Event-Based Extractive Summarization

Elena Filatova
Department of Computer Science
Columbia University
New York, NY 10027, USA
filatova@cs.columbia.edu

Vasileios Hatzivassiloglou
Center for Computational Learning Systems
Columbia University
New York, NY 10027, USA
vh@cs.columbia.edu

Abstract

Most approaches to extractive summarization define a set of features upon which selection of sentences is based, using algorithms independent of the features themselves. We propose a new set of features based on low-level, atomic events that describe relationships between important actors in a document or set of documents. We investigate the effect this new feature has on extractive summarization, compared with a baseline feature set consisting of the words in the input documents, and with state-of-the-art summarization systems. Our experimental results indicate that not only the event-based features offer an improvement in summary quality over words as features, but that this effect is more pronounced for more sophisticated summarization methods that avoid redundancy in the output.

1 Introduction

The main goal of extractive summarization can be concisely formulated as extracting from the input pieces of text which contain the information about the most important concepts mentioned in the input text or texts. This definition conceals a lot of important issues that should be taken into consideration in the process of summary construction. First, it is necessary to identify the important concepts which should be described in the summary. When those important concepts are identified then the process of summarization can be presented as:

1. Break the input text into textual units (sentences, paragraphs, etc.).
2. See what concepts each textual unit covers.
3. Choose a particular textual unit for the output according to the concepts present in all textual units.
4. Continue choosing textual units until reaching the desired length of the summary.

Some current summarization systems add a clustering step, substituting the analysis of all the textual units by the analysis of representative units from each cluster. Clustering is helpful for avoiding repetitions in the summary.

In this paper we propose a new representation for concepts and correspondingly a new feature on which summarization can be based. We adapt the algorithm we proposed earlier (Filatova and Hatzivassiloglou, 2003) for assigning to each sentence a list of low-level, *atomic events*. These events capture information about important named entities for the input text or texts, and the relationships between these named entities. We also discuss a general model which treats summarization as a three-component problem, involving the identification of the textual units into which the input text should be broken and which are later used as the constituent parts of the final summary, the textual features which are associated with the important concepts described in the input text, and the appropriate algorithm for selecting the textual units to be included into the summary.

We focus on the latter two of those steps and explore interdependencies between the choice of features (step 2) and selection algorithm (step 3). We experimentally test our hypothesis that event-based features are helpful for summarization by comparing the performance of three sentence selection algorithms when we use such features versus the case where we use another, widely used set of textual features: the words in the input texts, weighted by their *tf*idf* scores. The results establish that for the majority of document sets in our test collection, events outperform *tf*idf* for all algorithms considered. Furthermore, we show that this benefit is more pronounced when the selection algorithm includes steps to address potential repetition of information in the output summary.

2 General Summarization Model

Many summarization systems (e.g., (Teufel and Moens, 1997; McKeown et al., 1999; Lin and

	c_1	c_2	c_3	c_4	c_5
t_1	1	1	0	1	1
t_2	1	0	0	1	0
t_3	0	1	0	0	1
t_4	1	0	1	1	1

Table 1: Matrix for Summarization Model

Hovy, 2000)) include two levels of analysis: the sentence level, where every textual unit is scored according to the concepts or features it covers, and the text level, where, before being added to the final output, textual units are compared to each other on the basis of those features.

In Section 1 we presented a four-step pipeline for extractive summarization; existing summarization systems largely follow this pipeline, although they introduce different approaches for every step in it. We suggest a model that describes the extractive summarization task in general terms. Consider the matrix in Table 1. Rows of this matrix represent all textual units into which the input text is divided. Columns represent the concepts discovered for the input text. Every concept is either absent or present in a given textual unit. Each concept c_i has also an associated weight w_i indicating the importance of this concept. These weights can be used for scoring the textual units.

Thus, the input text and the important information in it is mapped onto an $m \times n$ matrix. Using the above matrix it is possible to formulate the extractive summarization problem as extracting the minimal amount of textual units which cover all the concepts that are interesting or important. To account for the cost of long summaries, we can constrain the total length of the summary, or balance it against the total weight of covered concepts.

The presented model can be also used for comparing summaries consisting of different textual units. For example, a summary consisting only of textual unit t_1 renders the same information as the summary consisting of textual units t_2 and t_3. Both these summaries cover the same set of concepts, namely c_1, c_2 and c_3. We explore properties of this model in more detail in (Filatova and Hatzivassiloglou, 2004).

3 Associating Concepts with Features

Before extracting a summary, it is necessary to define what concepts in the input text are important and should be covered by the output text. There is no exact definition or even agreement between different approaches on what an important concept is. In order to use the model of Section 2 one has to approximate the notion of "concept" with some textual features.

Current summarization approaches use text features which give high scores to the textual units that contain important information, and low scores to those textual units which are not highly likely to contain information worth to be included in the final output. There exist approaches that deal mainly with lexical features, like *tf*idf* weighing of words in the input text(s), words used in the titles and section headings (Luhn, 1959; Edmundson, 1968), or the presence or absence of certain cue phrases like *significant*, *important*, and *in conclusion* (Kupiec et al., 1995; Teufel and Moens, 1997). Other systems exploit the co-occurrence of particular concepts (Barzilay and Elhadad, 1997; Lin and Hovy, 2000) or syntactic constraints between concepts (McKeown et al., 1999). Concepts do not have to be directly observable as text snippets—they can represent abstract properties that particular text units may or may not satisfy, for example, status as a first sentence in a paragraph or generally position in the source text (Baxedale, 1958; Lin and Hovy, 1997). Some summarization systems assume that the importance of a sentence is derivable from a rhetorical representation of the source text (Marcu, 1997).

The matrix representation of the previous section offers a way to formalize the sharing of information between textual units at the individual feature level. Thus, this representation is most useful for content-related concepts that should not be repeated in the summary. The representation can however handle independent features such as sentence position by encoding them separately for each textual unit.

4 Atomic Events

Atomic events link major constituent parts of the actions described in a text or collection of texts through the verbs or action nouns labeling the event itself. The idea behind this technique is that the major constituent parts of events (participants, locations, times) are usually realized in text as named entities. The more important the constituent part, the more often the corresponding named entity is mentioned.

Not all the constituent parts of events need to be represented by named entities. For exam-

ple, in an airline crash it is important to report information about the passengers and the crew. These are not marked by named entities but are highly likely to be among the most frequently used nouns. Thus, we add the top ten most frequent nouns to the list of named entities.

We use the algorithm for atomic event extraction proposed in (Filatova and Hatzivassiloglou, 2003). It involves the following steps:

1. Analyze each input sentence[1] one at a time; ignore sentences that do not contain at least two named entities or frequent nouns.
2. Extract all the possible pairs of named entities/frequent nouns in the sentence, preserving their order and all the words in between. We call such pairs of named entities *relations*, and the words in-between the named entities in a relation *connectors*.
3. For each relation, count how many times this relation is used in the input text(s).
4. Keep only connectors that are content verbs or action nouns, according to WordNet's (Fellbaum, 1998) noun hierarchy. For each connector calculate how many times it is used for the extracted relation.

After calculating the scores for all relations and all connectors within each relation, we calculate their normalized scores The normalized relation score is the ratio of the count for the current relation (how many times we see the relation within a sentence in the input) over the overall count of all relations. The normalized connector score is the ratio of the count for the current connector (how many times we see this connector for the current relation) over the overall count for all connectors for this relation.

Thus, out of the above procedural definition, an atomic event is a triplet of two named entities (or frequent nouns) connected by a verb or an action-denoting noun. To get a score for the atomic event we multiply the normalized score for the relation by the normalized score for the connector. The score indicates how important the triplet is overall.

In the above approach to event detection we do not address co-reference, neither we merge together the triplets which describe the same event using paraphrases, inflected forms and syntactic variants (e.g., active/passive voice). Our method uses relatively simple extraction techniques and shallow statistics, but it is fully automatic and can serve as a first approximation of the events in the input text(s).

Our approach to defining events is not the only one proposed—this is a subject with substantial work in linguistics, information retrieval, and information extraction. In linguistics, events are often defined at a fine-grained level as a matrix verb or a single action noun like "war" (Pustejovsky, 2000). In contrast, recent work in information retrieval within the TDT framework has taken event to mean essentially "narrowly defined topic for search" (Allan et al., 1998). Finally, for the information extraction community an event represents a template of relationships between participants, times, and places (Marsh and Perzanowski, 1997). It may be possible to use these alternative models of events as a source of content features.

We earlier established empirically (Filatova and Hatzivassiloglou, 2003) that this technique for atomic event extraction is useful for delineating the major participants and their relationships from a set of topically related input texts. For example, from a collection of documents about an airplane crash the algorithm assigns the highest score to atomic events that link together the name of the airline, the source and destination airports and the day when the crash happened through the verb *crashed* or its synonyms. It is thus plausible to explore the usefulness of these event triplets as the concepts used in the model of Section 2.

5 Textual Unit Selection

We have formulated the problem of extractive summarization in terms of the matrix model, stating that mapping concepts present in the input text onto the textual units out of which the output is constructed can be accomplished by extracting the minimal amount of textual units which either cover most of the important concepts. Every time we add a new textual unit to the output it is possible to judge what concepts in it are already covered in the final summary. This observation can be used to avoid redundancy: before adding a candidate textual unit to the output summary, we check whether it contains enough new important concepts.

We describe in this section several algorithms for selecting appropriate textual units for the output summary. These algorithms differ on whether they take advantage of the redundancy reduction property of our model, and on

[1] We earlier showed empirically (Filatova and Hatzivassiloglou, 2003) that a description of a single event is usually bound within one sentence.

whether they prioritize important concepts individually or collectively. They share, however, a common property: all of them operate independently of the features chosen to represent important concepts, and thus can be used with both our event-based features and other feature sets. The comparison of the results allows us to empirically determine whether event-based features can help in summarization.

5.1 Static Greedy Algorithm

Our first text unit selection algorithm does not support any mechanism for avoiding redundant information in the summary. Instead, it rates each textual unit independently. Textual units are included in the summary if and only if they cover lots of concepts. More specifically,

1. For every textual unit, calculate the weight of this textual unit as the sum of the weights of all the concepts covered by this textual unit.
2. Choose the textual unit with the maximum weight and add it to the final output.
3. Continue extracting other textual units in order of total weight till we get the summary of the desired length.

5.2 Avoiding Redundancy in the Summary

Two popular techniques for avoiding redundancy in summarization are Maximal Marginal Relevance (MMR) (Goldstein et al., 2000) and clustering (McKeown et al., 1999). In MMR the determination of redundancy is based mainly on the textual overlap between the sentence that is about to be added to the output and the sentences that are already in the output. Clustering offers an alternative: before starting the selection process, the summarization system clusters the input textual units. This step allows analyzing one representative unit from each cluster instead of all textual units.

We take advantage of the model matrix of Section 2 to explore another way to avoid redundancy. Rather than making decisions for each textual unit independently, as in our Static Greedy Algorithm, we *globally* select the subset of textual units that cover the most concepts (i.e., information) present in the input. Then our task becomes very similar to a classic theory problem, Maximum Coverage.

Given C, a finite set of weighted elements, a collection T of subsets of C, and a parameter k, the maximum coverage problem is to find k members of T such that the total weight of the elements covered (i.e., belonging to the k members of the solution) is maximized. This problem is NP-hard, as it can be reduced to the well-known *set cover* problem (Hochbaum, 1997). Thus, we know only approximation algorithms solving this problem in polynomial time.

Hochbaum (1997) reports that a greedy algorithm is the best possible polynomial approximation algorithm for this problem. This algorithm iteratively adds to the solution S the set $t_i \in T$ that locally maximizes the increase in the total weight of elements covered by $S \cup t_i$. The algorithm gives a solution with weight at least $1/(1-e)$ of the optimal solution's total weight.

5.3 Adaptive Greedy Algorithm

The greedy algorithm for the maximum coverage problem is not directly applicable to summarization, because the formulation of maximum coverage assumes that any combination of k sets t_i (i.e., k sentences) is equally good as long as they cover the same total weight of concepts. A more realistic limitation for the summarization task is to aim for a fixed total length of the summary, rather than a fixed total number of sentences; this approach has been adopted in several evaluation efforts, including the Document Understanding Conferences (DUC). We consequently modify the greedy algorithm for the maximum coverage problem to obtain the following *adaptive greedy algorithm* for summarization:

1. For each textual unit calculate its weight as the sum of weights of all concepts it covers.
2. Choose the textual unit with the maximum weight and add it to the output. Add the concepts covered by this textual unit to the list of concepts covered in the final output.
3. Recalculate the weights of the textual units: subtract from each unit's weight the weight of all concepts in it that are already covered in the output.
4. Continue extracting text units in order of their total weight (going back to step 2) until the summary is of the desired length.

5.4 Modified Adaptive Greedy Algorithm

The adaptive greedy algorithm described above prioritizes sentences according to the total weight of concepts they cover. While this is a reasonable approach, an alternative is to give increased priority to concepts that are individually important, so that sentences mentioning

them have a chance of being included in the output even if they don't contain other important concepts. We have developed the following variation of our adaptive greedy algorithm, termed the *modified greedy algorithm*:

1. For every textual unit calculate its weight as the sum of weights of all concepts it covers.
2. Consider only those textual units that contain the concept with the highest weight that has not yet been covered. Out of these, choose the one with highest total weight and add it to the final output. Add the concepts which are covered by this textual unit to the list of concepts covered in the final output.
3. Recalculate the weights of the textual units: subtract from each unit's weight the weight of all concepts in it that are already covered in the output.
4. Continue extracting textual units, going back to step 2 each time, until we get a summary of the desired length.

The modified greedy algorithm has the same mechanism for avoiding redundancy as the adaptive greedy one, while according a somewhat different priority to individual sentences (weight of most important concepts versus just total weight).

6 Experiments

We chose as our input data the document sets used in the evaluation of multidocument summarization during the first Document Understanding Conference (DUC), organized by NIST (Harman and Marcu, 2001). This collection contains 30 test document sets, each with approximately 10 news stories on different events; document sets vary significantly in their internal coherence. For each document set three human-constructed summaries are provided for each of the target lengths of 50, 100, 200, and 400 words. We selected DUC 2001 because ideal summaries are available for multiple lengths.

Concepts and Textual Units Our textual units are sentences, while the features representing concepts are either atomic events, as described in Section 4, or a fairly basic and widely used set of lexical features, namely the list of words present in each input text. The algorithm for extracting event triplets assigns a weight to each such triplet, while for words we used as weights their *tf*idf* values, taking *idf* values from `http://elib.cs.berkeley.edu/docfreq/`.

	50	100	200	400
events better	53.3%	63.3%	80.0%	80.0%
tf*idf better	23.3%	26.7%	20.0%	20.0%
equal	23.3%	10.0%	0.0%	0.0%

Table 2: Static greedy algorithm, events versus *tf*idf*.

Evaluation Metric Given the difficulties in coming up with a universally accepted evaluation measure for summarization, and the fact that obtaining judgments by humans is time-consuming and labor-intensive, we adopted an automated process for comparing system-produced summaries to "ideal" summaries written by humans. The method, ROUGE (Lin and Hovy, 2003), is based on n-gram overlap between the system-produced and ideal summaries. As such, it is a recall-based measure, and it requires that the length of the summaries be controlled to allow meaningful comparisons.

ROUGE can be readily applied to compare the performance of different systems on the same set of documents, assuming that ideal summaries are available for those documents. At the same time, ROUGE evaluation has not yet been tested extensively, and ROUGE scores are difficult to interpret as they are not absolute and not comparable across source document sets.

In our comparison, we used as reference summaries those created by NIST assessors for the DUC task of generic summarization. The human annotators may not have created the same models if asked for summaries describing the major events in the input texts instead of generic summaries.

Summary Length For a given set of features and selection algorithm we get a sorted list of sentences extracted according to that particular algorithm. Then, for each DUC document set we create four summaries of length 50, 100, 200, and 400. In all the suggested methods a whole sentence is added at every step. We extracted exactly 50, 100, 200, and 400 words out of the top sentences (truncating the last sentence if necessary).

6.1 Results: Static Greedy Algorithm

In our first experiment we use the static greedy algorithm to create summaries of various lengths. Table 2 shows in how many cases out

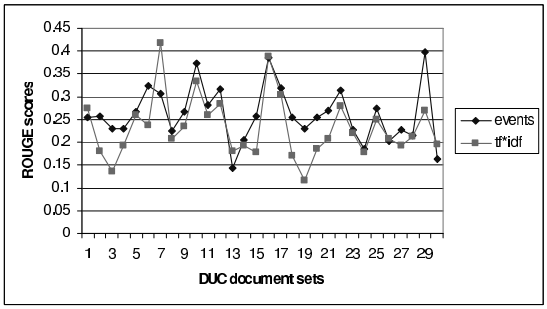

Figure 1: ROUGE scores for 400-word summaries for static greedy algorithm, events versus *tf*idf*.

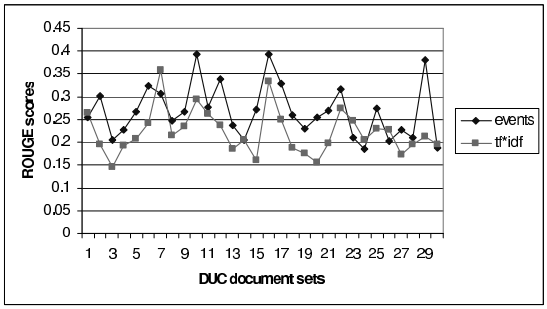

Figure 2: ROUGE scores for 400-word summaries for adaptive greedy algorithm, events versus *tf*idf*.

	50	100	200	400
events better	53.3%	66.7%	86.7%	80.0%
tf*idf better	23.3%	20.0%	13.3%	20.0%
equal	23.3%	13.3%	0.0%	0.0%

Table 3: Adaptive greedy algorithm, events versus *tf*idf*.

	50	100	200	400
static better	0.0%	3.3%	20.0%	23.3%
adaptive better	10.0%	16.7%	26.6%	40.0%
equal	90.0%	80.0%	53.3%	36.7%

Table 4: Adaptive greedy algorithm versus static greedy algorithm, using events as features.

of the 30 document sets the summary created according to atomic events receives a higher or lower ROUGE score than the summary created according to *tf*idf* features (rows "events better" and "tf*idf better" respectively). Row *equal* indicates how many of the 30 cases both systems produce results with the same ROUGE score. We chose to report the number of times each system is better rather than the average ROUGE score in each case because ROUGE scores depend on each particular document set.

It is clear from Table 2 that the summaries created using atomic events are better in the majority of cases than the summaries created using *tf*idf*. Figure 1 shows ROUGE scores for 400-word summaries. Although in most cases the performance of the event-based summarizer is higher than the performance based on *tf*idf* scores, for some document sets *tf*idf* gives the better scores. This phenomenon can be explained through an additional analysis of document sets according to their internal coherence. Atomic event extraction works best for a collection of documents with well-defined constituent parts of events and where documents are clustered around one specific major event. For such document sets atomic events are good features for basing the summary on. In contrast, some DUC 2001 document sets describe a succession of multiple events linked in time or of different events of the same type (e.g., Clarence Thomas' ascendancy to the Supreme Court, document set 7 in Figure 1, or the history of airplane crashes, document set 30 in Figure 1). In such cases, a lot of different participants are mentioned with only few common elements (e.g., Clarence Thomas himself). Thus, most of the atomic events have similar low weights and it is difficult to identify those atomic events that can point out the most important textual units.

6.2 Results: Adaptive Greedy Algorithm

For the second experiment we used the adaptive greedy algorithm, which accounts for information overlap across sentences in the summary. As in the case of the simpler static greedy algorithm, we observe that events lead to a better performance in most document sets than *tf*idf* (Table 3). Table 3 is in fact similar to Table 2, with slightly increased numbers of document sets for which events receive higher ROUGE scores for the 100 and 200-word summaries. It is interesting to see that the difference between the ROUGE scores for the summarizers based on atomic events and *tf*idf* features becomes more distinct when the adaptive greedy algorithm is used; Figure 2 shows this for 400-word summaries.

As Table 4 shows, the usage of the adaptive greedy algorithm improves the performance of a summarizer based on atomic events in comparison to the static greedy algorithm. In contrast, the reverse is true when *tf*idf* is used (Table 5). Figure 3 shows the change in ROUGE scores

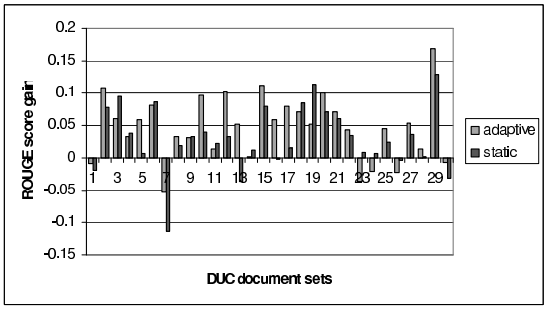

Figure 3: Gain in ROUGE scores (400-word summaries) when using events instead of *tf*idf* for the static and adaptive greedy algorithms.

	50	100	200	400
static better	3.3%	26.7%	43.3%	50.0%
adaptive better	3.3%	13.3%	30.0%	50.0%
equal	93.3%	60.0%	26.7%	0.0%

Table 5: Adaptive greedy algorithm versus static greedy algorithm, using *tf*idf* as features.

that the introduction of the adaptive algorithm offers for 400-word summaries. This indicates that *tf*idf* is not compatible with our information redundancy component; a likely explanation is that words are correlated, and the presence of an important word makes other words in the same sentence also potentially important, a fact not captured by the *tf*idf* feature. Events, on the other hand, exhibit less of a dependence on each other, since each triplet captures a specific interaction between two entities.

6.3 Results: Modified Greedy Algorithm

In the case of the modified adaptive greedy algorithm we see improvement in performance in comparison with the summarizers using the static greedy algorithm for both events and *tf*idf* (Tables 6 and 7). In other words, the prioritization of individual important concepts addresses the correlation between words and allows the summarizer to benefit from redundancy reduction even when using *tf*idf* as the features. The modified adaptive algorithm offers a slight improvement in ROUGE scores over the unmodified adaptive algorithm. Also, as Table 8 makes clear, events remain the better feature choice over *tf*idf*.

6.4 Results: Comparison with DUC systems

For our final experiment we used the 30 test document sets provided for DUC 2003 competition, for which the summaries produced by participating summarization systems were also released. In DUC 2003 the task was to create summaries only of length 100.

	50	100	200	400
static better	43.3%	43.3%	36.7%	43.3%
modified better	43.3%	56.7%	63.3%	56.7%
equal	13.3%	0.0%	0.0%	0.0%

Table 6: Modified adaptive greedy algorithm versus static greedy algorithm, using events as features.

	50	100	200	400
static better	6.7%	26.7%	36.7%	26.7%
modified better	30.0%	40.0%	56.7%	73.3%
equal	63.3%	33.3%	6.7%	0.0%

Table 7: Modified adaptive greedy algorithm versus static greedy algorithm, using *tf*idf* as features.

	50	100	200	400
events better	56.7%	70.0%	80.0%	66.6%
tf*idf better	33.3%	30.0%	20.0%	33.3%
equal	10.0%	0.0%	0.0%	0.0%

Table 8: Modified adaptive greedy algorithm, events versus *tf*idf*.

We calculated ROUGE scores for the released summaries created by DUC participants and compared them to the scores of our system with atomic events as features and adaptive greedy algorithm as the filtering method. In 14 out of 30 cases our system outperforms the median of the scores of all the 15 participating systems over that specific document set. We view this comparison as quite encouraging, as our system does not employ any of the additional features (such as sentence position or time information) used by the best DUC summarization systems, nor was it adapted to the DUC domain. Again, the suitability (and relative performance) of the event-based summarizer varies according to the type of documents being summarized, indicating that using our approach for a subset of document sets is more appropriate. For example, our system scored below all the other systems for the document set about a meteor shower, which included a lot of background information and no well-defined constituents of events. On the contrary, our system performed better than any DUC system for the document set describing an abortion-related murder, where it was clear who was killed and where and when it happened.

7 Conclusion

We have introduced atomic events as a feature that can be automatically extracted from text and used for summarization, and described algorithms that utilize this feature to select sentences for the summary while minimizing the overlap of information in the output. Our experimental results indicate that events are indeed an effective feature, at least in comparison with words in the input texts that form the basis of many of current summarizers' feature sets. With all three of our summarization algorithms, we achieved a gain in performance when using events. This gain was actually more pronounced with the more sophisticated sentence selection methods, establishing that events also exhibit less interdependence than features based directly on words. The advantage was also larger in longer summaries.

Our approach to defining and extracting events can be improved in many ways. We are currently looking at ways of matching connectors that are similar in meaning, representing paraphrases of the same event, and methods for detecting and prioritizing special event components such as time and location phrases. We are also considering merging information across many related atomic events to a more structured representation for each event, and allowing for partial matches between such structures and input sentences.

8 Acknowledgements

We wish to thank Rocco Servedio and Mihalis Yannakakis for valuable discussions of theoretical foundations of the set cover problem. We also thank Kathy McKeown and Noemie Elhadad for comments on an earlier version. This work was supported by ARDA under Advanced Question Answering for Intelligence (AQUAINT) project MDA908-02-C-0008. Any opinions, findings, or recommendations are those of the authors.

References

James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. 1998. Topic detection and tracking plot study: Final report. In *Proceedings of the DARPA Broadcast News Trascription Workshop*, April.

Regina Barzilay and Michael Elhadad. 1997. Using lexical chains for text summarization. In *Proceedings of the ACL/EACL 1997 Workshop on Intelligent Scalable Text Summarizaion*, Madrid, Spain, July.

P. B. Baxedale. 1958. Machine-made index for technical literature—An experiment. *IBM Journal of Research and Development*, **2**:354–361.

H. P. Edmundson. 1968. New methods in automatic extracting. *Journal of the Association for Computing Machinary*, **23**(1):264–285, April.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Elena Filatova and Vasileios Hatzivassiloglou. 2003. Domain-independent detection, extraction, and labeling of atomic events. In *Proceedings of RANLP*, pages 145–152, Borovetz, Bulgaria, September.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. A formal model for information selection in multi-sentence text extraction. In *Proceedings of COLING*, Geneva, Switzerland, August.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Jamie Callan. 2000. Creating and evaluating multi-document sentence extract summaries. In *Proceedings of the 9th CIKM Conference*, pages 165–172.

Donna Harman and Daniel Marcu, editors. 2001. *Proceedings of the Document Understanding Conference (DUC)*. NIST, New Orleans, USA, September.

Dorit S. Hochbaum. 1997. Approximating covering and packing problems: Set cover, vertex cover, independent set, and related problems. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 94–143. PWS Publishing Company, Boston, MA.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th ACM SIGIR Conference*, pages 68–73, Seattle, Washington, May.

Chin-Yew Lin and Eduard Hovy. 1997. Identifying topic by position. In *Proceedings of the 5th ANLP Conference*, Washington, DC.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the COLING Conference*, Saarbrücken, Germany, July.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT-NAACL*, Edmonton, Canada, May.

H. P. Luhn. 1959. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, **2**(2):159–165, April.

Daniel Marcu. 1997. From discourse structures to text summaries. In *Proceedings of the ACL/EACL 1997 Workshop on Intelligent Scalable Text Summarizaion*, pages 82–88, Madrid, Spain, July.

E. Marsh and D. Perzanowski. 1997. MUC-7 evaluation of IE technology: Overview of results. In *Proceedings of MUC-7*.

Kathleen R. McKeown, Judith L. Klavans, Vasileios Hatzivassiloglou, Regina Barzilay, and Eleazar Eskin. 1999. Towards multidocument summarization by reformulation: Progress and prospects. In *Proceedings of AAAI*.

James Pustejovsky, 2000. *Events and the Semantics of Opposition*, pages 445–482. CSLI Publications.

Simone Teufel and Marc Moens. 1997. Sentence extraction as a classification task. In *Proceedings of the ACL/EACL 1997 Workshop on Intelligent Scalable Text Summarization*, pages 58–65, Madrid, Spain, July.