EXTRACTING AND USING RELATIONSHIPS FOUND IN TEXT FOR TOPIC TRACKING

A Senior Honors Thesis Presented

By

Paul Ogilvie

Submitted May 2000

APPROVED

James Allan, Computer Science

David Jensen, Computer Science

Walter Rosenkrantz, Statistics

Abstract

We investigate the extraction of linked-object representations (LORs) from text for use in topic tracking. LORs provide us a way to represent relationships between objects found in text. We show the use of naive coreference resolution during the extraction of objects does not provide improvement over the absence of coreference resolution. We investigate the creation of links, or relationships, between objects through closeness of the objects in text for small and large window sizes. We present a new algorithm, "centers", for document comparison and evaluate its effectiveness, showing that it approaches traditional cosine similarity when the window size is large. It gives different answers when the window size is small, but does not perform as well as cosine similarity where vectors contain words as components. Also, the "centers" algorithm is able to provide lower miss rates than when using vectors with LORs as components.

1 Introduction

The field of Information Retrieval relies heavily on "bag of words" techniques. These techniques make use of the presence of words in a story but little other information. They are called "bag of words" techniques because they treat text merely as a set, or bag, of words, possibly with numbers associated with the words. Text contains much more information, but it is difficult to make use of other information. In this paper we try to make use of more information than just the presence of words.

We consider the problem of topic tracking, which is a task in the DARPA Topic Detection and Tracking (TDT) program. In topic tracking, a system is given a few example news articles of a topic and the system must then make decisions as to whether other news articles are on the same topic as the example stories.

We investigate the extraction and use of linked object representations (LORs). LORs allow representation of relationships between objects found in text. We consider relationships based on proximity of objects in a story. We evaluate a variety of methods for choosing objects and creating relationships. It is not obvious how the LORs should be used for tracking, so we investigate a method similar to a "bag of words" approach and also present a new method. The modified "bag of words" approach we use is cosine similarity, or dot product, on vectors where the components in the vector are relationships. The new method, called the "centers" method in this paper, creates multiple vectors for a story from the LORs and compares the vectors using cosine similarity. We compare these uses of LORs to a traditional information retrieval method of cosine similarity on vectors with components of words with $tf \cdot idf$ weighting.

While we examine the use of relationships only in the context of topic tracking, the techniques and evaluation of techniques are applicable to additional tasks. This work is applicable to any task where there are document-to-document comparisonsm including filtering with example documents, relevance feedback, and any of the other tasks in TDT.

In the next section, we discuss extraction of LORs. In Section 3, we discuss the use of LORs in topic tracking. Section 4 gives a brief description of implementation and efficiency concerns. We present the evaluation methodology and results in Sections 5 and 6. We discuss related work in Section 7. Our conclusions and possible future work are in Section 8.

2 Extraction

2.1 Common Information Retrieval Techniques

Before giving of linked object representations, we recall some useful standard information retrieval techniques. These are stopping, stemming, and part-of-speech tagging.

Stopping is the removal of common words such as "the", "and", and so on. Stemming is the conversion of words to their root form. For example, "marketing" has a stem of "market". Part-of-speech tagging labels each word with their part of speech.

2.2 Linked Object Representations

In an attempt to capture more information than current "bag-of-words" models, we will extract linked object representations (LORs). Linked object representations are a way to characterize relationships between objects. As objects, we will primarily use people, organizations, locations, and other nouns. While we will focus on the use of information found in nouns, objects could be constructed from other information. The relationships created between objects can carry typing information or remain without type.

2.3 Finding Objects

The objects we are interested in are people, organizations, locations, and sequences of nouns (nounruns). We chose this set of objects because they seem to be intuitively important to the story. That is, they are the "who" and the "where" of the events reported in the story.

We extract people, organizations, and locations from text using a named-entity tagger. Named-entities are specific references to people, organizations, and locations. For example, "James Brown" is a named-entity, but "he" is not. Named-entity taggers have been developed through work done for the Message Understanding Conference (MUC, 1998). They tag the named-entities found in the text and also specify whether the named-entity is a person, organization, or a location. Named-entity extractors extract more named-entity types than people, organizations, and locations. Some of the other types are dates and numbers. We choose to ignore these other object types, as they do not seem as intuitively important to the story.

Since there is a lot of information present in stories not found in named-entities, we will also extract nounruns. A nounrun is a sequence of one or more nouns. Some examples of nounruns are "airport", "credit card", and "rock music station". In order to reduce the variations of form of equivalent nounruns, we construct them from the stemmed version of the nouns. This way, the nounruns constructed for "rock music station" and "rock music stations" would both be "rock music station".

2.4 Coreference Resolution

Coreference resolution is often desirable in situations when information is extracted from text. Coreferences are variations in the phrase used to refer to an object. The construction of LORs is an information extraction task. Grishman (1997) includes coreference resolution as an important step in information extraction, and we consider coreference resolution's importance in our system. There are several types of coreferences of interest. One is the use of pronouns to refer to a named-entity. For instance, if "James Brown" is mentioned in text, the object may later be referred to as "he" in the text. Another type consists of variations on named-entities or nounruns. An example of this type of coreference is "James Brown" and "Mr. Brown". There are other types of coreferences that we do not attempt to resolve. One type similar to pronouns is the use of phrases such as "the musician" or "the company" to refer to a named-entity. Acronyms are another type we do not consider. Performing coreference resolution perfectly is a difficult task, as it would require some degree of natural language parsing and perhaps understanding. However, we will investigate simple techniques that we conjecture perform reasonably and evaluate their use. We only consider coreference resolution within one story, as opposed to performing coreference resolution across stories.

2.4.1 Pronouns

We use a simple approach to resolve pronoun coreferences. The pronouns of "he", "she", "I", "me", "him", "her", and "my" are mapped to the previous person named-entity found in the text. The pronouns of "we", "us", "they", "our", and "their" are mapped to the previous organization named-entity found in the text. This can give both good and bad results. Consider the sentences "James Brown was recently interviewed by Craig Kilborn. He answered questions about his musical influences." We would incorrectly replace "He" and "his" in the second sentence with "Craig Kilborn". On the other side of the coin, consider these sentences: "Craig Kilborn said 'I recently interviewed James Brown. He discussed his musical influences."' In this case, we would replace "I" with "Craig Kilborn" and "He" and "his" with "James Brown," all of which would be correct replacements. We will investigate the performance of this approach in the evaluation section.

2.4.2 Substring Matching

We attempt to resolve variations in object references through substring matching. Within a named-entity type, we treat all objects of that type where one is a substring of the others as the same object. For example, if "James Brown", "Mr. Brown", and "Brown" all occurred in a story, we would consider them to be the same object: "Brown". However, if "James Brown" and "Mr. Brown" occurred in a story, but "Brown" did not, we would not treat "James Brown" and "Mr. Brown" as the same object. We do not do the substring match for objects of different types (the types are people, organizations, locations, and nounruns). Or if we encounter the phrase "Brown, founder of Brown Publishing", we would not treat "Brown Publishing" as "Brown" because "Brown Publishing" is an organization and "Brown" is a person. We also use this method for nounruns. In a story about credit cards, credit cards may be referred to as both "credit cards" and simply "cards". This resolution method

Original		*Resolved*	
type	*object*	*type*	*type*
person	James Brown	person	Brown
person	Brown	person	Brown
person	Craig Kilborn	person	Craig Kilborn
person	Mr. Kilborn	person	Mr. Kilborn
nounrun	credit card	nounrun	card
nounrun	busines card	nounrun	card
nounrun	card	nounrun	card
location	Washington	location	Washington
location	Washington DC	location	Washington DC
person	George Washington	person	George Washington

Table 1: Good and bad examples of substring matching.

would treat the "credit card" and the "card" nounruns as the same objects ("card," not "cards," because we use the stem in nounrun construction). Table 1 shows some examples where the substring matching works and fails. The first example is likely to be a correct resolution. The "Craig Kilborn" example is a case where substring matching misses a coreference, and the "card" example is a case where substring matching produces undesirable replacements. The final example illustrates that the resolution is not performed across object types.

2.5 Relationship Creation

After we have extracted the objects of interest, we create the relationships between the objects. We focus on extracting relationships based on distance of objects in text. This will not give us any type information on the relationships. However, creating relationships based on distance of objects is easy to compute. Also, it is unclear how to use typed links. We define the distance between two objects as the number of words and objects between the objects. Table 2 helps illustrate this. We treat the "James Brown" object as taking only one position. The words "to" and "a" are stopped, so they do not take any positions. To find the distance between "James Brown" and "audience", we subtract the position of the "audience" object from the

Position	*Word*	*Object*
1	James	James Brown (1 of 2)
1	Brown	James Brown (2 of 2)
2	sang	
-	to	
-	a	
3	large	
4	audience.	audience (1 of 1)

Table 2: Computing the distance of two objects.

position of the "James Brown" object, getting a distance of 4 - 1 = 3. Whenever we compute a distance between objects, we subtract the lower position from the higher position, giving us positive distance at all times. In most of our tests, we will only create a relationship if the objects have a distance of five or less. We will keep track of the distance when we create a relationship.

3 Topic Tracking

3.1 Topic Detection and Tracking

Topic Detection and Tracking (TDT) is a project with the goal of better organizing information found in a stream of news. See the summary article (Allan *et al.*, 1998) for more information. We will consider the task of topic tracking.

3.2 Approaches to Tracking

The goal in topic tracking is to track a known topic. That is, the system is given a few example stories of a topic, and as more news stories are presented to the system, the system makes a judgment as to whether or not each story is on the same topic as the example stories. The number of example stories is denoted N_t. In addition to providing a hard YES/NO judgment for each story, the system also provides a number corresponding to the whether the story is on-topic. The higher the number associated with an story, the more certain the system is that the story is on the same

topic as the example stories. There is no fixed range for these numbers. The hard YES/NO judgments are chosen by setting a threshold, θ, and marking an story as YES if the score for the story is greater than θ and marking an story NO otherwise.

3.3 Comparing Stories to Topics

In order to generate a score for whether the story is on-topic, the system needs to compare the story to some representation of the topic. Since the system is given only N_t example stories of the topic, the representation is constructed from the N_t example stories. If a system can produce a similarity score for any two stories, that scoring method can be used in several ways. Two of the common approaches are centroid (agglomerative clustering) and k-nearest neighbor (Allan *et al.*, 1998). We use the centroid approach in our evaluation.

In the centroid approach, we build a representation of a story that is the "middle" of the example stories. One way to do this is to concatenate the stories together, making one story out of the examples. Another way is to take the system's representation of the example stories and average them. This requires knowledge of the system's representation and of how the system compares two stories. The system then compares the centroid to the new story to generate a score that the new story is on-topic. We compute the centroid by averaging the representations. The system may additionally normalize the score by dividing it by the average of the scores that each example story is on-topic. This helps to normalize scores across system runs for different topics.

One criticism of the centroid approach is that averaging the representations of several stories may not make sense. For instance, one could assign each month of the year with a number, and compute the average birth month of a group of people. However, there is some evidence to show that averaging a document does not behave poorly compared to other techniques.

3.4 Comparison Measures

The similarity function is a very important component to a topic tracking system. Not only does it generate similarity measures between stories, it gives constraints on what information in the stories is used and how that information is used. We consider the cosine similarity measure and a new approach that attempts to make better use of LORs.

3.4.1 Cosine Similarity and $tf \cdot idf$ Weighting

Cosine similarity is a standard measure used in information retrieval for comparing stories (van Rijsbergen, 1979). In this approach, a story is treated as a vector. Each of the components in the vector is assigned a weight. The weights are chosen so that the higher the weight, the more important the component is to the story. The cosine similarity between two stories is the inner product of the two vectors normalized to unit length. If p and q are vectors representing two stories, and p_i is the weight of the feature i in p, then the formula for cosine similarity is:

$$cossim(p, q) = \frac{\sum p_i q_i}{\sqrt{\sum p_i^2 \sum q_i^2}}$$

The components in the vector are traditionally words present in the story. These words are stopped and stemmed. The weights on components are given using a scheme called $tf \cdot idf$. There are many variations on exactly how $tf \cdot idf$ is computed. We will use the formulas used by UMass for TDT (Allan *et al.*, 2000). This is the Okapi weighting scheme (Robertson *et al.*, 1996). The tf component is a measure that indicates the importance of each term, or stemmed word, to the story. The abbreviation tf is short for term frequency. Frequency typically means the number of times an event can be observed during a period of time. We do not make use strict interpretation of the word frequency. Term frequency loosly means the number of times a word occurs in a story. The idea is that the more times a word occurs in a

story, the more important it is to the story. The formula for tf that we will use is given by:

$$tf = \frac{rawtf}{rawtf + 0.5 + 1.5\frac{len_d}{len_{avg}}}$$

where $rawtf$ is the number of times the term occurs in the story, len_d is the length of the story (number of words excluding stopped words), and len_{avg} is the average length of the stories in the story set. This gives a tf score that ranges from zero to one, with more commonly occurring words receiving a higher weight. The $\frac{len_d}{len_{avg}}$ component helps to normalize the weights across stories of different lengths. The abbreviation idf stands for inverse document frequency. Again, we use the word frequency loosly. This component gives higher weights to words that aren't common in the story set and lower weights to words that occur often in the set of stories. The idf is defined as:

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

where df is the number of stories the term occurs in and N is the number of stories. The $tf \cdot idf$ weight for a term is simply tf times idf.

In the context of tracking, len_{avg}, N, and df are not known, as we do not know information about stories that have not come in yet. One method of accommodating for this is to calculate these numbers for the stories the system has seen up to the current story. Alternatively, len_{avg} and idf can be computed on a set of stories independent from the tracking set. In our tests, we use the true values of len_{avg}, N, and df. Since we are treating the $tf \cdot idf$ weighting scheme as a baseline to our performance, it is acceptable for us to use the true values.

3.4.2 Relationships in a Vector

In our system, we wish to evaluate the quality of relationships extracted from text. One way to do this would be to treat the relationships as features in a vector. Then,

we could take the cosine similarity of the vectors for stories and centroids as a similarity function. Instead of using $tf \cdot idf$ as weights in the vector, we will make use of the distance between objects in our weighting scheme. When creating a relationship between two objects, we kept track of the distance between the objects. To calculate the weight for a relationship feature in a story, we treat the distances, d_i, on the relationships as resistors. We then take the net resistance if the resistors were in a parallel circuit. This gives a net resistance ranging from zero to the maximum distance, $maxdist$, allowable on a relationship, with more frequent and close relationships having a resistance closer to zero. In order to adjust this number to be usable in the cosine similarity function, we need higher weights to be more important. To do this, we subtract the net resistance from $maxdist$. The formula for this is:

$$maxdist - \frac{1}{\frac{1}{d_1} + \frac{1}{d_2} + \cdots + \frac{1}{d_n}}$$

where n is the number of that type of relationship. For example, if the relationships were created with $maxdist = 5$ and the objects "Bill Gates" and "Microsoft" occurred in a story twice, both times with four words between them, the weight on the feature for "Bill Gates is related to Microsoft" in the vector for that story would be $5 - (1/(1/4 + 1/4)) = 5 - (1/(1/2)) = 5 - 2 = 3$. Note that this sets a weight of zero on a link if it occurs only once with $d = maxdist$. This approach to weighting the features gives results similar to tf. It does not capture any notion of inverse document frequency. This method does not make much use of the relationships. It requires the presence of specific relationships to exist. This greatly reduces the number of similarities across stories. We create a vector for the topic by averaging the vectors of the example stories.

3.4.3 "Centers" Approach

In an attempt to use the relationships in a more flexible manner than treating reach relationship as a feature in a vector, we developed a new comparison measure. LORs

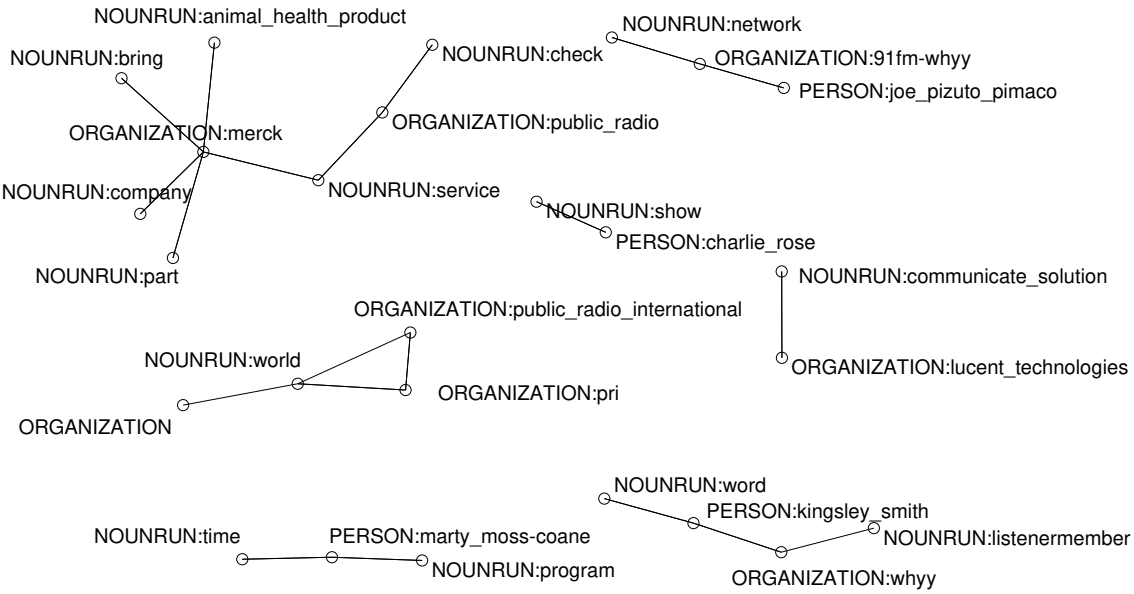

Figure 1: Graphical representation of an LOR.

can be viewed of as a graph where the nodes are objects and lines exist between objects if they are related, or linked. Figure 1 is the graph of a LOR for a story. Viewing LORs as graph leads us to think about social network analysis. The "centers" method compares two stories by comparing the objects, or nodes on a graph, within the stories. Rather than just looking at the objects themselves, we look at what the objects are connected to. That is, we try to capture some notion of role equivalency among two objects. Role equivalency in social network analysis tries to measure how well one node in a graph could replace another node (Wasserman & Faust, 1994). Instead of using a social network analysis measure of role equivalency, we use the dot product, or cosine similarity, to determine role equivalency. Since we do not wish to rely solely on the role equivalency of two objects, we provide a little weight to what the objects themselves are. The weights we place on the importance of role equivalency and matching objects are somewhat arbitrarily chosen, and as of yet we have not been able to investigate the use of different weights. We place most of the importance on the role equivalency, because we feel that it captures more information about the object than the object value or type itself.

We now go into the specifics of the "centers" method. We represent story by a set of object-vector pairs. The objects in the pairs are the objects in the LOR we create for the document. The vector paired with an object is constructed from the objects linked to the object. The weights on the objects in the vector are taken from the weights on the links, as calculated using the resistance technique in Section 3.4.2. We call an object-vector pair a center. Figure 2 may help clarify the construction of a vector paired with an object. The nodes in the chart are objects, while the lines between nodes indicate a link between two objects. The arrow points to the object of type nounrun with the value of plan. We build the vector for the center with object nounrun:plan using the connected objects. We have circled them on the graph. Objects with no relationships are discarded. If C is the set of centers for a story, we call c_i a center. The object associated with c_i is denoted as $c_i.object$, and the vector paired with the object is $c_i.vector$. Figure 3 gives a graphical representation a story and its components. In the figure, whenever a word or number field is used, we provide example values. In order to generate the topic representation, we average the weights on the relationships present in the example stories and create the vectors as above.

To compare the representation of a story to that of a topic, let c and d be the sets of centers constructed for the topic and the story, respectively. For each c_i, compare its vector, $c_i.vector$ using cosine similarity to each $d_j.vector$. Multiply that by one minus the weight of the center, $1 - w$. Calculate the similarity of the center objects $c_i.object$ and $d_j.object$, multiplying it by w. Take the max of this sum over all c_i. Then average the maximum similarities across all d_j. The formula for this is:

$$sim(c, d) = \frac{\sum_{j=1}^{n} max(csim(c_1, d_j), csim(c_2, d_j), \ldots, csim(c_m, d_j))}{m}$$

where

$$csim(c_1, c_2) = (1 - w)cossim(c_1.vector, c_2.vector) + w \cdot osim(c_1.object, c_2.object)$$

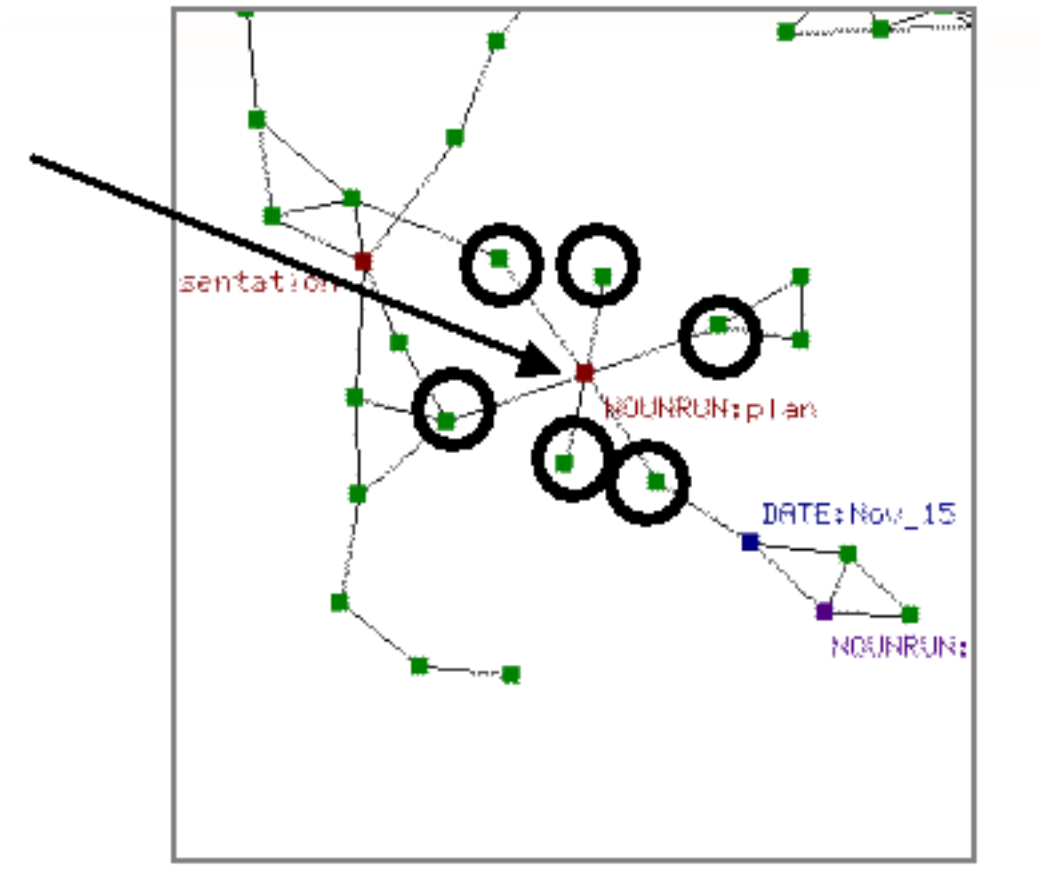

Figure 2: Vector construction for a center.

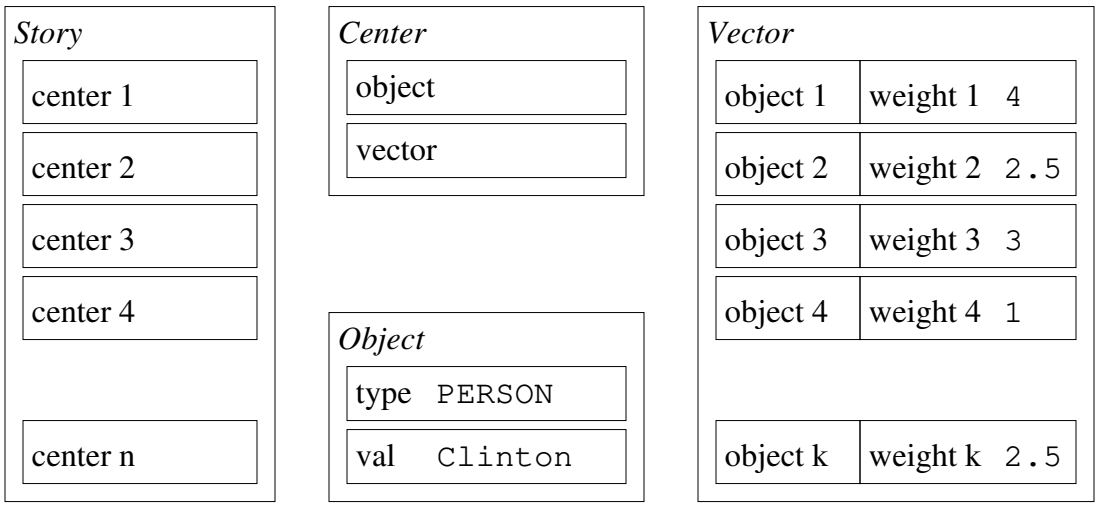

Figure 3: Story representation in "centers" algorithm.

$osim(o_1, o_2) =$

1 if $o_1.type$ equals $o_2.type$ and $o_1.val$ equals $o_2.val$

0.1 if $o_1.type$ equals $o_2.type$ but $o_1.val$ does not equal $o_2.val$

0 otherwise.

m is the number of centers in c, n is the number of vectors in d. Figure 4 shows the computation of $sim(c, d)$. For every center in c and d, we compute their center similarity using csim and store the value in a table. We then select the max csim score for each center in d. In the figure, that means taking the max in each row. We have circled the max scores in the figure. Notice that this allows more than one center in d to be paired with one center in c. $sim(c, d)$ is then the average of the circled scores. Since the similarity measure is not symmetric, if we were to compare a story to another story we might wish to average $sim(c, d)$ with $sim(d, c)$.

4 Implementation and Efficiency

The methods we have developed require more computation than the traditional techniques. Thus we are concerned not only with the quality of the results, but also the feasibility of running the algorithm on large amounts of information. In this section we will discuss the efficiency of the algorithms for performing extraction and tracking.

4.1 Extraction

The data set we worked with for evaluation already had stopping, stemming, part-of-speech tagging, and named-entity extraction performed on it. However, as this is not always the case, we will discuss the efficiency of performing these tasks. For each of these tasks, we use existing systems. Stopping, stemming, and part-of-speech tagging can be done on a story in time linear to the number of words in the story. The extraction system can use any named-entity recognizer that outputs according

Story c

Story d / csim	c1	c2	c3	c4	c5	c6
d1	0.3	**0.5**	0	0	0.1	0
d2	**0.4**	0	0	0.25	0	0.1
d3	**0.6**	0	0	0.5	0.1	0
d4	0.1	0	0	0.1	**0.2**	0
d5	0	0.3	**0.4**	0	0	0.1
d6	0	0.1	0.1	**0.2**	0	0

Figure 4: Computing sim(c, d)

to the format developed in the Message Understanding Conference. We have successfully used both BBN's named-entity recognizer and one developed at UMass.

Finding nounruns in text is a simple task once we have the part-of-speech tagging information included with the story. It requires only one pass through the story. We do not outline the algorithm we use here, as it is a straightforward task.

We wrote our own code to perform the simple coreference resolution techniques we discussed earlier. Our code does pronoun coreference resolution in a single pass of the story. As the code progresses through the story, it remembers the last person and organization seen in the text. When it encounters a singular pronoun, it marks the pronoun with the previous person. When it finds a plural pronoun it marks the pronoun with the previous organization.

The substring method of coreference resolution takes somewhat more time than the pronoun resolution technique. In the worst case, a naive algorithm could take $O(n^3)$ time, where n is the number of words in the story, to fully resolve the objects. We do not do complete resolution. The algorithm we use is given below.

`For each object in the story, set` $resolve\{object\} = object$.

$resolve$ is a hashtable where the key is the object that we wish to resolve and the value is the object we resolve the key to.

`For every object` o_1 `in the set of keys of` $resolve$`,`

`For every object` o_2 `in the set of keys of` $resolve$`,`

`if` $o_1.type$ `equals` $o_2.type$`, and`

$o_2.val$ `is a substring of` $o_1.val$`, and`

$resolve\{o_1\}.val$ `is longer than` $o_2.val$

`then set` $resolve\{o_1\} = resolve\{o_2\}$.

In an effort to save time, our algorithm does at most n^2 steps. While this may not fully resolve the objects, the objects generally consist of only a few words, so this algorithm should at worst miss a few possible resolutions.

The time it takes to construct the relationships is dependent on more than just the length of the story. It also depends on *maxdist*. If *maxdist* is close to the length of the story, it can require around n^2 steps. If *maxdist* is constant and small, then it only requires around $maxdist \cdot n$ steps.

4.2 Tracking

The time for performing a tracking run is $t \cdot d \cdot c$ where t is the number of topics, d is the number of stories, and c is the time it takes to compare a story to a topic. For cosine similarity of vectors, c is the number of features in the two vectors, which is at worst linearly related the lengths of the stories in the topic and the story being compared. The time for the "centers" algorithm is $O(kf^2)$, where k is the average number of objects an object is related to (or the connectivity of the node) and f is the number of objects (or $f$eatures). k is influenced by *maxdist*; as *maxdist* grows, so will k. f is related to the lengths of the stories in the topic and the story being compared. f and k are also influenced by the method of LOR creation used. If *maxdist* is large, the c could be cubicly related to the lengths of the stories. In order to run the "centers" algorithm in a reasonable amount of time, we need to work with a small *maxdist*.

Getting efficient performance for the "centers" algorithm requires very quick lookup of values. For the operations we have to perform, it is simplest to store stories and topics in separate files.[1] During a tracking run, all of the topics are loaded into memory. Stories are then loaded one at a time to be compared to topics. Normalization of scores is performed after the tracking run is completed, but takes very little time.

[1]We spent a large amount of time examining various LOR storage techniques. We found that storing LORs database was not appropriate for storing LORs to be used in either vector comparison or "centers" comparison.

		truth	
		on-topic	off-topic
system	on-topic		false alarm
	off-topic	miss	

Table 3: False alarm and miss.

5 Evaluation

5.1 Evaluation Set

For evaluation, we use the TDT2 train.ccap set. It is a set of 8032 stories compiled from closed-captioned text of news broadcasts of ABC, CNN, and PRI. We track 37 topics over the set of stories. The number of training stories per topic, N_t, is set at four. We ignored the topics with less than four on-topic stories in the set.

5.2 DET curves

In our evaluation, we want to examine how well our systems match the true answers. We know the underlying distribution of on/off-topic status for all stories for each topic in our evaluation set. We can do this by measuring the false alarm rate and miss rate. A false alarm occurs when a system reports a story as on-topic when it is off-topic. A miss occurs when a system declares a story as off-topic when it is on-topic. Table 3 illustrates this. For a given topic, false alarm rate is the number of off-topic stories that a system declared on-topic divided by the number of stories the system declared off-topic. Miss rate is the number of on-topic stories the system declared off-topic divided by the number of on-topic stories. The formulas for the error rates for a given topic t are:

$$\text{false alarm}(t) = \frac{\text{number off-topic judged on-topic}}{\text{number off-topic}}$$

$$\text{miss}(t) = \frac{\text{number on-topic judged off-topic}}{\text{number on-topic}}$$

Our systems do not give hard judgments as to whether an story is on-topic or off-topic. Instead, we output a number: the higher the number, the more certain we

are that the story is on-topic. On order to convert a number into a strict on-topic or off-topic judgment, we set a threshold. If the score is higher than the threshold, we say that the story is on-topic. Otherwise, we say the story is off-topic. The threshold we choose affects the false alarm and the miss rate. Higher thresholds can decrease the false alarm rate but it will increase the miss rate. We find it desirable to construct a graph of false alarm and miss rates over all thresholds. This way, we can illustrate the tradeoff between false alarm rate and miss rate.

Detection error tradeoff (DET) curves give an appropriate way to graph this information. See (Martin *et al.*, 1997) for a more thorough description of DET curves. The TDT program uses DET curves as a standard evaluation tool. DET curves place false alarm probability along the x-axis and miss probability along the y-axis. As in (Martin *et al.*, 1997), instead of fixing the scale along the axes to the units of probability, we use the normal deviates of the probabilities to fix the scale. This does a few things to the DET curves. First, if the distributions of scores given to on-topic and off-topic stories are relatively normal, the curve will appear fairly straight. Second, this stretches out the lower ends of the graph. This enables easy distinction in regions of low error levels.

So far, we have described how to generate a DET curve for one topic. However, we are interested in the system performance across many topics. There are two common ways to construct a DET curve for multiple topics: macro averaging and pooled averaging. For a given threshold, the two methods construct false alarm and miss rates as follows:

Macro:

$$\text{false alarm}_{macro} = \frac{\sum_{t \in \text{T}} \text{false alarm}(t)}{\text{number of topics in T}}$$

$$\text{miss}_{macro} = \frac{\sum_{t \in \text{T}} \text{miss}(t)}{\text{number of topics in T}}$$

Pool:

$$\text{false alarm}_{pool} = \frac{\sum_{t \in \mathrm{T}} \text{number off-topic judged on-topic for topic } t}{\sum_{t \in \mathrm{T}} \text{number off-topic for topic } t}$$

$$\text{miss}_{pool} = \frac{\sum_{t \in \mathrm{T}} \text{number on-topic judged off-topic for topic } t}{\sum_{t \in \mathrm{T}} \text{number on-topic for topic } t}$$

where T is the set of topics. Macro averaging gives equal weight to every topic, while pooled averaging give more weight to topics with larger numbers of on-topic stories. In the evaluation, we will only use pooled averaging as the code for macro averaging, meaning that large topics will be heavily weighted.[2]

5.3 Scatter Plots

We are also interested in whether two systems give similar answers. We will use scatter plots to visually investigate correlation of scores among systems. If we plot every judgment the systems make for the evaluation set we are using, a scatter plot would have close to 300,000 data points. In order to reduce the number of points on the plots, we will randomly sample approximately 5% of the off-topic stories and 10% of the on-topic stories to display. We sample a larger percentage of on-topic stories because there are many more off-topic stories then on-topic stories. If we presented only 5% of the on-topic stories, we would not get a good sense of the correlation of scores given for on-topic stories.

6 Results

In this section we present the results of our tests. We first present some baseline tests using vectors and cosine similarity. We then present results using the centers method of comparing stories. We discuss the usefulness of various extraction techniques in the context of the "centers" method. Finally, we compare the "centers" method to the

[2]It is preferable to use macro averaging because all topics are given equal weight and it is possible to create 90% confidence intervals for the curves. We did not do this in our evaluation because the evaluation code for macro averaging was unavailable to us at the time.

baselines. In all of our tests, topics are created by the centroid method by averaging the representations of the example stories.

6.1 Baselines

We first examine tracking runs performed with cosine similarity and Okapi $tf \cdot idf$ weighting. Figure 5 shows the results for various methods of feature selection. It is not surprising that the best performing system is the one that uses all words as features. The other two lines show the feature selection using named-entities and nounruns. No coreference resolution was performed for these tests. The difference between the use of named-entities and nounruns verses the use of just nounruns is somewhat surprising. While the feature selection for both make use of only nouns, the difference between the lines can be explained by the fact that grouping of nouns is different. It is likely that there are better matches of named-entities across stories than nounruns.

6.2 "Centers" Algorithm and Variations on Extraction

We now analyze different extraction techniques in the context of the "centers" algorithm. Unless stated otherwise all relationships are created with $maxdist = 5$. First, we investigate whether the simple coreference resolution techniques described earlier impact tracking. All of the tests shown in Figure 6 use relationships constructed using named-entities and nounruns. Relationships with both objects as nounruns were removed. We tested use of substring method, pronoun method, and both methods of coreference resolution. The graph shows that we get no gain performing coreference resolution. For the rest of the evaluation we will not use coreference resolution.

Next, we investigate removing relationships where both objects are nounruns. The intuitive thought behind removing them is that named-entities seem more important to the story. Additionally, there will be much fewer relationships, reducing

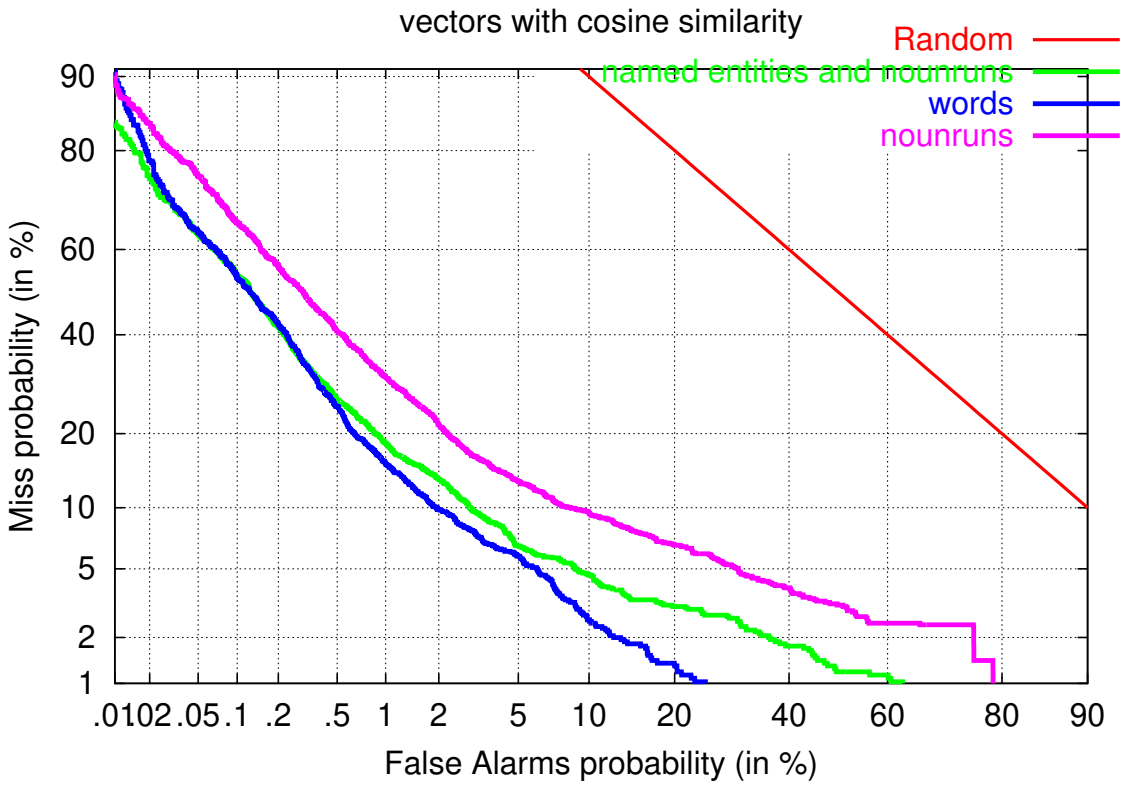

Figure 5: Vectors with $tf \cdot idf$ weighting.

the running time of a tracking run. Figure 7 shows that there is little difference in the performance when relationships with both objects as nounruns are removed. The figure also has a line for using only nounruns. Note that the light blue "no coreference resolution" line of Figure 6 is the same as the purple "named entities and nounruns with trimming" line in this figure. As with the vector based approach, the use of named-entities improves over the use of part-of-speech tags in the construction of objects.

It seems wasteful to not use information found in verbs, so we have tests where we created relationships using words. We do not perform named-entity extraction or part-of-speech tagging. All words are considered objects. The words we use are stopped and stemmed. Using words improves results some (Figure 8), but it creates more relationships and takes longer for the tracking test to run. Similarly to the

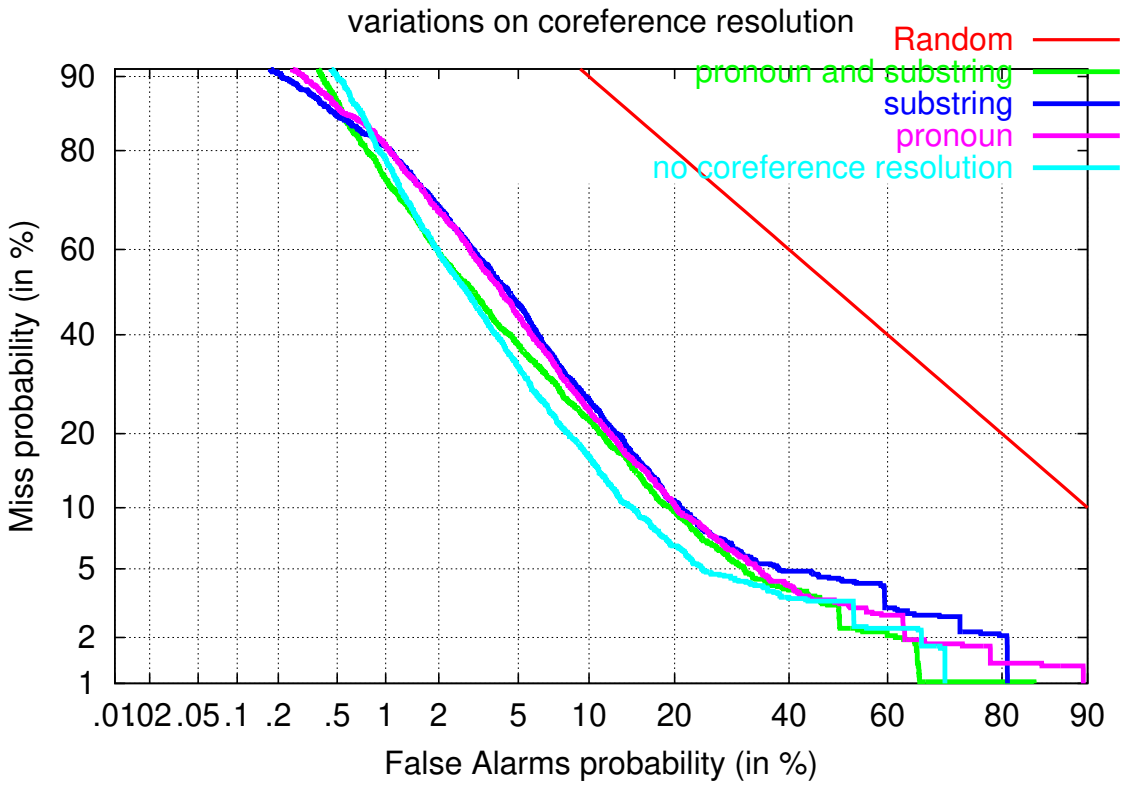

Figure 6: Variations of coreference resolution used for the "centers" algorithm.

pruning done with nounrun relationships, we have a test where we use named-entities, words, and remove relationships where both objects are words. Again, we see that doing this does not hurt performance.

We next compare the "centers" algorithm to the vector model with relationships as features. Figure 9 shows the performance for two methods of relationship creation. One method creates relationships between named-entities and nounruns with nounrun-to-nounrun links trimmed. The other method creates relationships between named-entities and words with word-to-word links trimmed. We also include the corresponding "centers" algorithm performance. The lines for the vector representations end abruptly because there are many vectors with no overlap, even with on-topic stories. We see that performance for the "centers" comparison method is worse than the vector representation with cosine similarity. As with the "centers" comparison

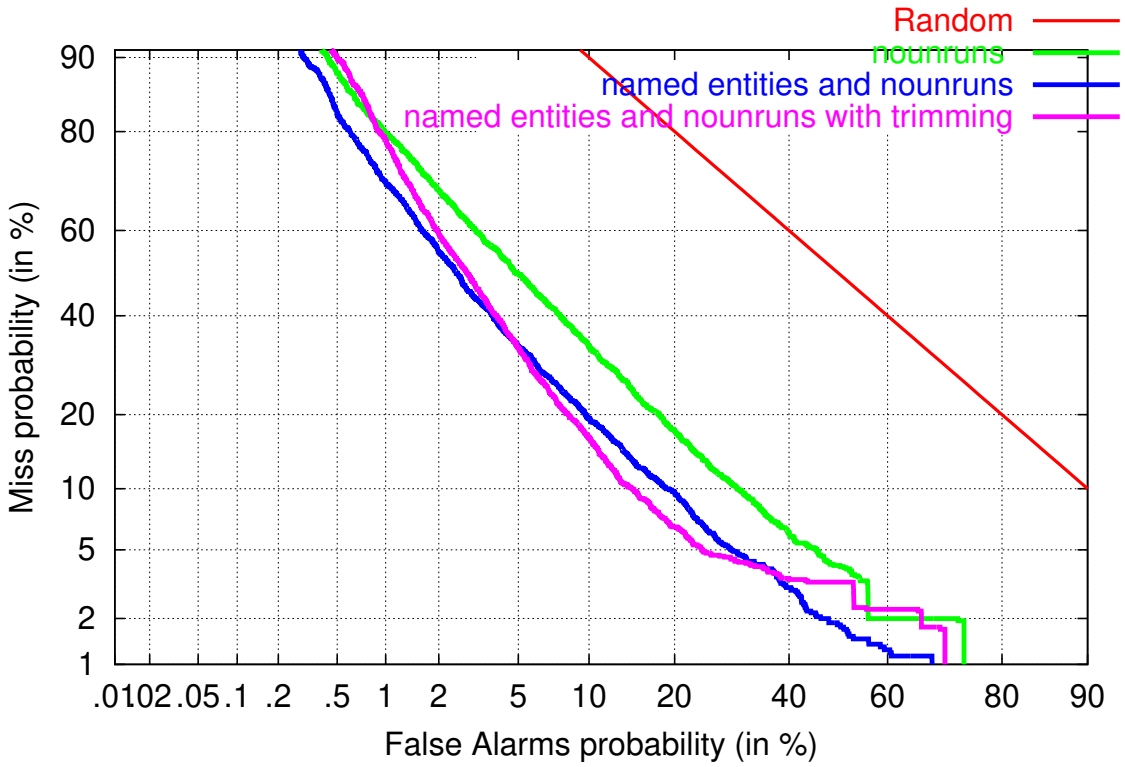

Figure 7: Trimming and named entity information.

method, the named-entities and words is better than named-entities wand nounruns using the vector representation and cosine similarity. This suggests that the conclusions made about extraction and coreference resolution would hold for cosine similarity where the features in the vector are relationships. Figure 10 shows us that the "centers" comparsion method gives answers different than using the relationships components in a vector using cosine similarity.

Similarly, Figure 11 illustrates that the "centers" method gives answers different from using the objects as features in a vector with $tf \cdot idf$ weighting. The objects used were named-entities and nounruns and the relationships used had nounrun-to-nounrun links trimmed. If we increase *maxdist* to the length of the story, it seems plausible that the system would give answers similar to the vector based method with $tf \cdot idf$ weighting. This could happen because as *maxdist* approaches, the vectors

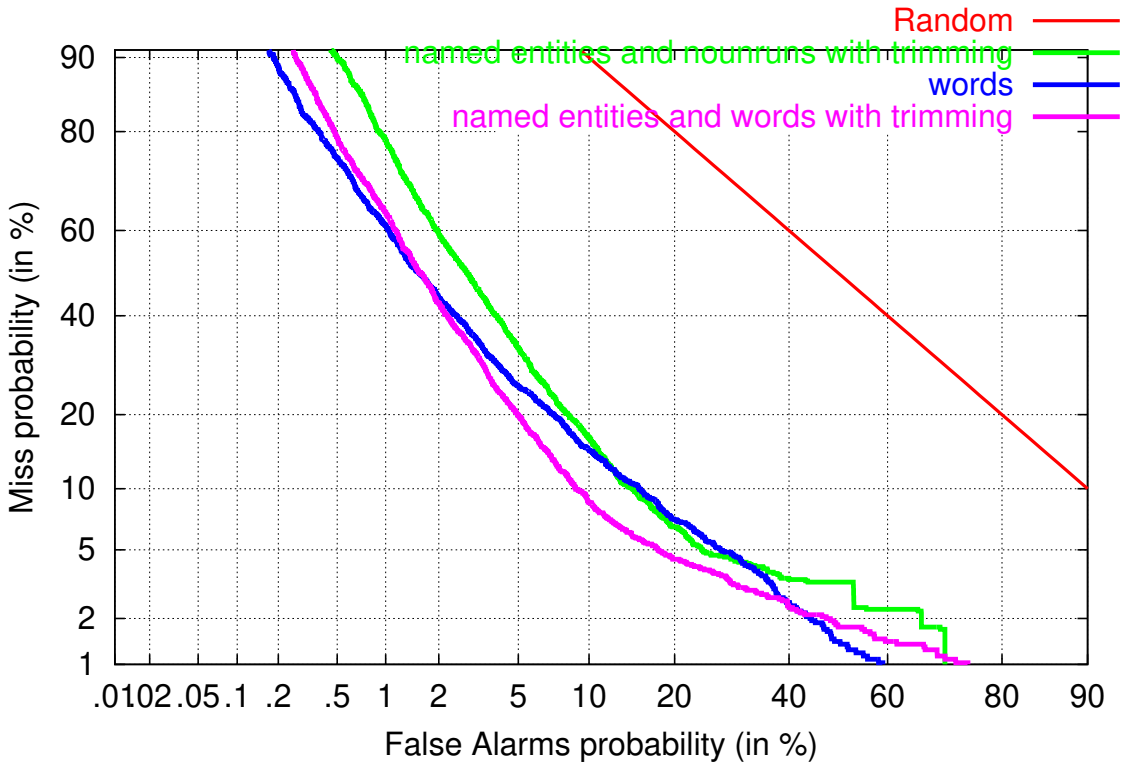

Figure 8: Words as objects.

created for the centers will have the same features as a single vector for the story. To investigate this, Figure 12 is a scatter plot where the vector based method is on the horizontal axis and the "centers" approach with $maxdist = 100$ is placed on the vertical access. In both cases, nounruns were used as features/objects. On-topic stories are marked by plus signs and off-topic stories are dots. We see that there is some correlation of scores between the two methods.

7 Related Work

There has been work on the use of named-entities for first-story detection (Allan *et al.*, 1999), another TDT task. They find, as do we, that use of only named-entities harms performance, as other important words found in text are disregarded. Additionally, they discuss the use of pairs of named-entities. This is very similar to

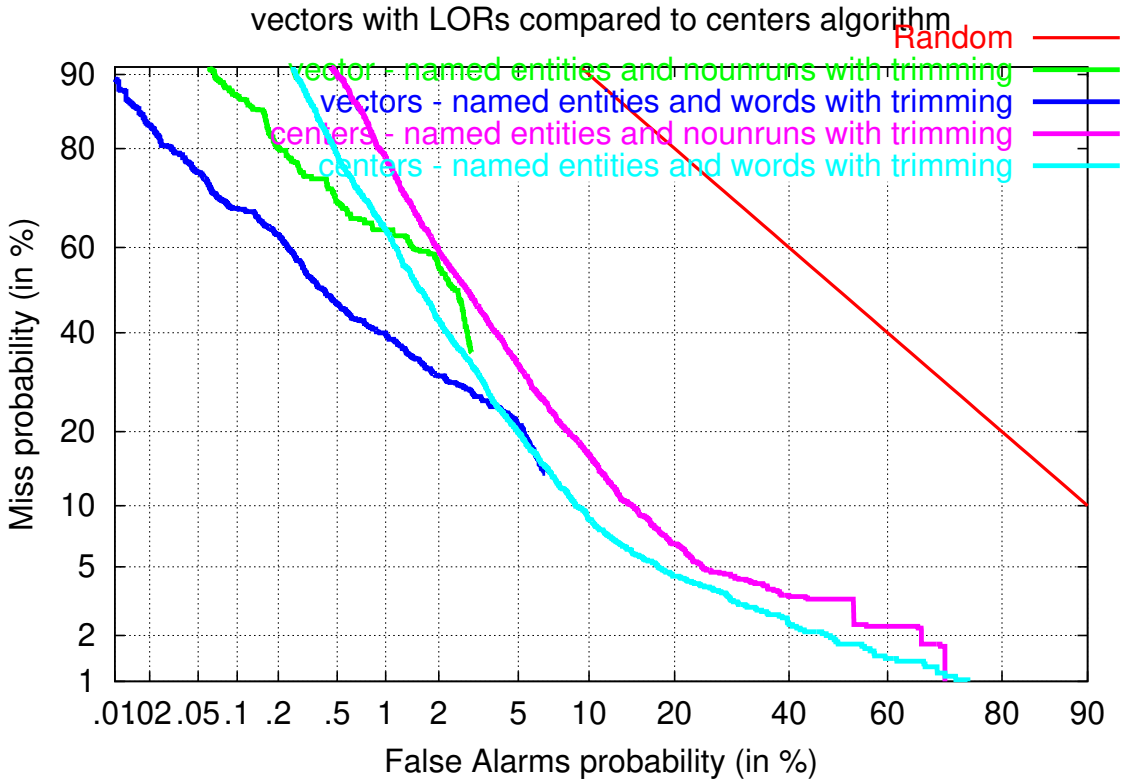

Figure 9: "Centers" algorithm compared to vector with features of relationships.

LORs where *maxdist* is large. Their use of the named-entity pairs is specific to first story detection, and would not necessarily be directly applicable to topic tracking. Our construction of LORs is different than the named-entity pairs because we allow relationships where an object is not a named-entity. We also place weights on the relationships, while the named-entity pairs had no weights.

Other forms of information extraction have been used for text classification. Riloff & Lehnert (1994) discuss the use of more sophisticated information extraction for high precision text classification within a specific domain. This work is reliant on manually engineered information for domains narrower than information found in news. We use extraction in a domain independent manner, so that manual engineering is not required.

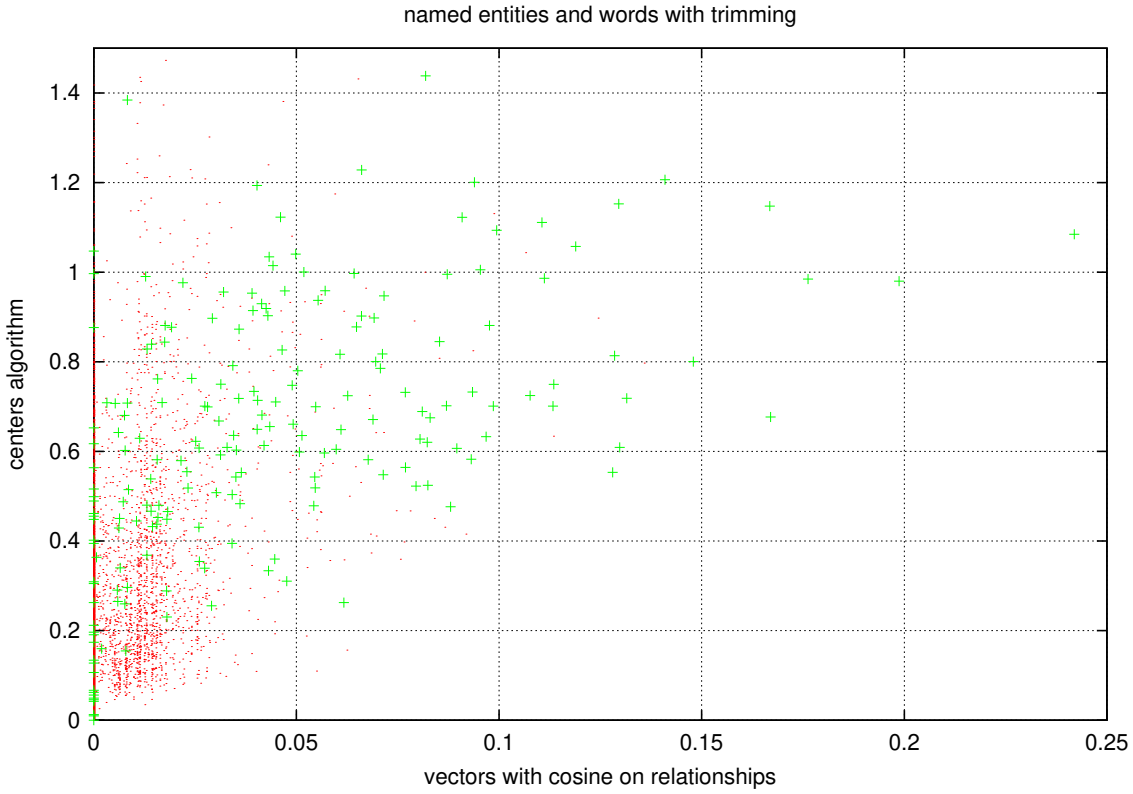

Figure 10: Scatter plot of "centers" algorithm verses vector with features of relationships.

Bigrams are similar to our LORs in a few ways. Bigrams can refer to two-word phrases, a pair of words found in the same passage, or a pair of words found in the same document. When bigrams refer to a pair of words in the same passage or document, this is highly related to LORs. A generalization of bigrams that refer to a pair of words in the same passage or document is called multi-word concepts. Multi-word concepts is a set of words that appear (or are required to appear in document, if the concept is in a query) within a n word window. The use of multi-term concepts for information retrieval has been investigated. The LORs presented in this paper are two-term concepts with a window size of *maxdist*. Papka & Allan (1998) show that for retrieval of documents, the use of multi-term concepts with small windows performs worse than with larger windows. We take the work further by including named-entities in multi-term concepts.

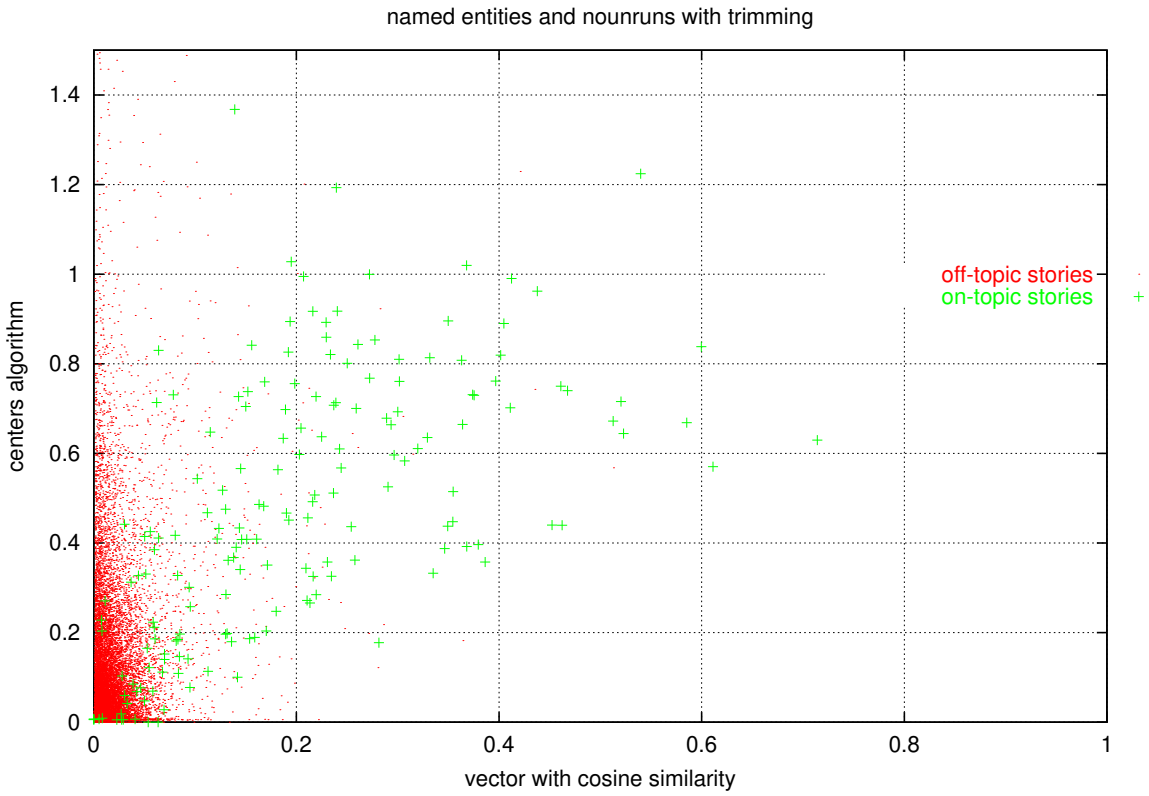

Figure 11: "Centers" algorithm gives different answers than vector with objects as features.

8 Conclusions and Future Work

We have learned about what information is useful in LORs. The use of named-entities with words allows the removal of word-to-word relationships without greatly changing the quality of the performance. Also, the use of naive coreference resolution does not improve performance. We also introduced a new text comparison measure that makes use of LORs. The "centers" method of comparison performs better than random and gives answers that are different than using cosine similarity on vectors.

It is not clear whether more accurate forms of coreference resolution would improve performance. Also, we did not look at resolution across documents, which might provide better resolution. We do not evaluate any methods for resolving phrases such as "the musician" or "the company". It would be possible to do a coreference resolu-

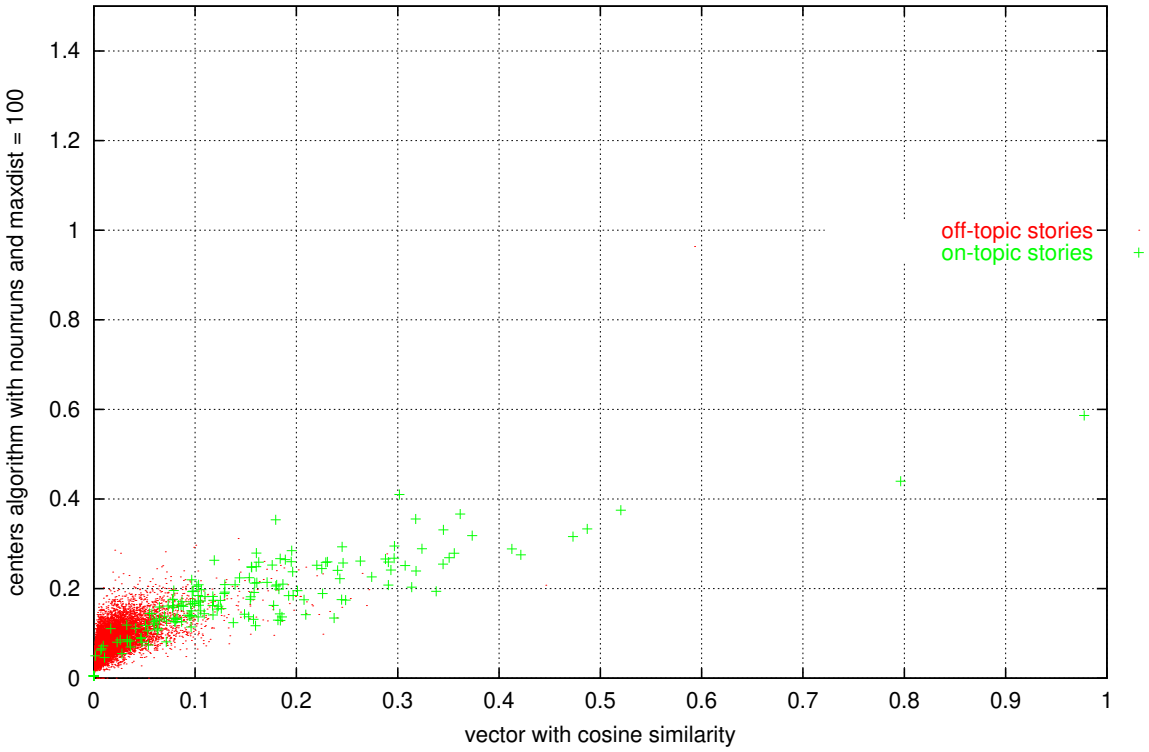

Figure 12: "Centers" algorithm approaches vector based techniques with large *maxdist*.

tion for this type in a manner similar to what we do for pronouns. We could keep a list of nouns that refer to a person and a list of nouns that refer to an organization. When we encounter a noun in either list, we could associate the appropriate person/organization with the noun in text according to which list the noun was found in. Another coreference type we do not consider is the use of acronyms in text. For instance, IBM may be referred to as both "International Business Machines" and "IBM" in a document. To resolve these types of coreferences, one could use an acronym database for assistance . If an acronym occurs in text, then one could look up the acronym in the database and check to see if any of the acronym's definitions also occur in the text (Larkey *et al.*, 2000). Then the acronym and definition could be treated as the same object.

The "centers" method for comparing text did not perform as well as existing techniques. However, we did little fine-tuning of the algorithm. One variation we would like to consider is instead of taking the maximum of the center similarity for an object in a document with the objects in the topic, force a match where an object in the topic can only be paired with one object in the document. Another variation would be to place more weight on the center similarity of objects with a large number of related objects. We did not test different parameters much. We could change the weights on the center object matching or change *maxdist*. We could impose restrictions on which objects are centers.

DET curves give us a way to visually compare the results of one system with another, but the DET curves we presented do not have confidence levels or test whether two systems perform significantly different. Confidence intervals can be computed when using macro averaging of topics.

9 Acknowledgments

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement numbers EEC-9209623 and EIA-9820309. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

Allan, J., J. Carbonell, G. Doddington, J. Yamron, & Y. Yang (1998). Topic detection and tracking pilot study final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop.*

Allan, J., H. Jin, M. Rajman, C. Wayne, D. Gidea, V. Lavrenko, R. Hoberman, & D. Caputo (1999). Topic based novelty detection. Final Report (1999 Summer Workshop at CLSP.

Allan, J., V. Lavrenko, D. Malin, & R. Swan (2000). Detections, bounds, and timelines: UMass and TDT-3. In *Proceedings of Topic Detection and Tracking Workshop (TDT-3).*

Grishman, R. (1997). Information extraction: Techniques and challenges. *Information Extraction (International Summer School SCIE-97).*

Larkey, L., P. Ogilvie, M. A. Price, & B. Tamilio (2000). Acrohpile: An automated acronym extractor and server. In *Proceedings of Digital Libaries 2000.*

Martin, A., G. Doddington, M. Ordowski, & M. Przybocki (1997). The DET curve in assessment of detection task performance. In *Proceedings of EuroSpeech '97*, pp. 1895–1898.

MUC (1998). *http://www.muc.saic.com/proceedings/muc_7_toc.html - Proceedings of MUC 7.*

Papka, R. & J. Allan (1998). Document classification using multiword features. In *Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM).*

Riloff, E. & W. Lehnert (1994). Information extraction as a basis for high precision text-classification. *ACM Transactions on Information Systems 12*(3), 296–333.

Robertson, S. E., S. Walker, S. Jones, M. M. Hancock-Beaulieu, & M. Gatford

(1996). OKAPI at TREC-3. In D. Harman (Ed.), *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*.

van Rijsbergen, C. J. (1979). *Information Retrieval* (Second ed.). Butterworths.

Wasserman, S. & K. Faust (1994). *Social Network Analysis: Methods and Applications*, pp. 487–491. Cambridge University Press.