Event Extraction from Heterogeneous News Sources

M. Naughton and **N. Kushmerick** and **J. Carthy**
School of Computer Science and Informatics, University College Dublin, Ireland
{martina.naughton, nick, joe.carthy}@ucd.ie

Abstract

With the proliferation of news articles from thousands of different sources now available on the Web, summarization of such information is becoming increasingly important. Our research focuses on merging descriptions of news events from multiple sources, to provide a concise description that combines the information from each source.

Specifically, we describe and evaluate methods for grouping sentences in news articles that refer to the same event. The key idea is to cluster the sentences, using two novel distance metrics. The first distance metric exploits regularities in the sequential structure of events within a document. The second metric uses a TFIDF-like weighting scheme, enhanced to capture word frequencies within events even though the events themselves are not known *a priori*.

Typical news articles contain sentences that do not describe specific events. We use machine learning methods to differentiate between sentences that describe one or more events, and those that do not. We then remove non-event sentences before initiating the clustering process. We demonstrate that this approach achieves significant improvements in overall clustering performance.

Introduction

As the amount of news articles available on-line dramatically increases, so too does the need to locate concise news articles quickly and efficiently. Due to the large number of news sources (e.g. Reuters, Aljzeera, CNN) that exist, multiple news articles are generated worldwide on a daily basis for a given news event. Following (Li *et al.* 2005) we define a news event as a specific thing that happens at a specific time and place. Such articles often contain duplicate information concerning the same event, but differ in choice of language used. In addition, specific details regarding the event may vary from source to source. For example, one article about a given bombing in Iraq may say *"at least 5 people were killed"* while a second may contain the phrase *"6 people were found dead"*. Our research focuses on merging descriptions of events from multiple sources to provide a concise description that combines the information from each source.

We decompose this problem into three sub-problems: (1) Annotation: identifying the spans of text in an article corresponding to the various events that it mentions; (2) Matching: identifying event descriptions from different articles that refer to the same event; and (3) Aggregation: converting the event descriptions into a structured form so that they can be merged into a coherent summary.

This paper focuses on the first sub-problem. Event annotation is challenging for several reasons. Many news articles describe multiple events. Moreover, sentences that refer to the same event are usually scattered through the article in no obvious pattern. A typical news article often contains sentences that do not describe any event(s). For example, *"At least 1,347 U.S. soldiers have died in Iraq since the war began"* and *"George Bush condemned acts of terror in Iraq"*. Figure 1 shows a sample article that illustrates these issues.

Specfically, in this paper we describe and evaluate clustering methods for the task of grouping sentences in a news article that refer to the same event. We generate sentence clusters using two variations of the well-documented Agglomerative Hierarchial Clustering algorithm (Manning & Schtze 1999) as a baseline for this task. We extend this approach by incorporating an important constraint associated with each sentence: the position of the sentence in the document. We also present a TFIDF-like weighting scheme that iteratively re-calculates term weights based on the previous iteration's clustering solution. Experimental results show that an increase in performance is achieved through the use the sequence-based and iterative TFIDF clustering algorithms.

In addition, we address the issue that many news articles contain sentences that do not refer to any event. We investigate a Machine Learning approach which classifies each sentence as one that references one of more events, or as one which does not. The idea here is to remove non-event sentences before initiating the clustering process. Experimental evaluation demonstrates that this preprocessing step improves clustering accuracy.

To summarize our empirical results, we demonstrate a modest increase in accuracy in the sequence and iterative-based approaches compared to a simple bag of words baseline. Our sentence classifier has an accuracy of 84%. By incorporating this classification phase into the clustering process we observe an average increase in accuracy of 8%.

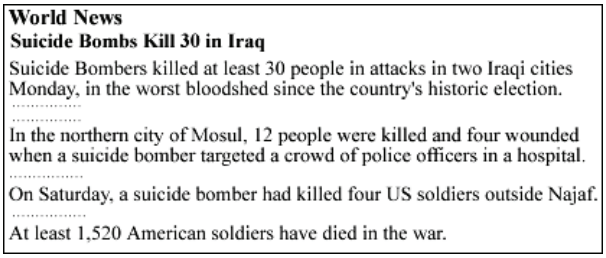


Figure 1: Sample news article that describes multiple events.

Related Research

Related research in this area falls under two main categories: multi-document summarization and sentence clustering. MEAD (Radev, Jing, & Budzikowska 2000), a centroid-based multi-document summarizer, generates summaries using cluster centroids produced by a topic detection and tracking system. It then selects a small number of sentences from each centroid to construct a multi-document summary. SUMMONS (McKeown & Radev 1995) is a system that summarizes a series of news articles on the same event. It extracts relevant information from different documents using an information extraction system to fill in a set of pre-defined templates and used the instantiated slots to summarize a series of news articles on the same event. However, SUMMONS adopts a more linguistic approach to determine what information to include in the event summary. The objective of our system is to distinguish the description of separate events within a single document and then to match the description of each event with corresponding event descriptions from other documents.

The task of clustering similar sentences has been investigated particularly in the area of text summarization. SimFinder (Hatzivassiloglou *et al.* 2001) is a clustering tool for text summarization that accomplishes the task of finding similar text units within a given document. However, they define a text unit to be a paragraph rather than a sentence. In addition, the text features used in their similarity metric are selected using a Machine Learning model.

Event Extraction as Sentence Clustering

We treat the task of annotating sentences with the event(s) they describe as a clustering problem. As a baseline, we generate sentence clusters using average-link and complete-link agglomerative clustering. Hierarchical agglomerative clustering (HAC) initially assigns each data point to a singleton cluster, and then repeatedly merges clusters until a specified termination criteria is satisfied (Manning & Schtze 1999). HAC clustering methods require a similarity metric between two sentences. As our baseline, we use the standard cosine metric over a bag-of-words encoding of each sentence. We remove stopwords and stem each remaining term using the Porter stemming algorithm (Porter 1997). No term weighting is used at this stage. Our algorithms begin by placing each sentence in its own cluster. At each iteration we merge the two closest clusters. A fully-automated approach must use some termination criteria to decide when to stop clustering. In our current experiments, we adopt the following termination methods:

1. **"correct k":** Halt the clustering process when the "correct" number of clusters remain. The value of k refers to the true number of distinct events in each news article. This value was obtained during the annotation process . describe in the evaluation section.
2. **"best k":** Halt the clustering process when the "best" k clusters remain. For this termination mode, we run the algorithm for each value of k and return the clustering solution that maximizes clustering accuracy.

Exploiting Article Structure

Our baseline ignores an important constraint on the event associated with each sentence: the position of the sentence within the document. Documents consist of sentences arranged in a linear order and near-by sentences in terms of this intrinsic linear order tend to be about the same topic (Zha 2002). Similarly we assume that adjacent sentences are more likely to refer to the same event, later sentences are likely to introduce new events, etc. In this section, we describe an algorithm that exploits this constraint during the sentence clustering process. To confirm the intuition that such latent structure exists, we treat each document as a sequence of event labels (one label per sentence, where each label is an integer that uniquely identifies each event). Using MDI (Thollard, Dupont, & de la Higuera 2000), we train a finite state automaton (FSA) from the sequences, where:

- **States** corresponded to event labels.
- **transitions** corresponded to adjacent sentences that mention the pair of events.

The automaton is stochastic: we count the number of each transition across a set of training documents (as well as the fraction of documents whose first and last sentences are labeled with each event). We calculate the probability that the trained automaton generated a given document as the product of the probability that the first sentence's event is an initial state, the probabilities of each transition in turn, and the probability that the last sentence's label is a final state. This procedure assumes that each sentence mentions at most one event. We explore six ways to deal with multi-event sentences. The simplest strategy is to simply discard such sentences. The most complicated strategy is to treat each multi-event sentence as a sequence of single-event mini-sentences, and then make a "copy" of each article sequence for each permutation of the mini-sentences. For example, the article sequence "1 {1,2}, 2, {2,3}" is mapped to four sequences: "1 1 2 2", "1 2 2 2", "1 1 2 3" and "1 2 2 3".

More formally, Let $\mathrm{L} = \{\mathrm{l}_1, \mathrm{l}_2 \ldots \mathrm{l}_n\}$ be a sequence of n event labels. We define $P(I(l_1))$ as the fraction of documents that start with event label l_1. Similarly, $P(F(l_n))$ is the fraction of documents that end with event label l_n and $P(l_{i+1}|l_i)$ is the fraction sentences labeled with l_i that are followed by sentences with label l_{i+1}. $P(L)$ is the probability that event sequence L is generated by the automata, and

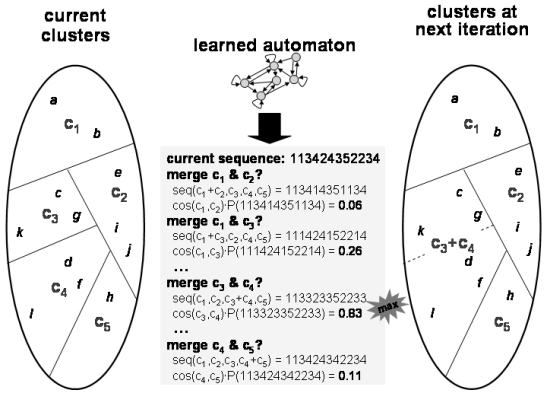


Figure 2: Sequence-based clustering process.

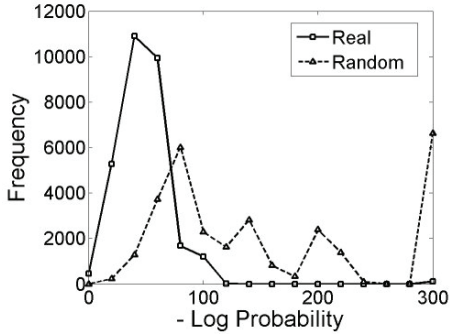


Figure 3: Distribution in the probability that actual and random event sequences are generated by the tuned FSA.

is defined as follows:

$$P(L) = P(I(l_1)) \times \prod_i P(l_{i+1}|l_i) \times P(F(l_n)) \quad (1)$$

We estimate how much sequential structure exists in the sentence labels as follows. The document collection was split into training and test sets. The automaton parameters were learned from the training data, and the probability that each test sequence was generated by the automaton was calculated. These probabilities were compared with those of a set of random sequences (generated to have the same length distribution as the test data). The probabilities of event sequences from our dataset and the randomly generated sequences are shown in Fig. 3. The test and random sequences are sorted by probability. The vertical axis shows the rank in each sequence and the horizontal axis shows the negative log probability of the sequence at each rank. The data suggests that the documents are indeed highly structured, as real document sequences tend to be much more likely under the trained FSA than randomly generated sequences.

Our sequence-based clustering approach utilizes this information as follows: Let $L(c_1, c_2)$ be a sequence of labels induced by merging two clusters c_1 and c_2. If $P(\mathrm{L}(c_1, c_2))$ is the probability that sequence $L(c_1, c_2)$ is accepted by the automaton, and let $\cos(c_1, c_2)$ be the cosine distance between c_1 and c_2. We can measure the similarity between c_1 and c_2 as:

$$SIM(c_1, c_2) = \cos(c_1, c_2) \times P(\mathrm{L}(c_1, c_2)) \quad (2)$$

Let r be the number of clusters remaining. Then there are $\frac{r(r-1)}{2}$ pairs of clusters. For each pair of clusters c_1,c_2 we generate the resulting sequence of labels that would result if c_1 and c_2 were merged. We then input each label sequence to our trained FSA to obtain the probability that it is generated by the automaton. At each iteration, the algorithm proceeds by merging the most similar pair according to this metric. Fig. 2 illustrates this process in more detail. To terminate the clustering process, we adopt either the **"correct k"** or **"best k"** halting criteria described earlier.

Iterative-TFIDF Clustering

In algorithms described so far, term weighting was not applied. We propose an iterative TFIDF-like approach as follows. We define a "document" to be the set of sentences which discuss a given event and then weight terms according to their frequency in the document compared to the entire corpus. Of course, these "documents" are precisely what the clustering algorithm is trying to discover. We therefore initialize the term weights uniformly, and iteratively update the weights based on the current clusters. After each iteration we re-calculate the term weights based on the previous iterations clustering solution. The algorithm halts when the labels start to converge.

To explain our iterative TFIDF weighting method, consider an article with six sentences $s_1, \ldots, s_6$, with $\{s_1, s_2, s_3\}$ discussing one event, and $\{s_4, s_5, s_6\}$ discussing a second event. If we knew these correct clusters, then it would be straightforward to define TFIDF weights for each term in each cluster, and then assign a sentence's term weight to be zero if the sentence doesn't contain the term, or the term weight for the sentence's cluster if it does. We do not know the correct cluster, but our results below indicate that using just uniform weights yields reasonably accurate clusters. If the clustering algorithm happens make relatively minor mistakes (i.e, swapping a few sentences, or splitting an event into multiple clusters), then we can expect that the computed cluster's TFIDF weights will be a reasonable approximation to the weights computed from the true clusters. For example, if the term "bomb" occurs in one of the two true clusters, then its true IDF value is $\log(2/1) = 0.69$. If one of the clusters is incorrectly split, then the computed IDF value will be either $\log(3/1) = 1.1$ or $\log(3/2) = 0.41$.

We iteratively define the TFIDF weight for term t in sentence s as:

$$W_m(\mathrm{t}, \mathrm{s}) = \alpha(W(\mathrm{t}, \mathrm{c(s)})) + (1 - \alpha)W_{m-1}(\mathrm{t}, \mathrm{s}) \quad (3)$$

where $c(s)$ is the cluster to which sentence s belongs,

$$W(\mathrm{t}, \mathrm{c}) = tf(\mathrm{t}, \mathrm{c}) \times \ln\left(\frac{\mathrm{N}}{df(\mathrm{t})}\right) \quad (4)$$

is the TFIDF weight for a term t in a cluster c ($tf(\mathrm{t}, \mathrm{c})$ is the term frequency of term t in a cluster c, N is the number of

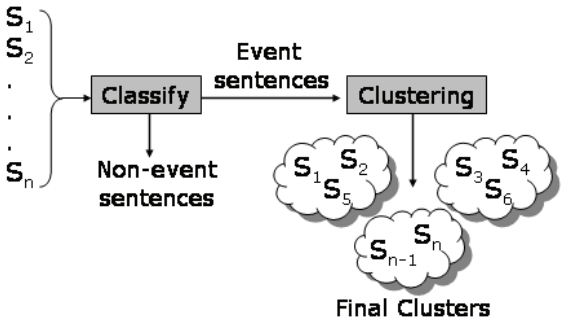


Figure 4: System architecture with an additional preprocessing phase.

clusters at this iteration and $df(\mathrm{t})$ is the number of distinct clusters containing term t). We use simulated annealing to ensure that $W_k(\mathrm{t},\mathrm{s})$ converges. α controls the rate at which the weights change. Initially $\alpha = 1$. We then decrease α by a fixed amount β after each iteration. For our experiments, we choose $\beta = 0.4$.

Finally, we formulate the similarity between two sentences s_1 and s_2 in equation 5 where s_{1j} is the weight of term t_j in s_1 and s_{2j} is the weight of term t_j in s_2.

$$SIM(s_1, s_2) = \frac{\sum_{j=1}^{t} s_{1j} s_{2j}}{\sqrt{\sum_{j=1}^{t} s_{1j}^2 \sum_{j=1}^{t} s_{2j}^2}} \quad (5)$$

Filtering Non-event Sentences

In the previous section, we presented two approaches for grouping together sentences in a news article that refer to the same event. However, we have discovered that in any given news article, some sentences do not make reference to any event(s). To eliminate such sentences, we train a classifier to discriminate between sentences that describe one or more events, and those that do not.

As shown in Fig. 4, we use this classifier as a preprocessing phase to remove non-event sentences before initiating the clustering process. Note that if the classifier misclassifies an event sentence as a non-event sentence, we cannot recover (ie, cluster the sentence with others that discuss the same event). On the other hand, if the classifier misclassifies a non-event sentence as an event sentence, the clustering task is made more difficult but it is possible for the algorithm to recover by segregating these non-event sentences by putting them in different clusters from those containing "event" sentences. We therefore want a classifier that is biased to have high precision for the "non event" class. To do so, we compare the classification confidence to a constant δ. We tune δ to maximize the quality of the final clusters.

Each sentence in our document collection forms a training/test instance for our classifier. We represent each sentence as a bag of stemmed words, with stopwords discarded. This representation works well where the target of classification is the topic of the document. Since we wish to classify each sentence as one which describes an event or as one which does not, we add additional features. Typically, sentences that refer to an event are longer and contain more proper nouns (for example, locations, person names, days etc.). To capture such characteristics, we add the following features: sentence length, sentence position in the document, number of capitalized terms, number of stopwords, number of days, number of city/town names (using a gazetteer) and the number of numeric tokens in the sentence.

Evaluation

In our experiments, we use a collection of 219 news stories from 46 different sources describing events related to the recent Iraq war. The articles are of varying size (average sentence length per document is 19). Most of the articles contain references to multiple events. The average number of events per document is 5.09 and the average number of events per sentence is 1.12. Excess HTML (image captions etc.) was removed, and sentence boundaries were identified using the *Lingua::EN::Sentence* perl module.

Our dataset was annotated by two volunteers. Within each article, events were uniquely identified by integers. Starting at the value 1, the annotators were asked to assign labels to each sentence representing the event(s) it describes. If a sentence did not refer to any event, it was assigned the label 0. Sentences may refer to multiple events. For example, consider the sentence *"These two bombings have claimed the lives of 23 Iraqi soldiers"*. This sentence would be annotated with two labels, one for each of the two bombings. Note that sentences from the same document that refer to the same event are assigned the same label.

To evaluate our clustering methods, we use the definition of precision and recall proposed by (Hess & Kushmerick 2003). We assign each pair of sentences into one of four categories: a, clustered together (and annotated as referring to the same event); b, not clustered together (but annotated as referring to the same event); c, incorrectly clustered together; d, correctly not clustered together. Precision and recall are thus found to be computed as $P = \frac{a}{a+c}$ and $R = \frac{a}{a+b}$, and $F1 = \frac{2PR}{P+R}$.

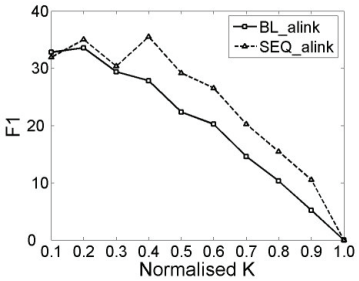
We also need to consider sentences annotated with multiple event labels. For each pair, where one or both of the sentences were annotated as referring to multiple events, we consider them as belonging in the same event cluster if the intersection between their labels is not empty. For example, we consider that a sentence pair with labels "1,2" and "1,3" respectively as belonging to the same cluster. Furthermore, for sentence pairs assigned with the label "0" (i.e non-event sentences) it would be incorrect to treat these as belonging to the same cluster, therefore we ignore such pairs when calculating accuracy.

Results

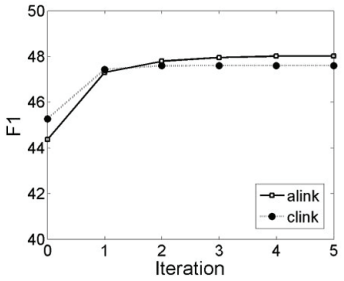
Table 1 shows the F1 achieved by each variation of our baseline and sequence-based algorithms when the "correct" and "best" k halting criteria are used. When comparing the performance of the sequence-based algorithm against our baseline, we see that for the majority of runs it out-preforms our baseline by 3% on average. Fig. 6(a) illustrates the F1

Sequence-Based Solution - Accuracy: 73.6%	Baseline Solution - Accuracy: 28.5%
Cluster 1	**Cluster 1**
S_Number: 1 - Two people were killed and at least and 8 were...	S_Number: 1 - Two people were killed and at least and 8 were...
S_Number: 2 - An American military spokesman said that one ..	**Cluster 2**
S_Number: 3 - Captain Sin Kerly said that the explosion which ..	S_Number: 2 - An American military spokesman said that one ..
S_Number: 4 - The explosion took place near al-Qanat hotel	S_Number: 3 - Captain Sin Kerly said that the explosion which ..
Cluster 2	S_Number: 7 - In al-Khaledeyah town to the west of Baghdad, ..
S_Number: 5 - In the same context, the general headquarters ...	S_Number: 9 - An American soldier was killed in a bomb attack..
Cluster 3	S_Number: 8 - The American forces did not confirm the incident.
S_Number: 6 - No human nor material losses were reported.	S_Number: 4 - The explosion took place near al-Qanat hotel ...
Cluster 4	**Cluster 3**
S_Number: 7 - In al-Khaledeyah town to the west of Baghdad, ...	S_Number: 5 - In the same context, the general headquarters ...
S_Number: 8 - The American forces did not confirm the incident.	**Cluster 4**
S_Number: 9 - An American soldier was killed in a bomb attack..	S_Number: 6 - No human nor material losses were reported.

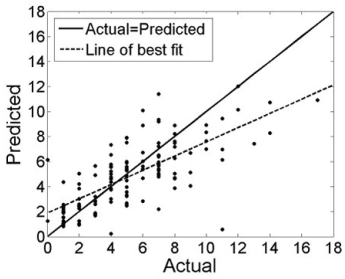
Figure 5: Sample clustering solutions generated by our sequence-based and baseline algorithms respectively



(a) F1 for the alink variation of both baseline and sequence-based algorithms for varying values of k.



(b) F1 at each iteration of our Iterative-TFIDF clustering algorithm.



(c) Actual Vs predicted values of k as predicted by an SVM regression learner.

Table 1: F1 achieved by the complete-link (clink) and average-link (alink) variations of our baseline (BL) and sequence-based (SEQ) clustering algorithms with (w/ c) and without the classification phase (w/o c). Figures towards the bottom shows F1 achieved when both algorithms are combined with our iterative-TFIDF (I_{tfidf}) method.

Algorithm	clink		alink	
	w/o c	w/ c	w/o c	w/ c
BL(correct k)	32.31	41.34	33.82	40.99
BL(best k)	44.59	**52.89**	45.78	**53.82**
SEQ(correct k)	35.77	47.58	37.93	47.72
SEQ(best k)	46.57	**56.16**	47.79	**56.85**
BL(correct k) + I_{tfidf}	32.33	39.12	33.61	39.65
BL(best k) + I_{tfidf}	48.62	51.83	48.81	51.77
SEQ(correct k)+I_{tfidf}	35.56	44.47	37.41	45.33
SEQ(best k)+I_{tfidf}	49.28	**53.88**	49.05	**54.20**

achieved by the average-link variation of our baseline and sequence-based approaches as a function of k, where k is normalized by document length. We see that even for small values of k, the sequence-based algorithm out-performs our baseline by a considerable margin. Fig. 5 depicts two sample clustering solutions generated by our sequence-based and baseline clustering algorithms respectively. We observe that our sequence-based algorithm actively merges closely positioned sentences in a document and does so in a manner that greatly improves accuracy.

In addition, we evaluated the performance observed by combining our baseline and sequence-based algorithms with our iterative-weighting method. To do this, we first generated an initial clustering solution using the baseline/sequence-based algorithm. Using this solution we iteratively updated the weights based on the previous iteration's generated clusters until the solution starts to converge. Table 1 shows the F1 achieved by combining these methods. The results show that when the correct k halting criterion is selected, we observe no significant improvement by combining the Iterative approach with our baseline and sequence algorithm. Notably however, when the "best" k criterion is chosen, we see improvements of about 4% in performance. Fig. 6(b) shows the level of F1 achieved at each iteration by our iterative-weighting algorithm when combined with our baseline for the "best" k halting criterion. We observe an overall increase in accuracy for both variations of the algorithm. Typically, this method converges after 3-6 iterations.

The Weka framework (Witten & Frank 2000) was used to carry out our classification experiments. We adopted Weka's SMO (Platt 1999) algorithm as our classifier. We randomly split our dataset into training and test sets using varying levels of training data for each run. SMO achieves 82%

accuracy on this task using only 10% of the training data. When evaluated with a 90/10 training/test split for example, our classifier achieves 89% accuracy. Table 1 shows the F1 achieved by all variations of our algorithms when we pre-classify each sentence using our trained classifier. On average, the performance of each run is improved by 8%. It most noted that this margin of increase is somewhat dependant on the error rate of the classifier. In experiments where δ is set to 0.5 (binary classification), the classifier tends to have a higher error rate, as a result we observe a smaller increase (5%) in performance.

Conclusions and Future Work

In this paper, we describe and evaluate methods for annotating each sentence in an article with a set of identifiers specifying which event(s) the sentence mentions. We introduce two novel approaches to event annotation and through evaluation with our baseline system, we demonstrate a modest increase in accuracy for both approaches. Typical news articles contain varying levels of sentences that do not refer to an event. We adopt a Machine Learning-based approach which classifies each sentence as either an event or non-event sentence. We trained a classifier that has the ability to achieve over 80% accuracy with limited training data. However, it must be noted that our classifier was largely trained on data used in the clustering phase. To avoid this issue, we are currently expanding our dataset and exploring alternative ways of incorporating this classification step into the clustering process.

We are extending our work in several directions. Firstly, our experiments rely on two methods to set the desired number of clusters k: "correct" sets k to be the actual number of events, and "best" tunes k so as to maximize the quality of the resulting clusters. Both methods involve manual supervision. We are currently exploring methods for automatically predicting the number of events k in a document. In a preliminary experiment, we trained an SVM regression learner to predict the log of the number of events in a document. We used the following features to encode each document: document length, avg sentence length ,number of distinct terms, distinct capitalized terms, stopwords, locations, days, numeric tokens and abbreviated terms. Fig. 6(c) shows the actual and predicted values of k for a disjoint set of test documents. This data has a correlation of 0.72, suggesting that it may be feasible to automatically predict k. As future work, we will concentrate on improving the accuracy of predicted values by adding more features and on incorporating this technique into the clustering phase.

Preliminary results presented in this paper suggest that there is inherent structure in the order events are described in news articles. They also indicate that when this structural information is incorporated into the clustering process, more accurate solutions are observed. However, the FSA used during this process was trained on documents used for clustering. We are currently focusing on training our FSA on a disjoint set of training documents and evaluating it's influence during the clustering of a separate test set.

Finally, a single sentence often contains references to multiple events. Our algorithms assume that each sentence describes just one event. Future work will focus on developing methods to automatically recognize such sentences and techniques to incorporate them into the clustering process.

Acknowledgments. This research was supported by the Irish Research Council for Science, Engineering & Technology (IRCSET) and IBM under grant RS/2004/IBM/1.

References

Hatzivassiloglou, V.; Klavans, J.; Holcombe, M.; Barzilay, R.; Kan, M.-Y.; and McKeown, R. 2001. Simfinder; a flexible clustering tool for summarisation. In *NAACL Workshop on Automatic Summarisation*, 41–49.

Hess, A., and Kushmerick, N. 2003. Learning to attach semantic metadata to web services. In *Proc. Int. Semantic Web Conf.*

Li, Z.; Wang, B.; Li, M.; and Ma, W.-Y. 2005. A probabilistic model for retrospective news event detection. In *Proceedings of the 28th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, 106–113.

Manning, C. D., and Schtze, H. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

McKeown, K., and Radev, D. 1995. Generating summaries of multiple news articles. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 74–82.

Nomoto, T., and Matsumoto, Y. 2001. A new approach to unsupervised text summarization. In *SIGIR 01 Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 26–34.

Platt, J. C. 1999. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning* 185–208.

Porter, M. F. 1997. An algorithm for suffix stripping. *Readings in information retrieval* 313–316.

Radev, D.; Jing, H.; and Budzikowska, M. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation and user studies. In *Proceedings of ANLP/NAACL 2000 Workshop on Automatic Summarization*.

Salton, G.; Singhal, A.; Mitra, M.; and Buckley, C. 1997. Automatic text structuring and summarization. *Information Processing and Management* 33:193–207.

Thollard, F.; Dupont, P.; and de la Higuera, C. 2000. Probabilistic dfa inference using kullback-leibler divergence and minimality. In *Proceedings of the Seventeenth International Conference on Machine Learning*.

Witten, I. H., and Frank, E. 2000. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers Inc.

Zha, H. 2002. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 113–120.