Towards Multidocument Summarization by Reformulation: Progress and Prospects

Kathleen R. McKeown, **Judith L. Klavans**, **Vasileios Hatzivassiloglou**, **Regina Barzilay** and **Eleazar Eskin**

Department of Computer Science
Columbia University
1214 Amsterdam Avenue
New York, NY 10027

{kathy, klavans, vh, regina, eeskin}@cs.columbia.edu

Abstract

By synthesizing information common to retrieved documents, multi-document summarization can help users of information retrieval systems to find relevant documents with a minimal amount of reading. We are developing a multi-document summarization system to automatically generate a concise summary by identifying and synthesizing similarities across a set of related documents. Our approach is unique in its integration of machine learning and statistical techniques to identify similar paragraphs, intersection of similar phrases within paragraphs, and language generation to reformulate the wording of the summary. Our evaluation of system components shows that learning over multiple extracted linguistic features is more effective than information retrieval approaches at identifying similar text units for summarization and that it is possible to generate a fluent summary that conveys similarities among documents even when full semantic interpretations of the input text are not available.

Introduction

Currently, most approaches to single document summarization involve extracting key sentences to form the summary (e.g., [Paice 1990; Kupiec *et al.* 1995; Marcu 1998]). Yet, given the multitude of sources that describe the same event in a similar manner (e.g., on-line news sources), it would be helpful to the end-user to have a summary of multiple related documents. Multiple document summarization could be useful, for example, in the context of large information retrieval systems to help determine which documents are relevant. Such summaries can cut down on the amount of reading by synthesizing information common among all retrieved documents and by explicitly highlighting distinctions. In contrast, with single document summarization, users would have to read numerous individual summaries, one for each of the top retrieved documents and infer similarities.

While sentence extraction may be adequate for single document summarization, it will not work effectively for multiple document summarization. Any individual document does not contain explicit comparisons with all other documents which can be extracted; alternatively, if all sentences containing similar information are extracted (Mani and Bloedorn, 1997; Yang *et al.* 1998), this would make for lengthy and repetitive reading.

Copyright © 1999, American Association for Artificial Intelligence (http://www.aaai.org). All rights reserved.

A federal office building was devastated by a car bomb in Oklahoma City on April 19th 1995. Around 250 people are still unaccounted for. More than 80 victims were killed in the explosion. The Oklahoma blast was the biggest act of suspected terror in U.S. history, overtaking the 1993 bombing of the World Trade Center in New York which killed six and injured 1,000 others.

President Clinton vowed to capture the bombers and brought them to a swift justice.

On 04/21 Reuters reported Federal agents have arrested a suspect in the Oklahoma City bombing. Timothy James McVeigh, 27, was formally charged on Friday with the bombing. Brothers James and Terry Nichols, known friends of McVeigh and reported to share his extreme right-wing views, have been held since last week as material witnesses to the Oklahoma bombing.

Figure 1: Summary produced by our system using 24 news articles as input.

We are developing a multi-document summarization system to automatically generate a concise summary by identifying similarities and differences across a set of related documents. Input to the system is a set of related documents, such as those retrieved by a standard search engine in response to a particular query. Our work to date has focused on generating similarities across documents. Our approach uses machine learning over linguistic features extracted from the input documents to identify several groups of paragraph-sized text units which all convey approximately the same information. Syntactic linguistic analysis and comparison between phrases of these units is used to select the phrases that can adequately convey the similar information. This task is performed by the content planner of the language generation component and results in the determination of the summary content. Sentence planning and generation are then used to combine the phrases together to form a coherent whole. An example summary produced by the system is shown in Figure 1; this is a summary of 24 news articles on the Oklahoma

Timothy James McVeigh, 27, was formally charged on Friday with the bombing of a federal building on Oklahoma City which killed at least 65 people, the Justice Department said.
Timothy James McVeigh, 27, was formally charged on Friday with the bombing of a federal building on Oklahoma City which killed at least 65 people, the Justice Department said.
The first suspect, Gulf War veteran Timothy McVeigh, 27, was charged with the bombing Friday after being arrested for a traffic violation shortly after Wednesday's blast.
Federal agents have arrested suspect in the Oklahoma City bombing Timothy James McVeigh, 27. McVeigh was formally charged on Friday with the bombing.
Timothy McVeigh, the man charged in the Oklahoma City bombing, had correspondence in his car vowing revenge for the 1993 federal raid on the Branch Davidian compound in Waco, Texas, the Dallas Morning News said Monday.

Figure 2: A collection of similar paragraphs (part of a *theme*) from the Oklahoma bombing event.

City bombing.

The key features of our work are:

1. **Identifying themes**. Given the 24 input articles, how can we identify the similar paragraphs shown in Figure 2? We term each set of similar paragraphs (or generally, text units) a *theme* of the input articles. There may be many themes for a set of articles; for these 24 articles, there are 9 themes. Unlike most systems that compute a measure of similarity over text, our features extend beyond simple word matching and include entire noun phrases, proper nouns, and semantic senses; we also utilize positional and relational information between pairs of words. We ran a series of experiments that compared the use of different features and the baseline provided by standard information retrieval matching techniques, establishing that targeted selection of linguistic features is indeed beneficial for this task.
2. **Information fusion**. Given the subset of one theme extracted from the articles, shown in Figure 2, how can we determine that only the phrases resulting in the sentence "Timothy James McVeigh, 27, was formally charged on Friday with the bombing." should be represented in the summary? We have developed and implemented a novel algorithm for this task which analyzes grammatical structure extracted from each theme with off-the-shelf tools. Our information fusion algorithm compares predicate argument structures of the phrases within each theme to determine which are repeated often enough to be included in the summary. This process yields an average accuracy of 79% when tested on a collection of text units from multiple documents already clustered into themes by hand.
3. **Text reformulation**. Once the content of the summary has been determined, how can we fluently use the similar phrases in novel contexts? Simply stringing the phrases together can produce ungrammatical results because phrases are placed in new syntactic contexts. We have developed an algorithm that maps the predicate argument structure of input document phrases to arguments expected by FUF/SURGE [Elhadad 1993; Robin 1994], a robust language generation system. This has required developing new techniques for identifying constraints on realization choice (e.g., on the order of circumstantial roles such as time, location, instrument, etc.), using surface features in place of the semantic or pragmatic ones typically used in language generation.

Related Work

To allow summarization in arbitrary domains, most current systems use sentence extraction, identifying and extracting key sentences from an input article using a variety of different criteria. These approaches have all been developed to produce a summary of a single input document. One recent statistical approach [Kupiec *et al.* 1995] uses a corpus of articles with summaries for training to identify the features of sentences that are typically included in abstracts. Other recent approaches use lexical chains [Barzilay and Elhadad 1997], sentence position [Lin and Hovy 1997], discourse structure [Marcu 1997; Marcu 1998], and user features from the query [Strzalkowski *et al.* 1998] to find key sentences.

While most work to date focuses on summarization of single articles, early work is emerging on summarization across multiple documents. Radev and McKeown [1998] use a symbolic approach, pairing information extraction systems with language generation. The result is a domain dependent system for summarization of multiple news articles on the same event, highlighting how perspective of the event has changed over time. In ongoing work at Carnegie Mellon, Yang et al. [1998] are developing statistical techniques to identify similar sentences and phrases across articles. While they have developed a novel statistical approach to identify similar sentences, their system simply lists all extracted similar sentences as the summary. Mani and Bloedorn [1997] use spreading activation and graph matching to compute similarities and differences between the salient topics of two articles. Output is presented as a set of paragraphs which contain similar and distinguishing words, emphasized in different fonts. The problem is a redundant summary since no synthesis of results through generation is attempted.

System Architecture

Our system follows a pipeline architecture, shown in Figure 3. Input to the system is a set of related documents, such as those retrieved by a standard search engine. The analysis component of the system breaks documents into smaller text units and then computes a similarity metric across text units, regardless of the source document. Once similar paragraphs are identified, they are passed to the generation component which further identifies and selects information to be reformulated as coherent text.

The analysis, or similarity computation component takes as input a set of articles that have been previously identified as being on the same topic. In building our system, we used

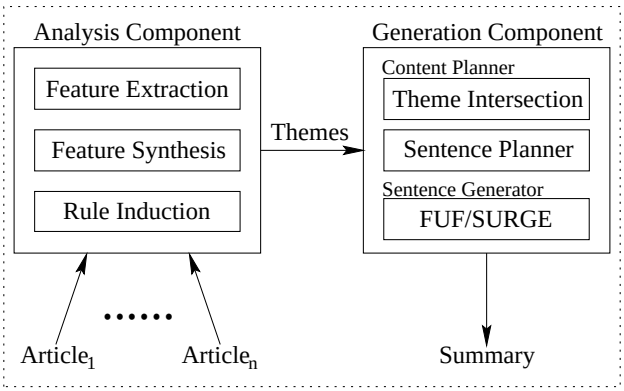

Figure 3: System architecture.

articles from the pilot Topic Detection and Tracking (TDT) corpus [Allan *et al.* 1998] for training and testing. The analysis component breaks the article into paragraph-sized units for comparison, and then extracts a set of linguistic and positional features for input into the similarity algorithm. We construct a vector for each pair of paragraphs, representing matches on each of the different features. These vectors are passed to a machine learning algorithm [Cohen 1996] which combines these features into a classifier using the most discriminating ones to judge similarity. Output is a listing of binary decisions on paragraph pairs, with each pair classified as containing similar or dissimilar text units. The similarity decisions drive a subsequent clustering algorithm, which places the most related paragraphs in the same group, and thus identifies themes.

The generation component consists of a content planner, sentence planner, and a sentence generator. Since input is full text, the process of selecting and ordering content is quite different from typical language generators. For each theme, the content planner identifies phrases within the paragraphs of a theme that are close enough to other phrases in the theme that they can be included in the summary. It does this by producing a predicate-argument structure for each sentence in each input paragraph, comparing arguments to select phrases that are similar. The sentence planner then determines which phrases should be combined into a single, more complex sentence, looking again at constraints from the input document as well as common references between phrases. Finally, the constituent structure produced by these two stages is mapped to the functional representation required as input by FUF/SURGE [Elhadad 1993; Robin 1994].

Document Analysis

Our definition of similarity is different than the one adopted in most text matching tasks (such as information retrieval) because of two factors: first, the size of the unit of text affects what is similar; documents have a lot of information, so even a modest amount of common elements can make two documents similar. Second, our goal is different. We are looking for text units that are quite close in meaning, not just for topical similarity.[1]

We thus consider two textual units similar if they both refer to the same object and that object performs the same action in both textual units, or the object is described in the same way in both of them. Such a common description must be more than just a single modifier. For example, the following two sentences satisfy our criteria for similarity,

> Britain Thursday sent back to the United States a possible suspect in the Oklahoma bomb blast, the interior ministry said.
>
> "A possible suspect connected with the Oklahoma bomb has been returned to the United States by the U.K. immigration service," a ministry statement said.

because they both refer to a common event (the returning of the suspect to the United States). On the other hand, the following sentence

> Federal agents have arrested a suspect in the Oklahoma City bombing and U.S. television networks reported he was a member of a paramilitary group called the Michigan Militia.

is not similar to the above, as it focuses on a different suspect and his paramilitary connections. Existing methods based on shared words are likely to identify all these sentences as related, since they all contain key words such as "suspect", "Oklahoma", and "bomb".

Traditional metrics for determining similarity among textual units compare term occurrence vectors using frequencies modified by the rarity of each term (the TF*IDF approach) [Salton and Buckley 1988]. Terms are single words, occasionally with simple transformations such as stemming, although sometimes multi-word units and collocations have been used [Smeaton 1992]. Since we are aiming for a different, more fine-grained notion of similarity and operate on much shorter texts than information retrieval does, we explored a number of alternative features. Our features draw on a number of linguistic approaches to text analysis, and are based on both single words and simplex noun phrases (sequences of adjectives and nouns with no embedded recursion). We thus consider the following potential matches between text units:

- **Word co-occurrence**, i.e., sharing of a single word between text units. Variations of this feature restrict matching to cases where the parts of speech of the words also match, or relax it to cases where the stems of the two words are identical.
- **Matching noun phrases**. We use the LinkIt tool [Wacholder 1998] to identify simplex noun phrases and match those that share the same head.
- **WordNet synonyms**. The WordNet semantic database [Miller *et al.* 1990] provides sense information, placing words in sets of synonyms (*synsets*). We match as synonyms words that appear in the same synset.
- **Common semantic classes for verbs**. Levin's [1993] semantic classes for verbs have been found to be useful for determining document type and text similarity [Klavans

[1] Note that we start with a set of documents about the same topic, which is the usual goal of information retrieval systems.

and Kan 1998]. We match two verbs that share the same semantic class in this classification.

In addition to the above *primitive* features that all compare single items from each text unit, we use *composite* features that combine pairs of primitive features. Our composite features impose particular constraints on the order of the two elements in the pair, on the maximum distance between the two elements, and on the syntactic classes that the two elements come from. They can vary from a simple combination (e.g., "two text units must share two words to be similar") to complex cases with many conditions (e.g., "two text units must have matching noun phrases that appear in the same order and with relative difference in position no more than five"). In this manner, we capture information on how similarly related elements are spaced out in the two text units, as well as syntactic information on word combinations. Matches on composite features indicate combined evidence for the similarity of the two units.

To determine whether the units match overall, we employ a machine learning algorithm [Cohen 1996] that induces decision rules using the features that really make a difference. A set of pairs of units already marked as similar or not by a human is used for training the classifier. We have manually marked a set of 8,225 paragraph comparisons from the TDT corpus for training and evaluating our similarity classifier.

For comparison, we also use an implementation of the TF*IDF method which is standard for matching texts in information retrieval. We compute the total frequency (TF) of words in each text unit and the number of units in our training set each word appears in (DF, or document frequency). Then each text unit is represented as a vector of TF*IDF scores, calculated as

$$TF(\text{word}_i) \cdot \log \frac{\text{Total number of units}}{DF(\text{word}_i)}$$

Similarity between text units is measured by the cosine of the angle between the corresponding two vectors (i.e., the normalized inner product of the two vectors), and the optimal value of a threshold for judging two units as similar is computed from the training set.

After all pairwise similarities between text units have been calculated, we utilize a clustering algorithm to identify themes. As a paragraph may belong to multiple themes, most standard clustering algorithms, which partition their input set, are not suitable for our task. We use a greedy, one-pass algorithm that first constructs groups from the most similar paragraphs, seeding the groups with the fully connected subcomponents of the graph that the similarity relationship induces over the set of paragraphs, and then places additional paragraphs within a group if the fraction of the members of the group they are similar to exceeds a preset threshold.

Language Generation

Given a group of similar paragraphs—a theme—the problem is to create a concise and fluent fusion of information in this theme, reflecting facts common to all paragraphs. A straightforward method would be to pick a representative sentence that meets some criteria (e.g., a threshold number of common content words). In practice, however, any representative sentence will usually include embedded phrase(s) containing information that is *not* common to all sentences in the theme. Furthermore, other sentences in the theme often contain *additional* information not presented in the representative sentence. Our approach, therefore, uses intersection among theme sentences to identify phrases common to most paragraphs and then generates a new sentence from identified phrases.

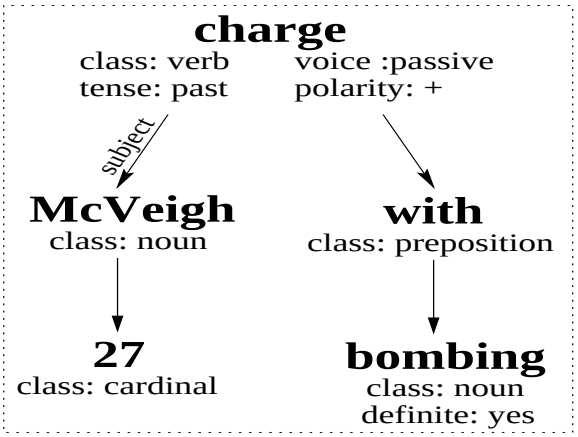

Figure 4: Dependency grammar representation of the sentence "McVeigh, 27, was charged with the bombing".

Intersection among Theme Sentences

Intersection is carried out in the content planner, which uses a parser for interpreting the input sentences, with our new work focusing on the comparison of phrases. Theme sentences are first run through a statistical parser [Collins 1996] and then, in order to identify functional roles (e.g., subject, object), are converted to a dependency grammar representation [Kittredge and Mel'čuk 1983], which makes predicate-argument structure explicit.

We developed a rule-based component to produce functional roles, which transforms the phrase-structure output of Collins' parser to dependency grammar; function words (determiners and auxiliaries) are eliminated from the tree and corresponding syntactic features are updated. An example of a theme sentence and its dependency grammar representation are shown in Figure 4. Each non-auxiliary word in the sentence has a node in the representation, and this node is connected to its direct dependents.

The comparison algorithm starts with all subtrees rooted at verbs from the input dependency structure, and traverses them recursively: if two nodes are identical, they are added to the output tree, and their children are compared. Once a full phrase (verb with at least two constituents) has been found, it is confirmed for inclusion in the summary.

Difficulties arise when two nodes are not identical, but are similar. Such phrases may be paraphrases of each other and still convey essentially the same information. Since theme sentences are *a priori* close semantically, this significantly

constrains the kind of paraphrasing we need to check for. We verified this assumption by analyzing paraphrasing patterns through themes of our training corpus, drawn from the TDT (see the Evaluation section). We found that that a high percentage of paraphrasing is at the "surface" level, using semantically related words and syntactic transformations.

When similar nodes are detected, the algorithm tries to apply an appropriate paraphrasing rule, drawn from the corpus analysis. For example, if the phrases "group of students" and "students" are compared, then the *omit empty head* rule is applicable, since "group" is an empty noun and can be dropped from the comparison, leaving two identical words, "students". In the case that a matching paraphrase rule cannot be found, the comparison is finished and the phrase is dropped. Other examples of paraphrase include ordering of syntactic components in the sentence (e.g., inversion of subject and object because of passive), realization of the predicate in a main clause versus a relative clause, and use of synonyms based on WordNet.

For the theme in Figure 2, intersection results in the clause "McVeigh was formally charged on Friday with the bombing" and the description "Timothy James McVeigh, 27".

Sentence Generation Component

The primary task of this component is to construct the summary text from phrases selected from the theme. During this process, the system orders phrases based on their time sequence of appearance, adds additional information needed for clarification (e.g., entity descriptions, temporal references, and newswire source references) that appeared in the full articles but not in the theme [Barzilay *et al.* 1999], and maps the dependency representations to an English sentence. Output of this component is the full summary, such as the one shown in Figure 1.

Ordering of phrases within the summary is based on chronological order inferred from their appearance in the original text. To do this, for each theme, we record a date that is the earliest date on which any of its sentences were published. We sort extracted phrases according to these dates.

Sentence realization is done by mapping the dependency structure to the input required by FUF/SURGE [Elhadad 1993; Robin 1994]. The use of a language generator, with full grammar, means that the system can combine phrases together in ways that did not occur in the input text. A phrase that occurred as the main clause in an input text, for example, may be realized in the summary as a subordinate clause. The grammar is used to determine inflections, subject-verb agreement, and word orderings. This gives us the ability to generate coherent, fluent text as opposed to rigidly using extracted sentences in the same ways they were used in the input.

Sentence generators such as FUF/SURGE, however, typically require a set of sentential semantic roles as input. They were developed for use within a system that builds a semantic sentence representation from data. However, in our case, we have no access to semantic information; all that we have are syntactic roles, derived from the parse tree for the sentence. In many cases, this simply means less work for FUF/SURGE. Processing starts using roles such as subject, object, and main verb instead of deriving these syntactic roles from roles such as agent, goal, and predicate. However, in other cases, it means producing a sentence with only vague knowledge of the function of a particular constituent. For example, for circumstantials such as time, location, or manner, the summarizer can only determine from the input that it is a circumstantial and cannot determine its type. The specific function determines the options for paraphrasing (e.g., whether it appears at the beginning or end of the sentence, or next to some other constituent). We use the ordering derived from the input to constrain these options and leave the specific function unspecified. We will continue to look at cases where different kinds of syntactic constraints on paraphrasing can be derived from the input in place of semantic or pragmatic constraints.

System Status

We have an initial system prototype which includes implementation of system components and integration of components for a number of examples. More work needs to be done to tune output of the theme identifier to produce better input for the generation component. Our experiments show that high recall and low precision in finding themes is better for the generator than low recall and high precision. This is because the content planner, which performs information fusion, makes a more thorough analysis and weeds out sentences that are not similar when comparing phrases. Thus, it can handle errors in its input, but it cannot handle missing input. As the noise in its input increases, the size of the intersection decreases. In such cases, the output summary will be short, but the system will never generate nonsensical output.

Given noisy input (i.e., dissimilar articles), the system components degrade in different ways. Given poor input, the theme identification component may produce low-quality themes. But, as noted above, the intersection component will weed out these errors and produce an empty, or small intersection, and a short summary. Also, we are focusing on articles that have already been judged similar by some means (e.g., by an information retrieval engine), so it is unlikely that our system will receive markedly different documents as input.

Evaluation

Given that our implementation of components is complete, but full integration of components is still underway, our evaluation at this stage focuses on quantifying results for each of the system components separately. Following the division in our system architecture, we identify three separate questions to which results can be produced independently and to which the system's answers can be evaluated. We elaborate on these three questions and on how we quantitatively measure performance on them in the first subsection, and then present the results of the evaluation on specific data collections.

Identifying Themes

The first question we address is how good our system is in identifying themes, or similar paragraphs. We quantify the answer to this question by presenting to the system a list of pairs of paragraphs (all extracted from documents on the same topic) and measuring how many correct decisions the system makes on characterizing the paragraphs in each pair as similar or dissimilar.

Since placing incoming documents into topical clusters is a non-trivial task for large document collections, we use as input a set of articles already classified according to subject matter, the pilot Topic Detection and Tracking (TDT) corpus. The TDT effort, sponsored by DARPA and NIST, aims to promote new research on document classification and clustering; as a necessary step towards comparing different systems on these tasks, a corpus of articles from written and broadcast news sources (Reuters and CNN) is marked with subject categories that correspond to our criteria for selecting similar documents (e.g., "Oklahoma City bombing" or "Pentium chip flaw"). We are using the Reuters part of the first such collection made available in early 1998 (the TDT pilot corpus), which contains 16,000 articles (see `http://morph.ldc.upenn.edu/Catalog/LDC98T25.html` for more details).

We selected 6 of the 25 topical categories in the pilot TDT corpus, favoring categories that had a significant number of member documents. For each such category, we selected articles from randomly chosen days, for a total of 30 articles.

Documents in each topical category are broken into paragraphs, and paragraphs from different documents in the same theme are compared using the various alternatives described in our document analysis section. For example, our first category about two Americans lost in Iraq has 61 paragraphs across 8 selected articles, and $61 \cdot 60 / 2 = 1,830$ comparisons are made between those 61 paragraphs. All the selected categories have 264 paragraphs and 8,225 comparisons between paragraphs, calculated as

$$\sum_{i=1}^{6} \binom{N_i}{2}$$

where N_i is the number of paragraphs in category i. We randomly divided these pairs of paragraphs into a training set (6,225 pairs) and a testing set (2,000 pairs). These 8,225 pairs were manually compared independently by two evaluators (who subsequently met and reconciled differences), and classified as either similar or dissimilar.

We extracted the primitive features discussed in the document analysis section, calculated our composite features, and trained both the machine learning model that uses these features and the TF*IDF classifier on the training set. Our feature-based approach was able to recover 39.7% of the similar pairs of paragraphs with 60% precision and had an overall accuracy over both similar and dissimilar pairs of 97%, while the corresponding numbers for the TF*IDF method were 31.4% recall of similar paragraphs, 41.8% precision and 96.5% overall accuracy.

Note that since we have a fine-grained model of similarity, most paragraphs are dissimilar to most other paragraphs. As a result, the baseline method of always guessing "dissimilar" will have a very high accuracy (percentage of total correct answers), 97% in our experiments. However, as in comparable information retrieval tasks with no pre-constructed, balanced evaluation document sets, it is important to focus primarily on evaluation results for the rarer similar pairs (respectively, on the documents relevant to a particular query), rather than all pairs (or documents) in the collection. Our results indicate that our approach outperforms traditional text matching techniques, especially on the harder-to-find similar paragraphs.

Information Fusion

The second evaluation question addresses the performance of our content planner, measuring how well we identify phrases that are repeated throughout the multiple paragraphs within a theme. We present the system with several manually constructed themes and compare the system-produced collection of phrases to manually identified common phrases. Manually constructed themes allow us to obtain an independent view of the content planner's performance, without including any misclassifications that the first stage of the system makes. Then standard measures such as precision and recall can be used to quantitatively compare the system's results to the reference list.

To carry out this evaluation, we constructed five themes each containing 3.8 paragraphs on average. To do this, we used themes automatically constructed by the first stage of the system and edited errors by hand. Repeated information was manually extracted from each theme, producing seven sentence-level predicate-argument structures corresponding to phrases that should be included in the summary. Then we applied our intersection algorithm which proposed six predicate-argument structures for the summary and was able to correctly identify 81% of the subjects, 85% of the main verbs, and 72% of the other constituents in our list of model predicate-argument structures.

Generating Sentences

The final evaluation task is to assess how well our surface generation component performs on the task of putting together the extracted phrases in coherent sentences. We evaluate performance on this task by asking humans to rate each produced sentence in terms of fluency, but not in terms of content. In fact, the evaluators do not see the original documents, and thus base their judgements only on the quality of the produced sentences in isolation.

This is an important first step in evaluation; given that we are taking apart sentences and putting them together in novel ways, we need to measure how well we do at producing fluent and grammatical sentences. While this is not an issue for extraction-based summarization systems, robustness at the sentence generation level is critical to success in our approach. We are also looking at alternative methods for rating sentences on fluency. A logical next step will be to evaluate sentences in context, measuring overall coherence. In addition, the grades that people assign to sentences are subjective; an alternative is to ask evaluators to order sentences

The defense department said an OH-58 military U.S. scout helicopter made an emergency landing in the North Korea friday. **Score: 95**
North Korea said it shot the helicopter down over the its territory. **Score: 80**
Richardson cancelled other discussions that it was taken place on the recent nuclear U.S.-NORTH KOREA agreement. **Score: 50**

Figure 5: Three of the sentences automatically generated by our system and the fluency scores assigned to them.

on the basis of fluency, presenting them with the system's output together with sentences written by humans.

We evaluated the fluency of our sentence generator by having it generate 31 sentences from the correct list of predicate argument structures used in the second evaluation experiment. Each of these sentences was read by an independent evaluator, who graded it on fluency with a numeric score between 0 and 100. This process resulted in an average score of 79.5, with 15 of the 31 sentences scored at 90 or more. Figure 5 shows three of the generated sentences and their assigned scores.

Conclusions and Future Work

This paper presents a novel architecture for accomplishing summarization of multiple documents in any domain. In order to achieve this, our work builds on existing tools, such as a parser and generator, as a springboard to take us further than would otherwise be possible. This has allowed us to address key higher-level issues including the development of a paragraph similarity module using learning over a set of linguistic features, an algorithm for identifying similar clauses within the resulting themes, and sentence generation techniques to combine clauses in novel ways within new contexts. These new features enable the development of a multi-document summarizer that uses reformulation to produce natural and fluent text. Unlike sentence extraction techniques which present a concatenated list of sentences or phrases picked on the basis of statistical or locational criteria, our system presents a synthesized summary, created using both statistical and linguistic techniques.

In the future, we plan to experiment with an alternative evaluation approach that rates the produced summary as a whole. We are in contact with professional journalists who perform the task of synthesizing an article from multiple news sources. One possibility is to ask them to evaluate the summaries directly; another is to identify through user analysis other measures that they internally use to arrive at "good" articles and try to apply them to the summarization task.

Acknowledgments

This material is based upon work supported by the National Science Foundation under grant No. IRI-96-1879. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[Allan *et al.* 1998] James Allan, Jaime Carbonell, George Doddington, Jon Yamron, and Y. Yang. Topic Detection and Tracking Pilot Study: Final Report. In *Proceedings of the Broadcast News Understanding and Transcription Workshop*, pages 194–218, 1998.

[Barzilay and Elhadad 1997] Regina Barzilay and Michael Elhadad. Using Lexical Chains for Text Summarization. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 10–17, Madrid, Spain, August 1997. Association for Computational Linguistics.

[Barzilay *et al.* 1999] Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. Information Fusion in the Context of Multi-Document Summarization. In *Proceedings of the 37th Annual Meeting of the ACL*, College Park, Maryland, June 1999. Association for Computational Linguistics.

[Cohen 1996] William Cohen. Learning Trees and Rules with Set-Valued Features. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-96)*. American Association for Artificial Intelligence, 1996.

[Collins 1996] Michael Collins. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, California, 1996.

[Elhadad 1993] Michael Elhadad. *Using Argumentation to Control Lexical Choice: A Functional Unification Implementation*. PhD thesis, Department of Computer Science, Columbia University, New York, 1993.

[Kittredge and Mel'čuk 1983] Richard Kittredge and Igor A. Mel'čuk. Towards a Computable Model of Meaning-Text Relations Within a Natural Sublanguage. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, pages 657–659, Karlsruhe, West Germany, August 1983.

[Klavans and Kan 1998] Judith Klavans and Min-Yen Kan. The Role of Verbs in Document Access. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (ACL/COLING-98)*, Montreal, Canada, 1998.

[Kupiec *et al.* 1995] Julian M. Kupiec, Jan Pedersen, and Francine Chen. A Trainable Document Summarizer. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 68–73, Seattle, Washington, July 1995.

[Levin 1993] Beth Levin. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, Illinois, 1993.

[Lin and Hovy 1997] Chin-Yew Lin and Eduard Hovy. Identifying Topics by Position. In *Proceedings of the 5th ACL Conference on Applied Natural Language Processing*, pages 283–290, Washington, D.C., April 1997.

[Mani and Bloedorn 1997] Inderjeet Mani and Eric Bloedorn. Multi-document Summarization by Graph Search and Matching. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 622–628, Providence, Rhode Island, 1997. American Association for Artificial Intelligence.

[Marcu 1997] Daniel Marcu. From Discourse Structures to Text Summaries. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 82–88, Madrid, Spain, August 1997. Association for Computational Linguistics.

[Marcu 1998] Daniel Marcu. To Build Text Summaries of High Quality, Nuclearity is not Sufficient. In *Proceedings of the AAAI Symposium on Intelligent Text Summarization*, pages 1–8, Stanford University, Stanford, California, March 1998. American Association for Artificial Intelligence.

[Miller *et al.* 1990] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An On-Line Lexical Database. *International Journal of Lexicography*, **3**(4):235–312, 1990.

[Paice 1990] Chris D. Paice. Constructing Literature Abstracts by Computer: Techniques and Prospects. *Information Processing and Management*, **26**:171–186, 1990.

[Radev and McKeown 1998] Dragomir R. Radev and Kathleen R. McKeown. Generating Natural Language Summaries from Multiple On-Line Sources. *Computational Linguistics*, **24**(3):469–500, September 1998.

[Robin 1994] Jacques Robin. *Revision-Based Generation of Natural Language Summaries Providing Historical Background: Corpus-Based Analysis, Design, Implementation, and Evaluation*. PhD thesis, Department of Computer Science, Columbia University, New York, 1994. Also Columbia University Technical Report CU-CS-034-94.

[Salton and Buckley 1988] G. Salton and C. Buckley. Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, **25**(5):513–523, 1988.

[Smeaton 1992] Alan F. Smeaton. Progress in the Application of Natural Language Processing to Information Retrieval Tasks. *The Computer Journal*, **35**(3):268–278, 1992.

[Strzalkowski *et al.* 1998] Tomek Strzalkowski, Jin Wang, and Bowden Wise. A Robust Practical Text Summarization. In *Proceedings of the AAAI Symposium on Intelligent Text Summarization*, pages 26–33, Stanford University, Stanford, California, March 1998. American Association for Artificial Intelligence.

[Wacholder 1998] Nina Wacholder. Simplex NPs Clustered by Head: A Method For Identifying Significant Topics in a Document. In *Proceedings of the Workshop on the Computational Treatment of Nominals*, pages 70–79, Montreal, Canada, October 1998. COLING-ACL.

[Yang *et al.* 1998] Yiming Yang, Tom Pierce, and Jaime Carbonell. A Study on Retrospective and On-Line Event Detection. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, August 1998.