights of so-called *unimportant* or *noise* terms. Such *noise* terms were selected in a rather ad-hoc fashion. They were identified as the three hundred most frequent terms, excluding named entities, that occur in the documents assigned to each ROI. The documents were the judged ones in TDT2. Thus, a *stop list* for each ROI was created. For example, the top ten terms for the Elections ROI stop list were *elect*, *party*, *vote*, *govern*, *president*, *lead*, *politics*, *say*, *people*, and *support*. While creating the vector representation of a document, depending on the ROI the document was assigned to by the classifier, terms that were present in the particular ROIs stop list were discarded. Comparison between story pairs was then performed using these pruned vector representations. This approach resulted in a significant improvement in NED performance (Figure 1).

In addition to vector pruning, the other features of the NED algorithm include

1. Similarity score normalization
2. Removing short stories from consideration

4. CROSS-LINGUAL MODELS FOR TRACKING (FANGFANG)

Basically our TDT core system has been used for the TDT4 tracking task. A story is represented as a vector in term-space, where coordinates represent the weight of a particular term in the story [3].

A small number, N_t , of training stories is given for a particular topic. Those training stories are grouped into a cluster. The cluster is represented by a centroid, which is an average of the vector representatives of the training stories.

Incoming stories are compared to the centroid of the cluster. If the similarity of the story to the centroid exceeds a threshold, then the story is considered as "on-topic", otherwise "off-topic".

The cosine has been selected as the similarity function, which simply measures an inner product of two vectors. In the tracking task, it measures the inner product of the centroid vector and the vector of the incoming story. Each vector is normalized to unit length, and has an identical number of terms. The terms are decrementally ranked by their weight. 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. Both vectors are weighted using $tf \cdot IDF$ weighting scheme [1]:

$$IDF = \frac{\log((N + 0.5)/DF)}{\log(N + 1)} \quad (2)$$

where tf is the number of times a given term occurs in a story, DF is the number of story in the collection that contain one or more occurrences of the terms, and N is the total number of stories in the collection. All collection-wide statistics, N and DF are taken incrementally with a deferral limit. That is that DF and N are computed from the total number of documents seen so far up to the time of the deferral period.

For the topic tracking task, we submitted eight runs for both required conditions and alternate conditions, four without ROI (Rule Of Interpretation) [4] and four with ROI. The primary system has been described above. All runs used the deferral one. The differences for each contrastive runs were the databases, the vector length and the thresholds. The submissions were:

1. *1DcosIDF*: Using single database, all stories were in English. The original Arabic and Mandarin stories were translated by provided machine translators. All stories were lowercased and reduced to a root form by a dictionary-based stemmer, called Kstemmer. Stop-words were removed. The vector length was 1000.

The minimum normalized cost was 0.1676 for the alternate condition run, and 0.1964 for the required condition run.

2. *UDcosIDF*: The only difference from *1DcosIDF* was that, instead of using provided translation, the original Arabic stories were translated by our own translator based on a probabilistic dictionary constructed from a UN parallel corpus [11]. The Arabic stories were encoded in CP1256 and stemmed using a light stemming algorithm developed at UMASS [2]. Each Arabic word was translated using the probabilistic dictionary, in which each Arabic-English word pair had an associated probability. The probabilities of each English word in the translated stories were summed. In order to limit the size of the translated English story, we retained those words whose summed probabilities were the greatest. The summed probabilities were multiplied by three. The integer part of the product repre-

sented the frequency of the English word in the translated story.

The minimum normalized cost was 0.1594 for the alternate condition run, and 0.2024 for the required condition run.

3. *ADcosIDF*: Using the same database as 1DcosIDF but an adaptive centroid with an adapting threshold, and a vector length of 100 rather than 1000. If an incoming story had a similarity score which greater than the adapting threshold then the story would be added to the centroid vector. At the maximum, 100 stories could be added to the centroid vector.

The minimum normalized cost was 0.1443 for the alternate condition run with the adapting threshold 0.5, 0.2007 for the required condition run with the threshold 0.4.

4. *4DcosIDF*: For avoiding the possible translation errors, we chose to make comparisons of stories in their native language whenever possible. We used four databases, a global one same as UDcosIDF, and three individuals for each language, English, Arabic, and Mandarin. The English one contains 28390 original English stories. all English stories were stopped and stemmed like 1DcosIDF.

The Arabic database contains 42713 Arabic stories without any translation. The Arabic stories were encoded in CP1256 and stemmed by a light stemming algorithm developed at UMASS.

The Mandarin database contains 27142 original Mandarin stories without any translation. All Mandarin stories were stopped by a list of Chinese stop words [10] and a set of stopping rules, then made bigrams of character pairs. For the required conditions (i.e. manual transcription) all Mandarin stories were tokenized in single characters, no Chinese word segmentation was involved. For the alternate conditions (i.e. ASR transcription), there were 8784 ASR stories having been segmented in words by speech recognizers. We had to use a Chinese segmenter[10] to segment other stories, then made the bigrams without crossing the word boundaries.

For the given training story (i.e. Nt=1) or stories (i.e. Nt=4) the system created two centroid vectors, one from the global database and the other one from the English database because all the training stories were in English. If an incoming story was in English then the system calculated the similarity with the English centroid otherwise calculated the similarity with the global centroid. If the similarity was greater than an average score of the centroid vector when Nt=4, or 0.5 when Nt=1, which was a default adaptive threshold because the average score was equal to one as Nt=1, then the story would be added to the centroid. If the story was in Arabic or Mandarin the system would establish a centroid for Arabic or Mandarin, and then for those subsequent Arabic or Mandarin stories the system would calculate the similarity with the Arabic or Mandarin centroid, rather than the global one.

When the system updated the centroid vector, only top 100 terms were selected, i.e. the vector length was

limited to 100. At the maximum, 100 stories could be added to the centroid vector.

The minimum normalized cost was 0.1501 for the alternate condition run with the average score as the adapting threshold, 0.2036 for the required condition run with the default adapting threshold 0.5.

5. CROSS-LINGUAL EXPERIMENTS IN STORY LINK DETECTION (MARGIE)

Here is a Description of specific runs done:

5.1 UMass02 cosTFIDF

The system settings that were used for this run, were

model	single link (set to knn 1)
condition	bnasr, defer=10
similarity	cosine
weighting	idf
threshold	0.1
deferral	10

We used a single database with all documents in English and the provided translations of non-English stories. All stories were lower-cased and reduced to a root form by a dictionary-based stemmer, called Kstemmer. Stop-words were removed.

Minimum normalized cost was 0.2472

5.2 UMass13 UDcosIDF

The system settings that were used for this run, were

model	single link (set to knn 1)
condition	bnasr, defer=10
similarity	cosine
weighting	idf
threshold	0.12
deferral	10

In this run we used an alternate English translation of the Arabic stories. The resulting translations replaced those that were provided. In order to convert Arabic stories into English, the Arabic was converted to Windows encoding (CP1256) and stemmed using a light stemming algorithm developed at UMASS [5]. Each Arabic word was translated using a probabilistic dictionary, derived from the UN Arabic/English parallel corpus distributed by LDC [11]. For each Arabic-English word pair, there is an associated probability $P(e|a)$. The probabilities of each English word in the translated document were summed. In order to limit the size of the translated English story, we retained those words whose summed probabilities were the greatest. The summed probabilities were multiplied by three. The integer part of the resulting number represents the frequency of the English word in the translated story. For all comparisons English words were stemmed using kstem and stop words were removed.

Minimum normalized cost was 0.2439

5.3 UMass07 4DcosIDF

The system settings used were:

model	single link (set to knn 1)
condition	bnasr, defer=10
similarity	cosine
weighting	idf
threshold	0.076
deferral	10

In this run the document comparisons were done in their native 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 when both stories were Arabic. Otherwise cross-language comparisons were done in English, with Arabic documents translated into English using the dictionary based on the UN parallel corpus and with the Chinese documents translated using the provided translation. Corpus statistics were incrementally updated separately for each language as story pairs arrived. The tf*IDF statistics were updated using a native language corpus for a same-language pair and the most recent of the pair was the last story considered in the update. The tf*IDF statistics for a cross-language pair were based on all stories available through the end of the deferral period. Although we experimented with score normalization to compensate for the effects of differing score distributions due to language, we found it did not affect the results significantly. We therefore did not normalize scores. When the comparison was done on English stories, words were stemmed and stop words were removed. Native Arabic documents were converted to Windows encoding (CP1256) and stemmed using a light stemming algorithm developed at UMASS [5]. Native Mandarin stories were stopped with a list of Chinese stop words and a set of stopping rules. Bigrams were then made of character pairs. Some of the stories had been segmented into words by speech recognizers. We used a Chinese segmenter [10] to segment the remaining unsegmented stories. Bigrams were then formed without crossing word boundaries.

Minimum normalized cost was 0.1983

6. CORRECTING NAMED ENTITY ERRORS USING SOUNDEX CODES (HEMA)

6.1 Motivation

It is intuitive that named entities -people, places and organizations are probably useful in determining whether two stories discuss the same topics or not. Often the top-terms for a document, when ranked by a TF-IDF like weighting mechanism are named entities. However, it is these named entities that are often the source of ASR and Machine Translation errors. For example "Lewinsky" and "Lewinsky" refer to the same entity, and a document which talks about "Monica Lewinsky" and one that talks about "Monika Lewinsky" will be measured as more dissimilar than they should be, when traditional document similarity metrics are used. Therefore the two forms of the same name should be normalized to one canonical form, before computing the similarity score.

6.2 Implementation

To implement this idea, we used a named entity recognizer, viz., BBN's Identifinder to mark all mentions of the named entities in the corpus. Now, given two documents D_1 and D_2 whose similarity we have to measure, we can do the comparison in one of two different ways.

In one approach we can compute the distance between two documents by computing the distance between words

in the two documents, and using these distances to do the normalization. We can compute the distance between two words a and b using the Levenshtein distance (or any of its variants), and if this distance is less than some specified constant δ , we can normalize b to a . However this means that all entities in D_1 have to be compared to all entities in D_1 and D_2 , which is inefficient.

The other alternative is to preprocess the corpus, and normalize all mentions of a named entity to a given canonical form, where the canonical form is independent of mentions of other entities in the two documents being compared. Soundex, Phonix, and other such codes offer us a means of normalizing a word to its phonetic form. Hence, similar sounding names like "Lewinsky" and "Lewinsky" are both reduced to the same soundex code "l520". If our original corpus is C , we can create a new corpus C' where all the named entities are replaced by their soundex codes. Hence in our new methodology, instead of computing $Sim(S_1, S_2)$, we compute $\frac{1}{2}(Sim(S'_1, S'_2) + Sim(S_1, S_2))$ where S'_1 and S'_2 are the corresponding documents in C' .

We tested our idea on the TDT3 corpus for the Story Link Detection Task, using the Cosine similarity metric, and found that performance actually degraded. On investigation we found that the named entity recognizer performs poorly on Machine Translated and ASR source data. Our named entity recognizer relies considerably on sentence structure, to make its predictions. MT output often lacks grammatical structure, and ASR output does not have punctuation.

We therefore decided to test our idea for newswire text. We created our own test set of 4752 pairs of stories from newswire sources. This test set was created by randomly picking on and off-topic stories for each topic.

We now compute $Sim(S_1, S_2) = \frac{1}{2}(Cos(S_1, S_2) + Cos(S'_1, S'_2))$. On this given test set, we get a nearly 10% improvement to the minimum cost as compared to using $Sim(S_1, S_2) = Cos(S_1, S_2)$ (the traditional cosine metric).

Since we found that our technique works well for newswire text, we decided to try the following methodology for SLD on the TDT3 evaluation NDX file:

- If, for a given pair, the sources for both documents are from newswire then $Sim(S_1, S_2) = \frac{1}{2}(Cos(S_1, S_2) + Cos(S'_1, S'_2))$.
- else compute $Sim(S_1, S_2) = Cos(S_1, S_2)$.

The performance of this method, when compared to using the traditional cosine metric, gave 0% improvement. On investigation we found that only about 5000 of about 27000 pairs were purely newswire-newswire pairs, and hence any improvement was completely drowned by the fact that most SLD pairs contained atleast one non-newswire source.

6.3 Conclusions

We showed that named entity normalization using soundex codes is useful. However named entity recognition in noisy data remains is hard. Since TDT tasks involve a significant amount of such noisy sources, we did not implement this idea in our final run.

7. REFERENCES

- [1] J. Allan, J. Callan, F. Feng, and D. Malin. Inquiry and trec-8. In *Proceedings of TREC-8*, 1999.

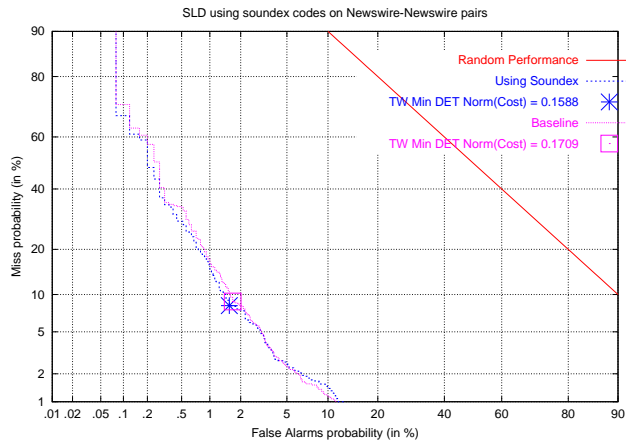


Figure 2: SLD performance

- [2] J. Allan, M. Connell, W. Croft, F. Feng, D. Fisher, and X. Li. Inquiry and trec-9. In *Proceedings of TREC-9*, pages 551–577.
- [3] J. Allan, H. Jin, M. Rajman, C. Wayne, D. Gildea, H. R. Lavrenko, V., and D. Caputo. Topic-based novelty detection. In *summer workshop at CLSP*, 1999.
- [4] A. Feng. description about the roi for tracking.
- [5] L. Larkey, L. Ballesteros, and M. Connell. Improving stemming for arabic information retrieval: Light stemming and co-occurrence analysis. In *SIGIR*, pages 275–282, 2002.
- [6] V. Lavrenko and W. B. Croft. Relevance models in information retrieval. In W. B. Croft and J. Lafferty, editors, *Language Modeling for Information Retrieval*, pages 11–56. Kluwer Academic Publishers, 2003.
- [7] D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.
- [8] S. E. Robertson. *The Probability Ranking Principle in IR*, pages 281–286. Morgan Kaufmann Publishers, Inc., 1997.
- [9] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. In *Machine Learning 39(2/3):1*, pages 35–168. Kluwer Academic Publishers, 2000.
- [10] L. D. C. URL. Chinese segmentation.
- [11] L. D. C. URL. Machine translation resources.
- [12] L. D. C. URL. Roi annotation instructions.