Information Retrieval, 1, 35–67 (1999)

Summarizing Similarities and Differences Among Related Documents

INDERJEET MANI imani@mitre.org
ERIC BLOEDORN bloedorn@mitre.org
The MITRE Corporation, W640, 11493 Sunset Hills Road Reston, VA 22090, USA.

Received August 28, 1997; Revised June 8, 1998; Accepted June 8, 1998

Abstract. In many modern information retrieval applications, a common problem which arises is the existence of multiple documents covering similar information, as in the case of multiple news stories about an event or a sequence of events. A particular challenge for text summarization is to be able to summarize the similarities and differences in information *content* among these documents. The approach described here exploits the results of recent progress in information extraction to represent salient units of text and their relationships. By exploiting meaningful *relations* between units based on an analysis of text cohesion and the *context* in which the comparison is desired, the summarizer can pinpoint similarities and differences, and align text segments. In evaluation experiments, these techniques for exploiting cohesion relations result in summaries which (i) help users more quickly complete a retrieval task (ii) result in improved alignment accuracy over baselines, and (iii) improve identification of topic-relevant similarities and differences.

Keywords: text summarization, information retrieval, natural language processing

1. Introduction

With the mushrooming of the quantity of on-line text information, triggered in part by the growth of the World Wide Web, it is especially useful to have tools which can help users digest information content. Text summarization attempts to address this problem by taking a partially-structured source text, extracting information content from it, and presenting the most important content to the user in a manner sensitive to the user's needs. Clearly, some sort of summarization is indispensible for dealing with these massive and unprecedented amounts of information. Now, in many modern information retrieval applications, a common problem which arises is the existence of multiple documents covering similar information, as in the case of multiple news stories about an event or a sequence of events. A particular challenge for text summarization is to be able to summarize the similarities and differences in information *content* among these documents.

A variety of approaches exist for extracting content for multi-document summarization, which vary in the extent of domain dependence. In constrained domains, e.g., articles on terrorist events, natural language message understanding systems can extract relationships between entities, such as the location and target of a terrorist event. Such relationships can be used to identify areas of agreement and disagreement across texts [34]. For arbitrary text, such techniques do not apply, and instead, word-based content representations have traditionally been exploited (e.g., [49]). However, as recent progress in information extraction reveals (e.g., [39]), it is possible to extract not just salient words but also phrases and

proper names from unrestricted text in a highly scalable manner. As a result, such extraction techniques are now being exploited in general purpose information retrieval tools (e.g., [10], [16], [18], [42], [56]).

The focus of the work described here is to provide a tool for analyzing document collections such as multiple news stories about an event or a sequence of events. Given a collection of such documents, the tool can be used to detect and align similar regions of text among members of the collection, and to detect relevant differences among members. It is worth noting here that the context-sensitive aspect of summarization is particularly important in this task. Depending on the users' interest, there may be many different sets of similarities and differences. Our summarization approach represents context in terms of a topic, which is a set of words which can be drawn from a user query or profile. Given a topic and a pair of related news stories, our method identifies salient regions of each story related to the topic, and then compares them, summarizing similarities and differences. Until now, only portions of this approach have been described [29], [30]. In this more detailed paper, we present our approach in more general terms, and include additional experimental data and techniques.

2. Overall Approach to Summarization

We will first clarify the variety of summarization being considered here. In general, there are many varieties of automatic summarization. A classical distinction (e.g., [40]) is that a summary can be "indicative", used to alert the user as to what the source is about (thus helping her decide whether the source might be worth reading) or it can be "informative", attempting, within the constraints of the particular compression desired, to stand in place of the source. A summary can also be "evaluative" [55] offering a critique of the source, as in a book review. In some situations (e.g., a scientific abstracting service), a high degree of fluency and connectedness of the summary text may be called for; in contrast, when a summary is used merely as a gist, more fragmentary or less connected text may suffice. A summary can be in the form of an extract, or an abstract; it can stand by itself, or it can be linked to the source or to more detailed summaries. A summary can cover a single source, or multiple sources. Finally, the audience for a summary can vary. Traditionally, abstracts were written by authors or by professional abstractors with the goal of dissemination to a particular - usually broad - readership community. These "generic" abstracts were traditionally used as surrogates for full-text. As our computing environments continue to accommodate increased full-text searching, browsing, and personalized information filtering, "user-focused" abstracts, which are customized to the user's interests, have assumed increased importance. As will be made clear, we report here on techniques for generating user-focused, indicative, moderately fluent, extract-based summaries for multiple sources.

Automatic text summarization can be characterized as involving three phases of processing: analysis, refinement, and synthesis[1]. The analysis phase builds a representation of the source text. The refinement phase transforms this representation into a summary representation, condensing text content by selecting salient information. The synthesis phase takes the summary representation and renders it in natural language using appropriate presentation techniques.

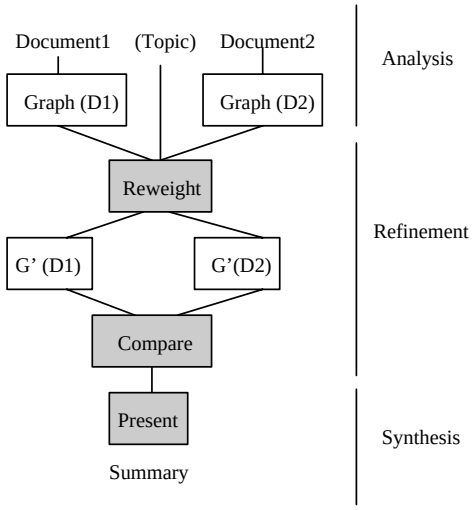

Figure 1. Multi-document Summarization Approach

In our approach, the analysis phase builds a representation based on domain-independent information extraction techniques. Text items such as words, phrases, and proper names are extracted and represented in a graph. In particular, nodes in the graph represent word instances at different positions, with phrases and names being formed out of words. The refinement phase exploits cohesion relationships (to be discussed below) between term instances to determine what is salient. Finally, the synthesis phase takes the set of salient items discovered by the refinement phase, and uses that set to extract text from the source to present a summary.

Of course, if we are able to discover, given a topic and a pair of related documents, salient items of text in each document which are related to the topic, then these salient items can be compared to establish similarities and differences between the document pair. This forms the basis for a general scheme for multi-document summarization. As shown in Figure 1, given a pair of documents, the Analysis phase builds a graph for each document. In the Refinement phase, salient nodes in each graph related to the topic are discovered, using a spreading activation search of the graph. The set of activated (i.e., reweighted) nodes for each graph are then compared; comparing just these salient items results in fewer comparisons than comparing the entire body of the two texts. Finally, the result of this comparison is used in a synthesis phase to extract sentences. Thus, given a pair of related news stories about an event or a sequence of events, the problem of finding similarities and differences becomes one of comparing text items which have been activated by a common topic. This allows different comparisons to be generated, based on the choice of common topic.

The overall approach in Figure 1 extends easily to comparing sets of documents rather than pairs, with some restrictions on the presentation strategies in the synthesis stage. In this paper, we explore several different synthesis techniques in multi-document summarization,

including identifying similarities and differences, and aligning text across multiple documents. While the former method applies to sets of documents, the latter (as will be explained in Section 7.2.3) is more suited to pairwise comparison.

Although our interest here is in user-focused summaries, the overall approach can be extended to deal with "generic" summaries. In that case, in Figure 1, instead of reweighting based on a query (using spreading activation), we weight the nodes based on a conventional weighting metric, such as tf.idf. The comparison and presentation steps in Figure 1 (after applying a segment finding operation on these graphs, described in Section 6.2) remains as in the user-focused case, allowing for generic summaries to be produced.

These summarization techniques yield useful summaries when applied to large quantities of unrestricted text, of the kind found on the World Wide Web. To investigate the degree of scalability of the approach, we investigate measures of algorithmic time and space complexity, timing results, and evaluation metrics for effectiveness. The approach has been embedded in an information retrieval tool which allows the user to issue queries to Internet search engines running queries against the World Wide Web. The user can choose any set of hits to summarize. The system offers a set of common terms, which the user can select one or more from, to constitute the common topic. For such applications, summarization needs to be able to help users minimize reading time on longer documents, to enable them to quickly select relevant information, and discard irrelevant information. In such situations, the summaries need not be highly polished, but must be intelligible enough to stand on their own to be archived or linked in for later perusal.

3. Distinguishing Features of Our Approach

Our summarization approach in Figure 1 has three main distinguishing features:

1. We explicitly identify commonalities and differences across documents. This may be contrasted with approaches such as [43], where (queries and) documents are matched for similarity, with statistically prominent terms in each document being highlighted. Among the advantages of identifying commonalities and differences is that in addition to the commonalities telling us what information is salient with respect to the topic in the entire set, the differences tell us what's unique about each document. Thus, if the set of documents is ordered, say, in chronological order, the differences for the latest document tells us what's *novel* in it (with respect to the current topic). Novelty, in turn, is rather fundamental to summarization, and the ability to distinguish what's new from a sequence of similar documents retrieved for a query is of practical value.

2. We are able to identify the salience of different *regions* of text in a document with respect to a query. This is in keeping with the assumption underlying much summarization research that location and linear order of text items are important (e.g., [15], [26], [41]) in determining what's salient. This might be achieved by a passage-level relevance ranking approach [5]. However, choosing the best window size for identifying passages is a problem, whether one uses fixed-length overlapping windows, or "discourse" windows (e.g., sentences, paragraphs, sections). If the window size is too small, one may end up with a set of adjacent windows which individually contribute little relevance information but which as a whole are highly relevant. If the window size is too large, it may

include too much irrelevant information[2]. Instead of using a fixed window size, or paying the increased time complexity of varying the window size dynamically, we use a text representation which assigns weights to different positions of a term (these term occurrences correspond to nodes in the graph representation). Further, identifying regions in each document relevant to the query allows us to compare just those regions, reducing the set of items to be compared for commonalities and differences.

3. Finally, we explore a model of text which takes into account how connected items in the text are. This therefore explores a similar model of connectivity to the approach of [49] and [51], where the strength of links (in their case based on similarity) between different text units is used to identify salient text units in one or more documents. However, instead of just using a cosine-similarity measure between word-based vectors for fixed-size text units, we assign weights to different word occurrences based on "cohesion" links between these occurrences, discovered in part based on information extraction techniques.

   The cohesion relations considered here include synonym/hypernymy relations, repetition, adjacency, and coreference. Cohesion itself is an abstract notion, expressing the intuition that certain relations help make the text "hang together", and in some sense cause portions of the text to "be about the same thing" [37][3]. While it is as a result a rather imprecisely defined concept, the linguistic devices grouped under text cohesion are directly observable. (Cohesion is often contrasted with *coherence*, which is a relation between larger units of text, typically sentences and clauses, which has to do with macro-level, deliberative structuring of the text, e.g., represented by text schemas and rhetorical structures [27], [31], [32], [35], [57]). In keeping with trends of improved information extraction capabilities, cohesion also appears to be a renewed focus of interest among researchers in the text summarization field, e.g., [4], [8], [6]. Various cohesion relations are also of considerable interest in identifying topic shifts, e.g., [6], [24].

   This cohesion approach is one way to allow term occurrences which are indirectly related to the topic to emerge as salient; in turn this allows them to be candidates for comparison across documents. As these cohesion relations are represented as edges in the document's graph, the topology of the graph can be used to compute salience, using a spreading activation search algorithm. While spreading activation has been used in information retrieval [50], [12] and text summarization [46], [3] to search hand-created semantic nets or networks derived from thesauri, the focus here is on directly activating the richly structured graph for the text. In contrast to summarization approaches such as [46], which use information extraction techniques to build graphs for text about specific events such as corporate takeovers, the graphs built here apply to unrestricted text.

To conclude this section, our summarization approach is distinguished from others as follows. We use a graph representation which explicitly represents position and linear order of text items. The connectivity of text based on robustly-extracted linguistic relations between word occurrences is used to compute a salience function for the text as a whole with respect to a particular topic. Text regions deemed salient by this method are then used to address summarization of sets of documents with respect to a topic.

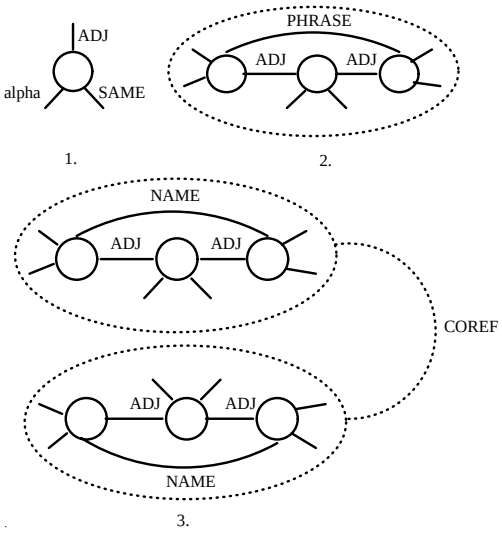

Figure 2. Graph Representation

4. Representing Meaningful Text Content

As shown in Figure 2, each node is a word *instance*, and has a distinct input position. Associated with each such node is a record characterizing the various features of the word in that position (e.g., absolute word position, position in sentence, weight). As shown in part 1 of the figure, a node can have adjacency links (ADJ) to textually adjacent nodes, SAME links to other instances of the same word, and other semantic links (represented by $alpha$). PHRASE links tie together strings of adjacent nodes which belong to a phrase (part 2). In part 3, we show a NAME link, as well as the COREF link between subgraphs, relating positions of name instances which are coreferential. NAME links can be specialized to different types, e.g., person, province, etc. In our work, the $alpha$ links are restricted to synonymy and hypernymy among words.

This representation is highly flexible, and general enough to encompass more fleshed-out linguistic representations; new relationships can be threaded easily into the graph. This results in a level of sentential analysis where words are grouped, where possible, into names and phrases, which in turn can make up a sentence. This representation allows for a degree of graceful degradation; in the worst case a sentence is just made up of a sequence of words.

Using this graph representation, the weights of nodes in the graph can be represented as an *activation vector*, as follows. A text D_i can be represented as a vector of weights $(wp_{i1}, .., wp_{ik}, .., wp_{in})$ where wp_{ik} is the weight of word position k in text i. In the initial activation vector, a given term has the same weight for all occurrences (positions). Given a topic T, the activation vector for D_i can be reweighted favoring term occurrences related to the topic, using the spreading activation techniques described in Section 6. Further reweighting is achieved by clipping the activation vector, as described in Section 6.2. A graph is implemented as an adjacency list, which requires $B.N$ storage, where N is the

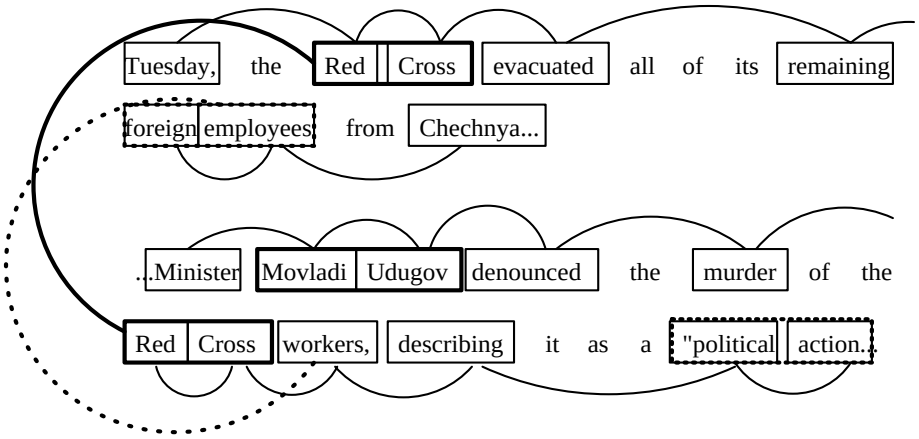

Figure 3. Graph Example. Names and phrases are grouped together. Dark edges: COREF; Light edges: ADJ; dotted edges: hypernym.

number of nodes in the graph and B is the maximum branching factor of a node. In practice B is observed to be small (maximum observed = 7, average = 3.5), so the graph requires $O(N)$ storage.

A sample graph is shown in Figure 3^4. We also introduce here Figure A.1 (Appendix), which serves as an example used throughout this paper to illustrate some of our observations about multi-document summarization. It shows the text (some of which is elided for reasons of space) of two related articles. The article in the left column is from the Associated Press (AP), the one in the right is from Reuters. Relevant subgraphs discussed in the paper are sketched. Some alignments between the two texts are shown boxed.

5. Tools for Building Meaningful Content Representations

The construction of text graphs for use in summarization requires at minimum a component for associating words with sentence and paragraph positions. We use a sentence and paragraph tagger which contains a very extensive regular-expression-based sentence boundary disambiguator [1]. Next, weights are computed for the words in the text. We use tf.idf [54] weighting, though any sensible weighting scheme could be substituted; here use is made of a reference corpus derived from the TREC [23] corpus. The weight $tf.idf_{ik}$ of term k in document i is given by:

$$tf.idf_{ik} = tf_{ik} * (ln(N) - ln(df_k) + 1) \tag{1}$$

where tf_{ik} = frequency of term k in document i, df_k = number of documents in the reference corpus in which term k occurs, N = total number of documents in the reference corpus.

It should be mentioned that the weights here are used for two purposes: first, as a filtering mechanism in extracting phrases (as described next), and second, to provide initial

weights for occurrences of query terms in the text, for use by the spreading activation search for query-related text regions (described in Section 6). Before beginning the spreading activation phase, the weights of all terms other than query terms are zeroed out; it is the spreading activation itself which determines the final weights based on the initial weights of query term occurrences. As a result, the main overall impact of the initial weighting scheme is in determining the heights of the "peaks" (i.e., occurrences of query terms) in the eventual distribution of weights of terms across text positions in the document. As a result, the summarization is not particularly sensitive to the particular weighting scheme; other weighting schemes have also been used effectively in our approach, including G^2 statistics [13]. Nor is the summarization particularly sensitive to different scaling factors used to normalize the tf_{ik} frequency; among the scaling factors we have used is the maximum frequency of any term in the document.

The remaining component tools include phrase extraction, name extraction, and synonym and hypernym extraction. The summarization algorithms described in this paper can work without all of these, but the use of these tools provides more structure to the graph, allowing use of these content-based features in summarization. To support phrase finding, MITRE's Alembic part-of-speech tagger [1] is invoked on the text. This tagger uses the rule-sequence learning approach of [9][5]. Names and relationships between names are extracted from the document using SRA's NameTag [25], a MUC6-fielded system. Phrases are extracted from the text using the word weights and part-of-speech and punctuation features. Finally, synonyms and hypernyms are extracted for the words using WordNet [38]. "Function" words, it should be noted, are stripped out using a stop-list, except where they occur within extracted names and phrases.

The name extraction techniques are now quite standard; for more details see [1], [25]. In what follows, we discuss the phrase and synonym/hypernym extraction analysis tools in more detail.

5.1. Phrase Extraction

Phrases are useful in summarization as they often denote significant concepts. In our application, phrases are of interest as summary descriptors rather than as index terms. Thus, we are not interested in extracting components of phrases, or syntactic variants of phrases; we require only a single phrase to describe a phrase-denoted concept. In general, one would prefer phrases which are as specific as possible; this is approximated by preferring longer phrases. Finally, we prefer phrases to be different from one another, to represent more of the conceptual content of the document. Our phrase extraction method finds candidate phrases using robust finite-state parsing techniques. We use several patterns defined over part-of-speech tags. One pattern, for example, uses the maximal sequence of one or more adjectives followed by one or more nouns. The weight of a candidate phrase is the average of the tf.idf weights of non-function (or content) words in the phrase, plus a factor β which adds a small bonus in proportion to the length of the phrase. We use a contextual parameter θ to avoid redundancy among phrases, by selecting each term in a phrase at most once in a window w. The size of the window is application dependent; our typical setting for news stories is the whole document. The weight of a phrase W of length n content words in document i is:

Table 1. Precision of 510 synonym links using two different techniques

Guessing "noun"	Part-of-speech tagging
.51 (264 links)	.67 (342 links)

$$wt(W,i) = \beta(n) + \frac{\sum_{k=1}^{n} \theta(ik) * tf.idf_{ik}}{n} \quad (2)$$

where $\theta(ik)$ is 0 if the word has been seen before in the window, and 1 otherwise.

5.2. Synonym and Hypernym Extraction

We now discuss the extraction of synonym and hypernym links. These are extracted using WordNet 1.5 [38]. Our algorithm takes every distinct word in the graph which is identified as a noun by the part-of-speech tagger, and looks up its synonyms and immediate hypernyms. First, if the word has an entry in a lookup table, that is used; otherwise, WordNet lookup is performed, with any hits being cached in the lookup table. Whenever a pair of words has a synonym or immediate hypernym link, an edge is drawn between all its instances. Although words tend to be highly polysemous in WordNet, no sense-disambiguation is carried out. Thus, even if a pair was linked by a very rare noun sense, the edge would be present on the graph. The reasoning is that automatic word sense disambiguation (e.g., distinguishing among different noun senses) within a text is quite hard. Further, an experiment described below suggests that investing in such an algorithm may not be worthwhile, as most of the synonyms found (using the part-of-speech method) are "correct".

For the texts in Figure A.1, we have good links like $captive \Leftrightarrow prisoner$, $head \Leftrightarrow chief \Rightarrow leader$, $ambassador \Rightarrow diplomat$, $assault \Leftrightarrow attack$, $residence \Rightarrow house$, $reception \Rightarrow party$[6]. Examples of bad links due to lack of sense disambiguation are $sister \Rightarrow member$, $head \Leftrightarrow question$.

To get a better handle on the performance of this method, we conducted a small experiment to examine precision of synonym linking over 17 articles (11986 words) with 510 synonym links in all. The articles were drawn from a collection of Internet news articles (where each article was on a different topic). (More details of the collection are described in Section 9.2.) A synonym link was judged correct if the linked pair of words appeared to have the same sense, given the context in which they each appeared in the article. As mentioned earlier, our algorithm takes every distinct word in the graph which is identified as a noun by the part-of-speech tagger, and looks up its synonyms and immediate hypernyms in WordNet. To compare with what would happen without part-of-speech tagging, we used a "dumb" baseline of treating every word to be looked up as a noun, leaving WordNet to do the rest. Table 1 shows the results under these two conditions. It can be seen from Table 1 that over two-thirds of the synonym links were correct using noun lookup in WordNet based on part-of-speech tagging, whereas guessing noun every time resulted in only about half the synonym links being correct. (In this experiment, the part-of-speech tags were correct in 78/87 = 89.65% of the cases involving synonyms.) This shows a substantial positive impact due to part-of-speech tagging. It is also worth noting that of the correct guesses

found, 20.17% (69/342) were simply morphological variants (noun inflections) found by WordNet (e.g., $protest \Leftrightarrow protests$).

However, synonyms by themselves may sometimes be misleading in terms of establishing cohesion. That's because a synonym link between a pair of nouns does not imply that their containing noun phrases are coreferential. For example, for the AP text in Figure A.1, we have $hostage \Rightarrow captive \Leftrightarrow prisoner$; however, in that text, "the remaining captives" refers to people taken hostage by the rebels, whereas, "the prisoners" refers to rebels imprisoned by the government. In the above experiment, only 32 of the correct synonym links (6.27% of the total) were cases where referring NPs containing the nouns were not coreferential. To exclude these would require extremely robust techniques for recognition and resolution of pronominal and definite noun phrase anaphora (i.e., not just proper-name coreference), e.g., as are beginning to be explored in the MUC-6 coreference task [39]. These figures suggest that even if one succeeded in doing so, it would not be a big win in terms of accuracy improvement.

While these accuracy figures are suggestive, more statistically interesting inferences can only be drawn from a much larger-scale experiment. Also, judgments of synonymy can be quite delicate; studies of agreement in judgments across subjects is clearly needed. Finally, the evaluation of a synonym component by itself does not tell us that much about its impact on the overall task of summarization. Clearly, one might expect spurious synonym links to lead the spreading activation search of the graph (to be discussed next) astray, but the size of such a possible effect is unclear.

In the course of working with synonyms, we also explored a more general semantic distance measure between words, based on the relative height of the most specific common ancestor class of the two words, i.e., the most specific common hypernym synset (synonym set) in Wordnet, subject to a context-dependent class-weighting parameter. This approach proved not to give good results, as the technique turned out to be oversensitive to the structure of the thesaurus. The approach of Resnick [47] gets around this by using information content rather than height of the class, where the information content of a synset is related to the probability of the synset and all its subordinates (hyponyms) occurring in a large reference corpus. This approach, when implemented by us, turned out to be very expensive to compute at run-time. Smeaton [53] has addressed this issue, by compiling out, based on Resnick's statistic, a very large table (150,000,000 word pairs) of semantic distances. While Smeaton [53] reports interesting results in image caption retrieval using this table (in combination with a particular query-caption matching metric), the scale of the compilation effort required and the uncertainty of whether it would meet our needs kept us from pursuing it further.

6. Discovering Topic-Related Text Regions

6.1. Finding Topic-Related Text Regions Using Spreading Activation

Given a topic that expresses the user's interest, the refinement phase of processing begins by computing a salience function for text items based on their relation to the topic. A spreading activation algorithm (derived from [12]) is used to find nodes in the graph related to topic nodes.

```
Algorithm Spread(Graph, Topic):
```

```
 Input := words(Topic);
 sort(Input);
 while (Continue?(Output))
   [Node := first(Input);
    insert(Output, Node);
    Succs := ActivateSuccs(Node, Graph);
    while (Succs)
      [insert(Input, pop(Succs));]]
Algorithm ActivateSuccs(Node, Graph):
 while (<Node1, Edge> := edges(Node, Graph))
   [Node1.wt
     = max(Node1.wt, (Node.wt * Edge.wt));
    if (type(Edge) = ADJ)
      [Node1.wt
       = ScaleDist(Node1, Node, Node1.wt).]]
```

The method, which corresponds to a strict best-first search of the graph, begins by adding the nodes matching the given query terms onto an input priority queue, which is kept sorted by decreasing weight[7]. The method then iterates until a terminating condition is reached, taking the first node off the input priority queue and placing it on the output, and then finding successor nodes linked to the current node in the graph and inserting them into the input priority queue. The weight of a successor node is a function of the source node weight and the link type weight. Each different link type has a dampening effect on the source node weight. Since the graph may have cycles, the weight of a successor node is determined by the strongest path to the node. Thus, the successor node weight is the maximum of its new weight and its old weight. ScaleDist returns an exponential function of the text distance between the input nodes. In the termination condition, *Spread* halts if either the number of output nodes is greater than a threshold t_1, or if a slope-based test succeeds.

The slope-based test is as follows. At each iteration of the outer while loop in *Spread*, we maintain the total activation weight of the nodes taken off the input priority queue so far. We compute, for that iteration, the change in activation weight compared to 40 iterations ago, which, divided by the window size of 40, gives us the slope for the change between this iteration and the 40-iterations-previous one. If the standard deviation of the last 40 slopes is less than 0.1, the test succeeds.

The spreading activation is constrained so that the activation decays by link type and text distance. We use the following ordering of different link types, with earlier links in the ordering being heavier (and thus having less dampening effect) than later links:

$$\begin{aligned} &SAME > \ COREFERENCE > \ NAME \\ &> PHRASE \quad > ALPHA > \quad ADJ \end{aligned} \tag{3}$$

For ADJ links, successor node weight is an exponentially decaying function of current node weight and the distance between nodes. Here distances are scaled so that travelling across sentence boundaries is more expensive than travelling within a sentence, but less

than travelling across paragraph boundaries. For the other link types, the successor weight is the product of link weight and current node weight.

As an example, the sentence-level plot of the activation weights for a Reuters article, where the weight at a given sentence position is calculated as the average of its constituent word weights, is shown in Figure 4. The results after spreading, given the topic *Tupac Amaru*, are shown in Figure 5. The spreading has changed the activation weight surface, so that some new related peaks (i.e., local maxima) have emerged (e.g., sentence 4), and old peaks have been reduced (e.g., sentence 2, which had a high tf.idf score, but was not related to *Tupac Amaru*). The exponential decay function is also evident in the neighborhoods of the peaks.

The worst-case algorithmic time complexity of Algorithm *Spread* can be calculated as follows. Assume there are N nodes in the graph and the maximum branching factor of a node is B. The initial sort of the Input priority queue takes at most $Nlog(N)$ time. The test for the termination condition is bounded by some constant k_1. The code in the while loop in *Spread* runs for at most N iterations. In each iteration, we sum the constant cost k_2 of picking the first element and putting it on the output, the cost $log(N)$ of insertion into the priority queue, and the cost of *ActivateSuccs*. The code in the while loop in *ActivateSuccs* runs at most B times; each time the operations are bounded by some constant k_3. *Spread*'s while loop thus takes $(k_1 + k_2 + Bk_3 + logN)$ time. Thus the worst case time complexity of *Spread* is $NlogN + N(k_1 + k_2 + Bk_3 + logN) = O(NlogN)$.

The space complexity is as follows. *Spread* allocates an Input priority queue of size N, where N is the number of nodes in the graph, and an output list. At each step of *Spread*'s while loop, no more than B nodes (where B is the maximum branching factor) are added to Input, with one node being removed. So, the Input requires $B(N-1)$ storage, with the Output requiring no more than the output threshold t_1 storage. The register *Succs* in *ActivateSuccs* reuses B elements of storage at each call. So the total worst case space allocation is $B(N-1) + t_1 + B = O(N)$, assuming, as above, that B is a constant.

6.2. Filtering Activated Regions by Segment Finding

As defined by [51], a text segment is "a contiguous piece of text that is linked internally but largely disconnected from the adjacent text". While the goal of the spreading activation is to reweight the nodes of the graph based on a topic, the goal of the segment finder is to select segments from the reweighted graph. This reduction of the search space is useful in increasing the speed of the system to find similarities and differences and align text segments. The segment finder uses the words of the topic to locate specific nodes in the graph which has first been reweighted by spreading activation. Depending on the parameters given it will define a segment as either all nodes with a weight within a user-defined delta of each peak value (the depth parameter), or it will output all nodes within a user-defined distance in the text from each peak (the width parameter). In the former case, which corresponds to a horizontal clipping of the activation signal, one or more text segments, each of whose words has values within the particular delta of the peak value, will be generated. In the latter case, which corresponds to sampling of the signal, the number of segments is less than (in case the width encompasses a neighboring peak) or equal to the number of peaks. The clipping involves sorting the nodes by weight and then filtering the nodes above a threshold;

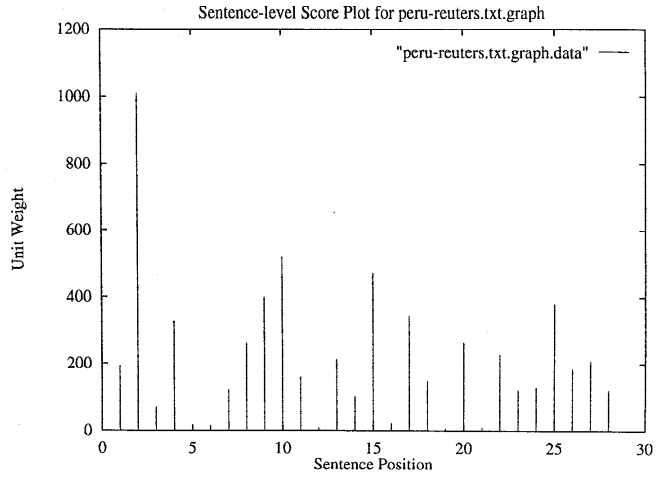

Figure 4. Sentence-level Activation Weights from Raw Graph (Reuters news)

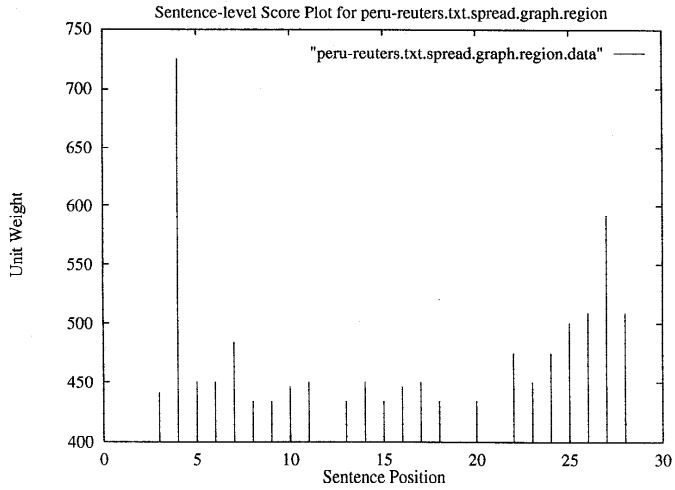

Figure 5. Sentence-level Activation Weights from Spread Graph (Reuters news; topic: *Tupac Amaru*)

the clipping thus has an algorithmic time complexity of order $O(NlogN)$ for a graph of size N nodes, and a space cost of order $O(N)$ (assuming non-destructive sorting).

Figure 6 and Figure 7 shows the results of segment finding on the activation weights in the Reuters and AP articles, using the default depth of 90. The segment-finder has removed 163 word-nodes from the Reuters graph (43% reduction) and 88 words (21% reduction)

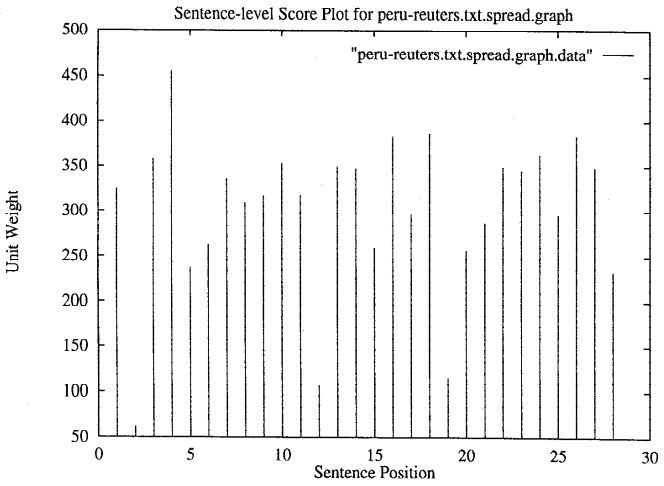

Figure 6. Sentence-level Activation Weights after Segment Finding (Reuters news; topic: *Tupac Amaru*)

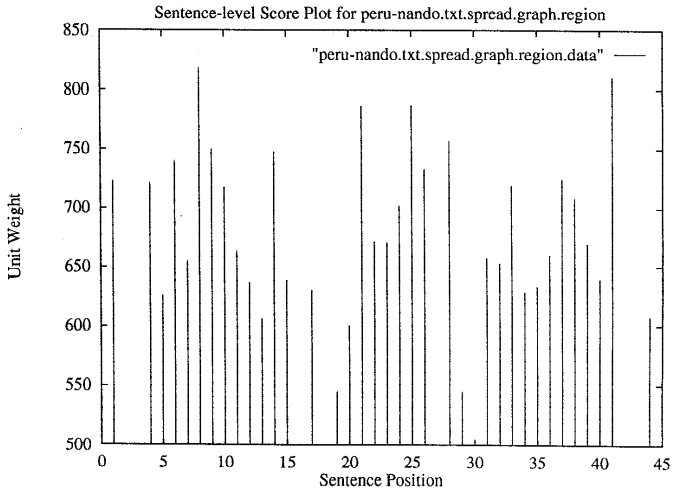

Figure 7. Sentence-level Activation Weights after Segment Finding (AP news; topic: *Tupac Amaru*)

from the AP news article. Note that the amount of reduction varies with the topology of the surface generated by the activation function. Where highly active nodes are uniformly distributed in the text, clipping will result in much less reduction than in cases where there are only a few distinct peaks. The result of this is that some of the sentences are eliminated

(sentences 1, 2, 12, 19 and 21) and the weight of the remaining sentences is increased. The important aspect of this reduction is that although it significantly reduced the number of words being compared, it left nodes with strong associations to *Tupac Amaru* (where the group is mentioned by name, e.g., sentence 4), and also those with less obvious associations (e.g., nodes in sentence 26 - a sentence about a past U.S. collaborator of the MRTA).

It is worth noting that this segment-finding approach differs substantially from previous approaches. For example, in contrast to [51], [24], these text segments are not generated by directly comparing blocks of text (problems with block sizes were discussed earlier). In addition, the segments correspond to potentially variable-sized neighborhoods around the peaks. Finally, unlike approaches such as [24] which use segments to discover topic shifts in text, the segments here are simply used to further restrict the set of salient terms.

6.3. Examples

We will now discuss an example, to illustrate the kinds of links discovered by the spreading activation. Of course, this does not tell us much about aggregate behavior over texts in general. See Section 8 for performance data on arbitrary newswire culled from the World Wide Web, and Section 9 for evaluation. Algorithmic complexity measures for *Spread* and and segment finding have been discussed; corresponding measures for remaining algorithms will also be offered.

Our use of spreading activation allows us to find word occurrences which are indirectly related to the query. Unlike traditional information retrieval approaches [50], [12], however, the final link weights are determined by the number and type of links in the graph. For example, the Reuters sentence 4 plotted in Figure 5 and shown in Figure A.1 might have been found via an information retrieval method which matched on the query *Tupac Amaru* (allowing for *MRTA* as an abbreviated alias for the name). However, it would have not found other information related to the *Tupac Amaru*: In the Reuters article, the spreading method follows an ADJ link from *Tupac Amaru* to *release* in sentence 4, to other instances of *release* via the SAME link, eventually reaching sentence 13 where *release* is ADJ to the name *Victor Polay* (the group's leader). In an Associated Press (AP) article describing the same event, a thesaurus link becomes more useful in establishing a similar connection: it is able to find a direct link from *Tupac Amaru* to *leaders* (via ADJ) in sentence 28, and from there to its synonym *chief* in sentence 29 (via ALPHA), which is ADJ to *Victor Polay*.

Of course, this cohesive relation could also be found more directly if the system could correctly interpret the expressions *its chief* in the AP article and *their leader* in the Reuters article. This raises the question of finding stronger evidence as to how effective the spreading activation is in finding salient topic-related items. In Section 9, we report on experiments which each confirm that the spreading activation is effective in summarization.

7. Summarizing Multiple Documents

7.1. Finding Commonalities and Differences

We now describe our algorithm for finding similarities and differences. Given a set of documents, the goal is to find their relevant shared and distinguishing terms with respect

to a topic. Once text segments are found, only nodes belonging to such segments are considered in building Commonalities and Differences; all other nodes are zeroed out. The set of common words given activated, clipped graphs $G'_1 \ldots G'_n$ is computed by Algorithm Compute-Common:

```
Algorithm Compute-Common(G'_1 ... G'_n):
 for k = 1..n
 [Words[k] = sort-alpha(nodes(G'_k));]
# sort-alpha removes duplicates
# remembering only the best weighted occurrences
# and their weights
# Words[k] =< t_1, w_1 > ... < t_m, w_m >
# where t_i ≺_α t_{i+1}, 1 ≤ i ≤ m
# where m is the number of distinct words in G'_k.
 Row-indices = intersection(Words[1] ... Words[n]);
# Contains terms from intersection, remembering weights
# Row-indices preserves the alphabetic sorting
# from Words
 Column-indices = 1 ... k;
# Common.Words = Row-indices
#Common.Docs = Column-indices;
 Common = build-matrix(Row-indices, Column-indices);
```

Note that Common contains only distinct terms, not term occurrences. Common is represented as a term-document matrix, where the weight of each distinct term in a document is the highest weight of any of its occurrences in that document, normalized by the maximum weight of any term in that document.

Differences, which are computed at the same time as Common, are defined as follows:

$$Differences = (G'_1 ... \cup G'_n) - Common.Words \tag{4}$$

These Differences are differences in query-related information.

Algorithm *Compute-Common* carries out K sorting operations to build $Words$, where K is the number of graphs being compared. If the graph with the most nodes has N nodes, the building of $Words$ costs $O(KNlogN)$. The intersection operation involves intersection of K lists each of length N in the worst case, which costs $K.N^2$. This gives us $O(KN^2logN)$ worst case time complexity for *Compute-Common*. The algorithm uses $O(N)$ space for the sorting and $O(KN)$ space for the Common matrix.

7.2. Presentation Strategies

7.2.1. Overview The kinds of presentation strategies used in the synthesis stage of summarization will vary with the application. However, since very little attention has been paid to this in discussions of multi-document summarization, we illustrate here a range of presentation strategies.

1. A very simple strategy for synthesis of multi-document summaries is to avoid computing commonalities and differences, and instead to simply rank sentences in each document based on weights of contained words, and then to merge the rankings to get multi-document extracts. However, such an approach will not guarantee that higher-ranked sentences in the merged ranking reflect common information among the documents. The presentation strategies we discuss here, therefore, rely on identification of terms in Common and Differences.

2. In *cross-document sentence extraction*, discussed in Section 7.2.2, the best sentences containing words in Common are selected from the set of documents based on the total weight of such words. Likewise, the best sentences containing words in Differences are selected from the set of documents based on the total weight of such words. The two are presented separately, as similarities and differences. When there is a chronological ordering in the set of documents, the differences are presented in terms of what's new in the latest document (with respect to the current topic).

3. In *cross-document sentence alignment*, discussed in Section 7.2.3, pairs of sentences, one from each document (the alignment algorithm is restricted to document pairs), are ranked for coverage of common words.

4. Finally, in Section 7.2.4, we discuss techniques where fragments are extracted instead of sentences. These include "bag-of-terms" presentation strategies, as well as generation of well-formed sentence fragments.

5. Of course, other presentation methods are also possible, e.g., "graphical" displays where we plot documents in a collection so that documents closer together in the plot have more terms in Common. We have not implemented these graphical strategies, but suggest them to indicate the wide space of possible presentation strategies.

7.2.2. Cross-Document Sentence Extraction The presentation strategy used to cover similarities and differences then simply outputs the set of sentences covering the terms in Common and the set of sentences covering the terms in Differences, highlighting the relevant terms in each, and indicating which document the sentence came from. This technique is what we call FSD (for Find Similarities and Differences):

```
Algorithm FSD(Common):
 For each doc k in Common.Docs
 [for each sentence p in doc k
    [score(p,k);]]
```

Sentence selection is based on the coverage of nodes in Common and Differences. Sentences are selected based on the average activated weight of the covered words: The score $score(p, k)$ for sentence p in document k (i.e., sentence s_{pk}) in terms of coverage of Common is:

$$score(p,k) = \frac{1}{|c(p,k)|} \sum_{i=1}^{|c(p,k)|} weight(w_i k) \tag{5}$$

where $c(p,k) = \{w|w\epsilon(Common.Words \cap s_{pk})\}$ and $weight(w_ik)$ is the weight of term i in document k in (the term-document matrix) Common.

The score for Differences is similar. The user may specify the maximal number of non-zero-weighted sentences in a particular category (common or different) to control which sentences are output.

The worst case time complexity of FSD is as follows. Let N be the most number of *distinct words* in any of the input graphs. The cost of the intersection operation to build $c(p,k)$ is bounded by $k_1 logN$, where k_1 is the maximum sentence length, since the $Common.Words$ is sorted alphabetically. The summation in Equation 5 iterates for k_1 times at most, with each iteration having unit cost. The computation of $score(p,k)$ is invoked, in the worst case, for all the sentences in all the documents i.e., KN/k_2 times, where k_2 is the minimum sentence length, and K is the number of graphs. The worst case time complexity of FSD is therefore $K(N/k_2)(k_1 + k_1 logN) = O(KNlogN)$. Holding the sentence scores requires (N/k_2) storage, with $c(s)$ using k_1 storage; this gives us $k_1 + (N/k_2) = O(N)$ storage cost for the algorithm.

It is possible to enhance FSD to ensure that all the commonalities in Common are represented in the summary. This could of course be done by outputting all sentences which contain common words, but this might yield many sentences which each cover the same subset of common words. Instead, it is possible to find smaller subsets of the sentences containing common words, which would reduce the redundancy of information content. We try to find such a subset in the enhanced version, called Algorithm *Greedy-FSD*:

```
Algorithm Greedy-FSD(Common):
while (not-empty(Common.Words))
 [FSD(Common);
  top-s = pop(Sentences);
  Common.Words = rest(Common.Words, top-s);
  output(top-s);]
```

Here, we score all the sentences using Equation 5, then pick the best-scored one, remove the terms covered in Common by that sentence (the $rest$ operator in the algorithm), then rescore all remaining sentences using the new Common, and repeat until Common or the set of remaining sentences is empty.

The while loop in *Greedy-FSD* runs as many times as $|Common.Words| = N$ in the worst case, where N is the most number of *distinct words* in any of the input graphs. The first step is given by the complexity of FSD. The "removal" of the current sentence's words from Common is bounded by k_1, the maximum sentence length, and the popping of the set of sentences has some small constant cost $k4$. So, the worst case time complexity of *Greedy-FSD* is $N(k4+k_1+K(N/k_2)(k_1+k_1 logN)) = O(KN^2 logN)$, where k_2 is the minimum sentence length and K is the number of graphs. The *Greedy-FSD* enhancement is thus relatively expensive compared to Algorithm *FSD* in terms of worst case time complexity, and as a presentation strategy, is useful when maximum compression is desired at the expense of (potentially) increased time. The space cost of *Greedy-FSD* is given by the cost of *FSD* plus the length of the output, which is no longer than $|Common|$, i.e., the space cost is $O(N)$.

To illustrate the behavior of FSD, consider the application of FSD to the extracted segments in Figure 6 (the Reuters article) and the extracted segments in Figure 7 (an AP article of

the same date describing the same hostage crisis). The extracted segments had 42 words in Common, out of 180 words for the first article's segments and 326 for the latter article's segments. The algorithm extracts 24 commonalities, with the commonalities with the strongest associations being on top. Among the high scoring commonalities and differences are the ones shown in Figure A.1, where the words in Common are in bold face. The algorithm discovers that both articles talk about *Victor Polay* (e.g., the Reuters sentence 13 mentioned earlier, and the AP sentence 29 shown in Figure A.1). A similar point could be made about the *Fujimori* sentences. Notice that the system is able to extract commonalities without *Tupac Amaru* being directly present. Regarding differences, the algorithm discovers that the AP article is the only one to explain how the rebels posed as waiters (sentence 12) and the Reuters article is the only one which told how the rebels once had public sympathy (sentence 27).

7.2.3. Cross-Document Sentence Alignment In aligning news stories, we directly compare text units from one text with text units from the other. Here, we have in general a choice between aligning segments or sentences, in the former case outputting sentences by completing (say) to the nearest sentence boundary. In this method, rather than comparing one unit and outputting another, we chose to consistently align sentences, where the weights of words which are not part of a text segment are zeroed out.

```
Algorithm Align(Common):
For each sentence p in Common.doc1
[for each sentence q in Common.doc2
 [score = score-overlap(p, q);
  if (best-match-row[p] < score)
   [best-match-row[p] = q;]
 if (best-match-col[q] < score)
   [best-match-col[q] = p;]]]
For each sentence p in Common.doc1
[q = best-match-row[p];
 if (best-match-col[q] = p)
  [then output(<p, q>);]]
```

The algorithm ranks pairs of sentences, one member from each document, for coverage of common words. First, as before, once a pair of graphs has been spread and clipped, terms in Common are computed. Only sentences containing terms in Common are considered. The basic one-to-one algorithm matches pairs of sentences based on their degree of overlap, where the overlap between a sentence pair is the total activation weight of terms common to both. Thus, given a pair of sentences s_1 and s_2, s_1 is scored for overlap with s_2 using Equation 5 with $S = Common \cap s_1 \cap s_2$ being used instead of $Common$. Once all the pairs are scored for overlap, the algorithm imposes a "symmetry check", picking the sentence pairs $< s_i, s_j >$ such that s_i's best overlapping alignment is with s_j, and s_j's best overlapping alignment is with s_i.

This overlap measure is somewhat insensitive to relative differences in weights, making it somewhat less precise than one that is more sensitive to the relative weights, such as

cosine similarity [48]. Section 9.2 explores the use of cosine similarity further, offering some experimental results using cosine similarity as the sentence matching metric. Our experience indicates a tradeoff exists between higher recall of common terms (emphasized by the word overlap measure) and higher precision (emphasized in this case by cosine similarity).

The worst case time complexity of this algorithm is as follows. Let S_1 be the number of sentences in the first document, and S_2 be the number of sentences in the other. To perform the one-to-one sentence alignment of the two documents, in the worst case the algorithm considers $S_1.S_2$ pairs. Each pair is measured for the degree of overlap by computing an intersection costing no more than k_1, where k_1 = the maximum sentence length. The symmetry check takes $min(S_1, S_2)$ time. So, the worst case time complexity of the algorithm is $S_1 S_2 k_1 + min(S_1, S_2)$. Now, $S_1 = N_1/k_2$ and $S_2 = N_2/k_2$, where N_1 is the number of distinct words in the first graph, N_2 is the number of distinct words in the second graph, and k_2 is the minimum sentence length. Thus we have the worst case time complexity as $((N_1/k_2)(N_2/k_2)k_1) + min(N_1/k_2, N_2/k_2)$. Letting $N = max(N_1, N_2)$, this gives $O(N^2)$ time complexity. The space complexity is given by the space required to store each maximum value for each row and column in a matrix of size $S_1.S_2$, which requires $S_1 + S_2 = O(N)$ storage.

Given *Align*'s quadratic worst case time complexity on document pairs, it is not particularly scalable, and it becomes even more computationally expensive to extend it to align sentences for every pair of documents in a set. Further, it is hard to for the user to interpret alignments between more than one pair at a time. Therefore, we have restricted its use to a document pair at a time. However, our experience shows that the algorithm is surprisingly useful in various applications, e.g., on static collections of news articles about related events. It is able to discover both obvious cases where the two articles use very similar sentences to describe a common event, as well as, in a large number of cases, ones where the sentences are rather different.

Figure A.1 shows the top two one-to-one alignments from the AP-Reuters pair. Here the alignments are shown boxed, with overlap terms in bold font. The first alignment's sentences, which do not mention the topic *Tupac Amaru*, are near the top in both documents. The algorithm often aligns initial texts, because the initial texts often use similar terms to encapsulate a story. A comparison of alignment methods is described in Section 9.2.

7.2.4. Extracting Fragments instead of Sentences In the above presentation strategies, the presentation unit is the sentence. However, while a sentence may have a certain degree of coverage of terms in Common (or Differences) (and therefore of terms related to the topic by cohesion relations) there will be other words in the presented sentence which aren't related to the topic. Some of these words may be function words, but others may not. Presenting units smaller than a sentence is thus often useful. One "bag-of-terms" presentation strategy is to present lists of words, phrases, and proper names relevant to the multi-document summary. These can be straightforward presentations of terms in Common and Differences, or they can be presentations of sentences extracted by other presentation modes . For example, terms in Common in the documents can be highlighted, much as salient terms are highlighted in [43]. There are also certain applications which call for well-formed, more connected extracts, but where, given a particular target compression

Topic: arrest
Doc1: (AP) **Man Held** by N. **Korea** Found Dead; 12/19/96
Doc2: (New York Times) **Man Held** as Spy in North **Korea** Is a Suicide; 12/19/96
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
(13)... Hunziker had said he entered communist **North Korea** from China "out of curiosity and to **preach** the **Gospel** ." ...
(17)... American said he was in **North Korea** because he wanted to **preach** the **Gospel** . ...
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
(27)... Evan refused to **talk** about his **time** in **North Korea** ...
(46)... interviews, preferring not to **talk** about his **time** in **North Korea** . ...
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
(20)... **Hunziker** had four **outstanding arrest warrants** for failing to comply with earlier court orders, such as getting evaluated for **alcohol** and **drug** abuse and attending ...
(7)... pursuit of him on three **outstanding arrest warrants** , or personal demons of drugs and **alcohol** . ...
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
(3)... (AP) **Evan C. Hunziker** would never say much about the three **months** he was **held** in **North Korea** as a **spy** suspect before diplomatic negotiations won his **release** just in time for **Thanksgiving** . ...
(4)... **month** after basking in the **Thanksgiving** Eve glow of **release** from **North Korea** , where he was **held** as a **spy** and threatened with execution, **Evan Hunziker** was found Wednesday ...
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
(5)... office ruled the **death** a **suicide** . ...
(53)... police said they considered the **death** a **suicide** . ...

Figure 8. Alignment Examples using Fragment Extraction

rate, we would like to pack the available space more efficiently. We will now illustrate its use in multi-document alignment, although it is applicable to FSD and other presentation modes.

Once we have aligned pairs of sentences, the synthesis component can choose to extract fragments rather than full-sentences. Given a (soft) reduction factor (e.g., 25% of full-text), the system picks off terms from Common from the reweighted positional term vector, and then extracts a context window around each selected term occurrence. The context window extraction is based on taking a minimal window (2-3 words) around each selected term occurrence (by itself not likely to yield well-formed fragments) and extending the context towards a boundary. The tests for a boundary uses patterns involving part-of-speech and punctuation features. Overlapping contexts are then merged by merging their context windows. The patterns are rather trivial at this point (e.g., extend to the right until the last noun in a noun group, or punctuation characters) but with more data analysis we expect to come up with a more definitive set of such patterns. This "middle-out" method allows for fine-grained control in summary output, cuts out potentially irrelevant sentence material and therefore packs the summary rather better with salient terms. This becomes useful in fitting a larger number of alignments (each of which can be rendered in fewer words than two full sentences) within the length limits required by the reduction factor for the summaries. Figure 8 illustrates this. The sentence number from which each fragment is drawn is shown alongside the fragment; the pairs are presented in decreasing order of degree of overlap. Terms in Common are boldfaced. This particular summary shows a reduction of 28% over the corresponding full-sentence summary.

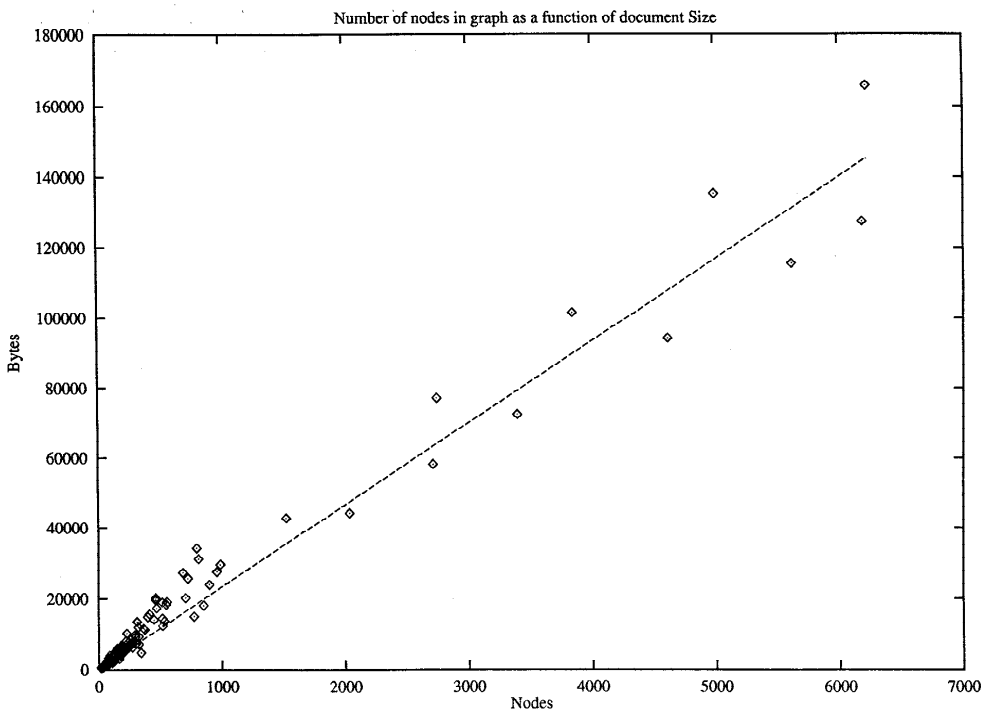

Figure 9. Number of nodes in graph as function of document size

8. Performance

We now measure the timing performance of the algorithms on Internet news sources. All results are cpu time on a sparc10 with a 55 MHz clock speed. In Figure 9, we show the number of nodes in the graph as a function of document size. Figures 10-13 show the times to compute the graph, spread, segment-finding using clipping, and total time. The times are plotted against document size. As can be seen, the time performance of these algorithms appears to be approximately linear in document size.

Next, we measure the timing performance of the algorithm to find common nodes between two graphs, described earlier in Section 7.1. This is shown in Figure 14. As can be seen from these figures, the time to find common nodes is approximately linear in the size of the documents. Finally, in Figures 15,16 and 17, we show the performance of *FSD*, *Greedy-FSD*, and *Align*, respectively.

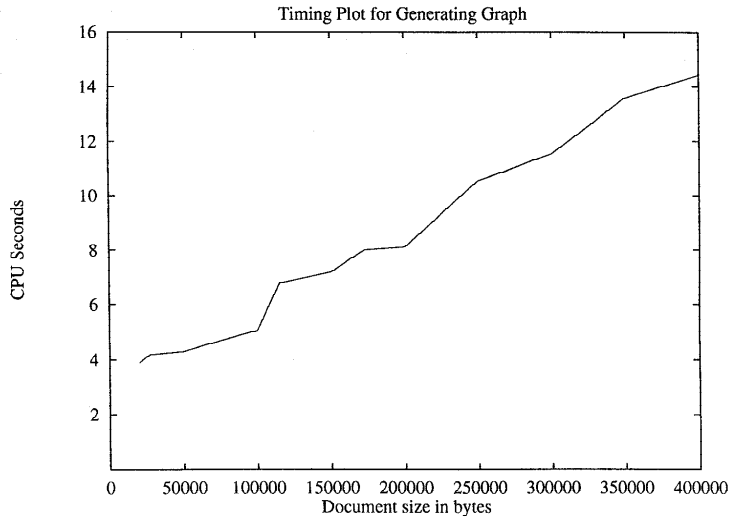

Figure 10. Time to build graph as a function of document size

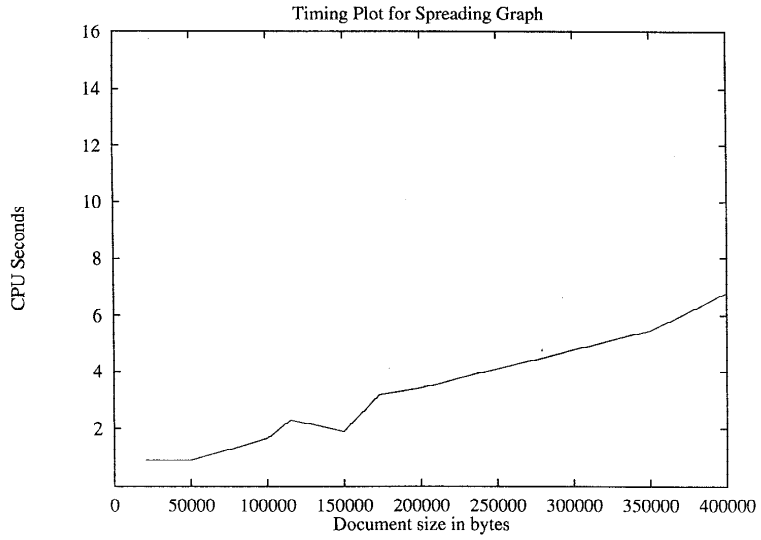

Figure 11. Spread Time as a function of document size

9. Evaluation

9.1. Overview

Text summarization is still an emerging field, and serious questions remain concerning the appropriate methods and types of evaluation. There is little consensus as to what basis

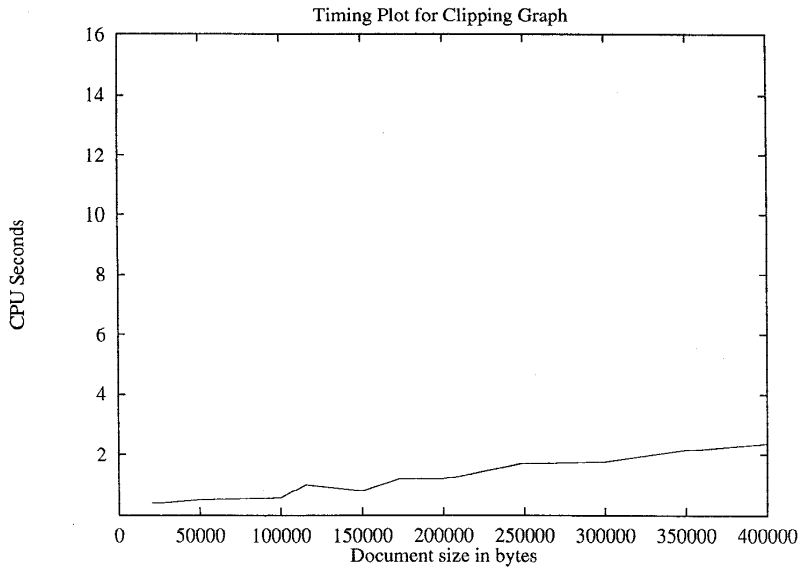

Figure 12. Clip Time as a function of document size

is best for comparison, e.g., summary to source, machine to human-generated, system to system. In comparing against human summaries, reports of low inter-annotator agreement over what should be included in a summary (e.g., [45], [36]) raise questions about the appropriateness of a "gold standard" for sentence extraction.

In general, methods for evaluating text summarization approaches can broadly classified into two categories. The first is an extrinsic evaluation in which the quality of the summary is judged based on how it affects the completion of some other task. The second approach, an intrinsic evaluation, judges the quality of the summarization directly based on user judgements of informativeness, coverage, etc. In our evaluation we performed both type of experiments. Evaluation experiments based on the intrinsic method are discussed in Section 9.2.

We believe the objective evaluation measures we introduce in Section 9.3 represents a significant step forward in terms of empirically demonstrating the utility of summarization in a practical information retrieval task. This method has since been adopted as a standard method for summarization evaluation in the U.S. government's TIPSTER program [22]. However, it is important to stress that our evaluations, while obtaining statistically significant results, are on small datasets.

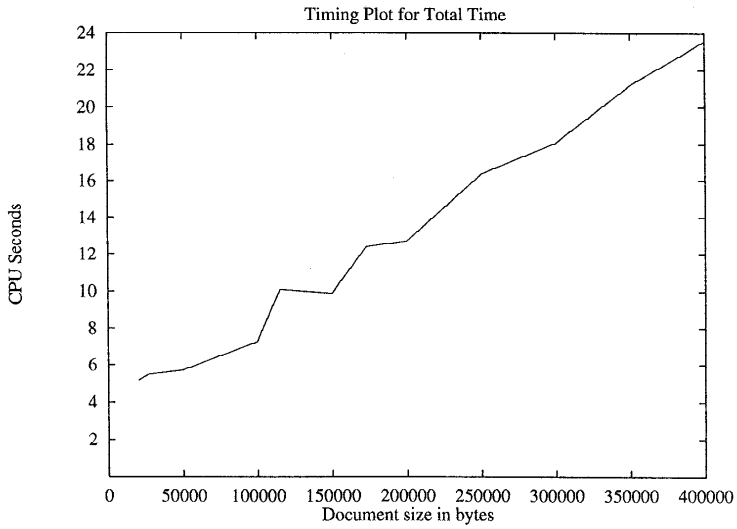

Figure 13. Total time (graph+spread+clip) as a function of document size

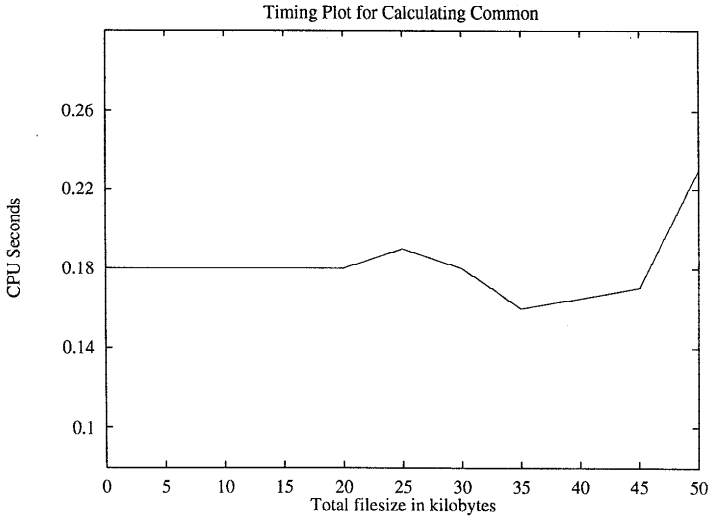

Figure 14. Time to find common nodes as a function of combined document pair size

9.2. Comparison of Weighting Methods in Cross-Document Alignment

In this experiment, six different schemes for reweighting words within the sentence were compared: 1) tf.idf (RAW), 2) tf.idf with weights increased for proper names by a constant factor (RAWPOL), 3) spreading

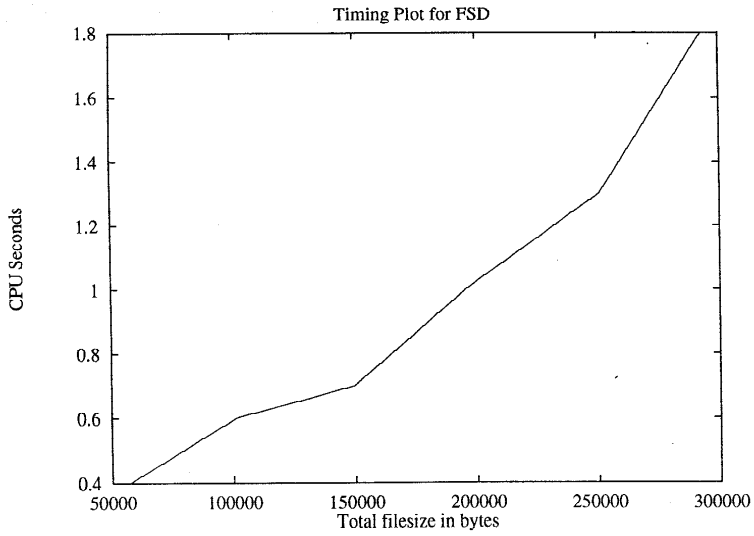

Figure 15. Time for FSD as function of combined document pair size

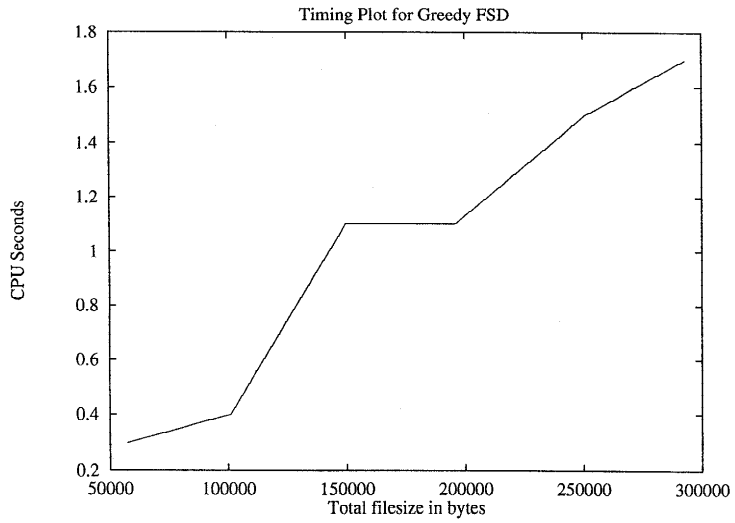

Figure 16. Time for Greedy-FSD as function of combined document pair size

(SPREAD), 4) raw tf.idf after removal of low-weight terms (RAW-CLIP), 5) clipping after RAWPOL (RAWPOL-CLIP) and 6) clipping after spreading

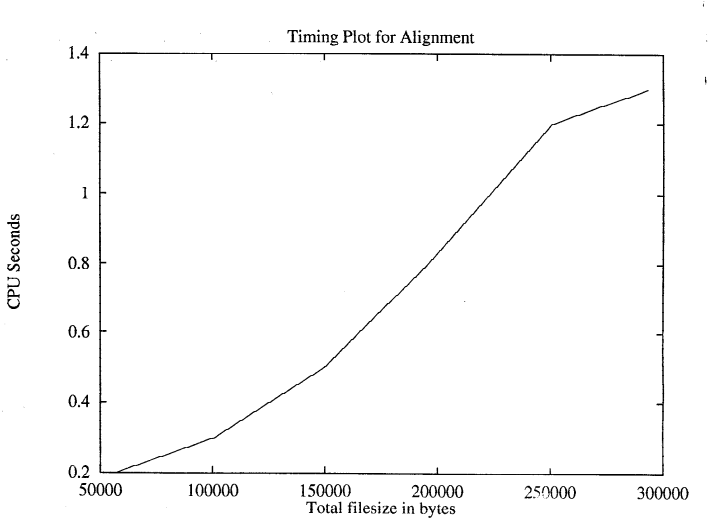

Figure 17. Time for Align as function of combined document pair size

Name	Source	Headline
Peru	AP	Rebels in Peru hold hundreds of hostages inside Japanese diplomatic residence
	Reuters	Peru rebels hold 200 in Japanese ambassador's home
Evangelist	New York Times	Man Once Held As Spy In North Korea Is A Suicide
	Washington Post	Man Held By N. Korea Found Dead
Chechnya	Washington Post	Gunmen Kill Aid Workers In Chechnya
	New York Times	6 Red Cross Aides Slain in Chechnya, Imperiling Peace

Table 2. Information about Sources Being Aligned

Article Pair	RAW	RAWPOL	SPREAD	RAW-CLIP	RAWPOL-CLIP	SPREAD-CLIP
Peru	10	25	11	37	20	44
Evangelist	15	27	24	21	20	27
Chechnya	15	13	14	14	17	12
Average	13.3	21.7	16.3	24	19	**27.7**

Table 3. Alignment Comparison Results

(SPREAD-CLIP). In all these schemes, we used cosine similarity instead of the overlap measure, as it allows for more standard baselines. Three different document pairs were used here for evaluation, as shown in Table 2. These pairs were selected from a larger

Table 4. Summaries versus Full-Text: Task Accuracy, Time, and User Feedback

Metric	Full-Text	Summary
Accuracy (Precision, Recall)	30.25, 41.25	25.75, 48.75
Time (mins)	**24.65**	**21.65**
Usefulness of text in deciding relevance (0 to 1)	.7	.8
Usefulness of text in deciding irrelevance (0 to 1)	.7	.6
Preference for more or less text	"Too Much Text."	"Just Right."

collection of pairs of articles on international events culled from searches on the World Wide Web, including articles from Reuters, Associated Press, the Washington Post, and the New York Times. Pairs were selected such that each member of a pair was closely related to the other, but by no means identical; the pairs were drawn from different geopolitical regions so that no pair was similar to another. In the Peru pair only the precision of the top ten sentence pairs is calculated. For the other pairs precision is calculated for all output sentence pairs (on average 50 sentence pairs for Evangelist and 60 for Chechnya). For each document pair the assigned weighting method was applied to each text and the single best match for each sentence was output. The goal of this experiment was to measure the ability of the alignment method to find correct alignments (those that are both correctly aligned and relevant to the user's given topic). Alignment correctness was determined by a human judge.

In Table 3, we see that all of the reweighting schemes outperform the baseline tf.idf measure for these tasks and that the highest average results are obtained with the method which uses spreading and clipping. The results with spreading alone (SPREAD) were also better on average than tf.idf (RAW) with the greatest difference on the Evangelist pair, but small differences on the other pairs. The removal of words using clipping resulting in improvements (on average) for the RAW and SPREAD based methods, but not for the RAWPOL. Clipping results in the most reduction when the differences between minimum and maximum word weights is greatest. This suggests that the proper name weight increment in RAWPOL may have been too large, causing more words, and sometimes useful words, to be removed. These results are only suggestive; conclusive results would require experimenting with a much larger data sample.

9.3. Effectiveness of Spreading Activation

In addition to the intrinsic evaluation of alignments, we also carried out an extrinsic evaluation, where we evaluated the usefulness of spreading in the context of an information retrieval task. In this experiment, subjects were informed only that they were involved in a timed information retrieval research experiment. In each run, a subject was presented with a pair of query and document, and asked to determine whether the document was relevant or irrelevant to the query. In one experimental condition the document shown was the full text, in the other the document shown was a summary generated with the top 5 weighted sentences. Subjects (four altogether) were rotated across experimental conditions, but no subject was in both conditions for the same query-document pair. We hypothesized that if the summarization was useful, it would result in savings in time, without significant loss in accuracy.

Four queries (204, 207, 210, and 215) were preselected from the TREC [23] collection of topics, with the idea of exploiting their associated (binary) relevance judgments. A subset of the TREC collection of documents was indexed using the SMART retrieval system from Cornell [11]. Using SMART, the top 75 hits from each query were reserved for the experiment. Overall, each subject was presented with four batches of 75 query-document pairs (i.e., 300 documents were presented to each subject), with a questionnaire after each batch. Accuracy metrics include precision (percentage of retrieved documents that are relevant, i.e., number retrieved which were relevant/total number retrieved) and recall (percentage of relevant documents that are retrieved, i.e., number retrieved which were relevant/total number known to be relevant).

In Table 4, we show the average precision and average recall over all queries (1200 relevance decisions altogether). The table shows that when the summaries were used, the performance was faster than with full-text ($F=32.36$, $p < 0.05$, using analysis of variance F-test), without significant loss of accuracy. While we would expect shorter texts to take less time to read, it is striking that these extracts are effective enough to support accurate retrieval. In addition, the subjects' feedback from the questionnaire (shown in the last three rows of the table) indicate that the spreading-based summaries were found to be useful.

10. Conclusion

This summarization approach exploits the results of recent progress in information extraction to represent salient units of text and their relationships. By exploiting meaningful *relations* between units based on text cohesion and the *perspective* from which the comparison is desired, the summarizer can pinpoint similarities and differences and align text extracts across articles. In evaluations, these techniques for exploiting cohesion relations result in summaries which helped users more quickly complete a retrieval task, which resulted in improved alignment accuracy, and which improved identification of topic-relevant similarities and differences. Our approach is highly domain-independent, even though we have illustrated its power mainly for news articles. However, despite these encouraging outcomes, we are also painfully aware that the field of summarization has still a long way to go, and that these methods only touch the surface of the problem. It is our hope that this paper will spur discussion and future work in this area. In the future, we expect to investigate incorporation of co-occurrence statistics, e.g., [14], [17], [19], [20], [52], and also to further investigate temporal sequences of stories, to summarize changes over time.

Acknowledgments

We would like to thank Barbara Gates for helping generate performance data, and David House and Gary Klein for help with the evaluation experiments. We are also grateful to two anonymous referees for their helpful comments.

Appendix

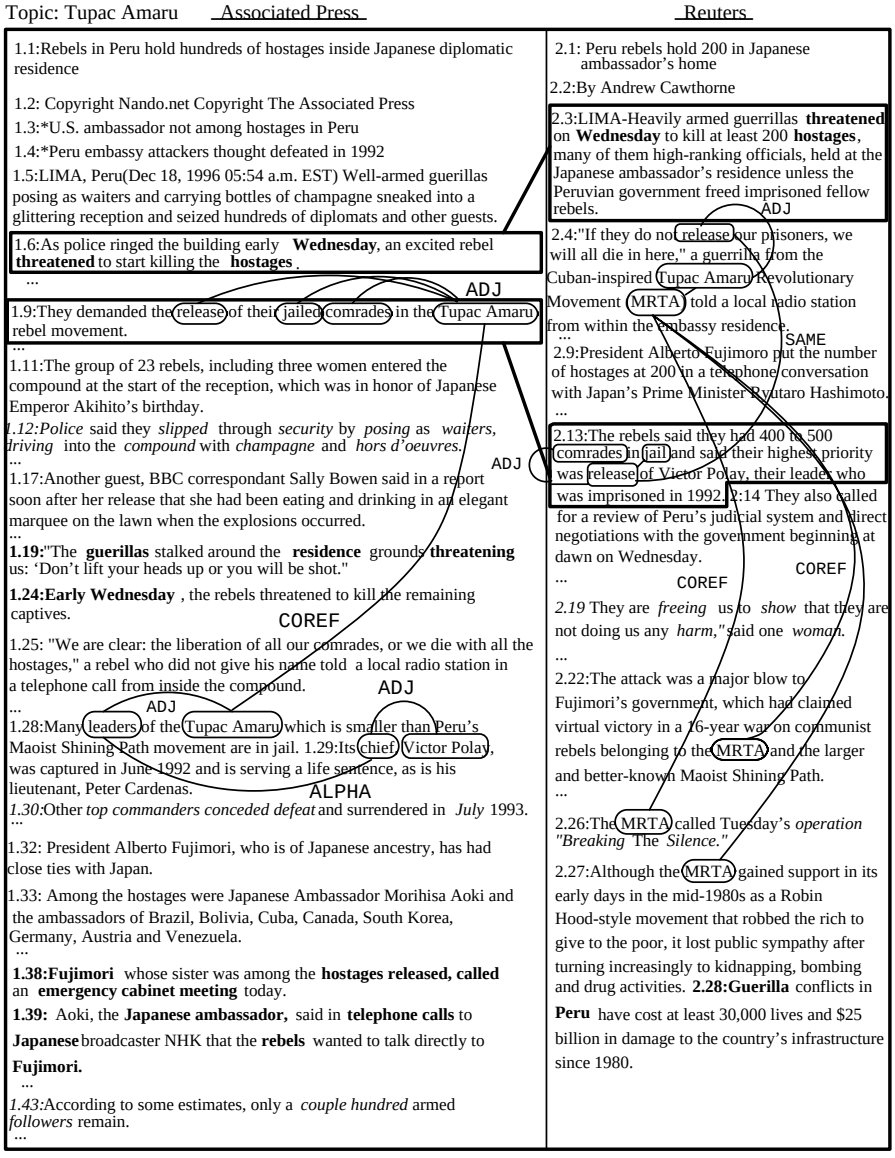

Figure A.1. Texts of two related articles. The top 5 salient sentences containing words in Common have these common words in bold face; likewise, the top 5 salient sentences containing words in Differences have these words in italics. Alignments are shown boxed.

Notes

1. There is some degree of consensus on this, though it is not entirely standard. [55] characterizes summarization in terms of a three-phase model, but chooses the term 'transformation' rather than 'refinement'. [33] assumes a four-phase model, where what we are calling 'refinement' is split into 'selection' and 'condensation'.
2. As [28] shows, in applying the TextTiling work of [24] to closed-captioned news broadcasts, it is hard to make do with a single block size; the block size must be small enough to catch relatively small topics, and yet large enough for the similarity metric to be useful.
3. In general, the relations grouped under 'text cohesion' as used by [21] include linguistic devices such as anaphora, ellipsis, conjunction and lexical relations such as reiteration, synonymy, hypernymy, and conjunction.
4. Repetition links are not evidenced in the example.
5. In terms of accuracy, when trained on about 950,000 words of Wall Street Journal text, the tagger obtained 96% accuracy on a separate test set of 150,000 words of WSJ [1].
6. Here the symbol $\Rightarrow$ stands for a hypernym link, $\Leftrightarrow$ for a synonym link.
7. The matching uses stemming based on [44].

References

1. J. Aberdeen, J. Burger, D. Day, L. Hirschman, P. Robinson, and M. Vilain. "MITRE: Description of the Alembic System Used for MUC- 6", Proceedings of the Sixth Message Understanding Conference (MUC-6), Columbia, Maryland, November 1995.
2. J. Abracos and G. Pereira Lopes. Statistical Methods for Retrieving Most Significant Paragraphs in Newspaper Articles, in Mani, I., and Maybury, M., eds., Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, 11 July 1997, pp. 51-57.
3. R. Alterman. "A Dictionary Based on Concept Coherence", Artificial Intelligence, 25, 1985, pp. 153-186.
4. C. Aone, M.E. Okurowski, J. Gorlinsky and B. Larsen. "A Scalable Summarization System using Robust NLP", in Mani, I., and Maybury, M., eds., Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, 11 July 1997, pp. 66-73.
5. J.P. Callan. "Passage-Level Evidence in Document Retrieval", Proceedings of SIGIR'94, p. 302-310, 1994.
6. A. Barzilay and M. Elhadad. "Using Lexical Chains for Text Summarization", in Mani, I., and Maybury, M., eds., Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, 11 July 1997, pp. 10-17.
7. P.B. Baxendale. "Man-made index for technical literature: an experiment", IBM Journal of Research and Development, 2, 4, 1958, pp. 354-361.
8. B. Boguraev and C. Kennedy. "Salience-based Content Characterization of Text Documents", in Mani, I., and Maybury, M., eds., Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, 11 July 1997, pp. 2-9.
9. E. Brill. "Some advances in rule-based part-of-speech tagging", Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, August 1-4, 1994, pp. 722-727.
10. J. Broglio and B. Croft. "Query Processing for Retrieval from Large Text Bases", ARPA Human Language Technology Workshop, 1993.
11. B. Buckley. "The Importance of Proper Weighting Methods", ARPA Human Language Technology Workshop, 1993.
12. C.H. Chen, K. Basu and T. Ng. "An Algorithmic Approach to Concept Exploration in a Large Knowledge Network", Technical Report, MIS Department, University of Arizona, Tucson, AZ, 1994.
13. J.D. Cohen. "Hilights: Language- and Domain-Independent Automatic Indexing Terms for Abstracting", Journal of the American Society for Information Science, 46, 3, 162-174, 1995. See also vol. 47, 3, 260 for a very important erratum.
14. S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. "Indexing by Latent Semantic Analysis", Journal of the American Society for Information Science, 41, 6, pp. 391-407.
15. H.P. Edmundson. "New methods in automatic abstracting", Journal of the Association for Computing Machinery, 1969, 16, 2, pp. 264-285.
16. D. Evans. "The Clarit Project", Technical Report, Laboratory for Computational Linguistics, Carnegie Mellon University, 1991.

17. D.A. Evans, K. Ginther-Webster, M. Hart, R.G. Lefferts and I.A. Monarch. “Automatic indexing using selective NLP and first-order thesauri”, Proceedings of RIAO’91, 2, pp. 624-643.
18. D. Evans and C. Zhai. “Noun Phrase Analysis in Unrestricted Text for Information Retrieval”, Proceedings of ACL-96, Cambridge, MA, June 1996.
19. G. Grefenstette. “Use of syntactic context to produce term association lists for text retrieval”, Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1992, pp. 89-97.
20. G. Grefenstette. “Explorations in Automatic Thesaurus Discovery”, Kluwer, Boston, 1994.
21. M. Halliday and R. Hasan. “Cohesion in Text”, 1996, London, Longmans.
22. T.F. Hand. “A Proposal for Task-Based Evaluation of Text Summarization Systems”, in Mani, I., and Maybury, M., eds., Proceedings of the ACL/EACL’97 Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, 11 July 1997.
23. D. Harman, editor, “An Overview of the Third Text Retrieval Conference”, National Institute of Standards and Tehnology, NIST Special Publication 500-225, 1994, Gaithersburg, MD.
24. M. Hearst. “Multi-Paragraph Segmentation of Expository Text”, Proceedings of ACL-94, Las Cruces, New Mexico, 1994.
25. G. Krupka. “SRA: Description of the SRA System as Used for MUC-6”, Proceedings of the Sixth Message Understanding Conference (MUC-6), Columbia, Maryland, November 1995.
26. J. Kupiec, J. Pedersen and F. Chen. “A Trainable Document Summarizer”, Proceedings of ACM-SIGIR’95, Seattle, WA, 1995, pp. 68-73.
27. E.R. Liddy. “The discourse-level Structure of Empirical Abstracts: An Exploratory Study”, Information Processing and Management, 1991, 27, 1, 55-81.
28. I. Mani, D. House, M. Maybury and M. Green. “Towards Content-Based Browsing of Broadcast News Video”, in Maybury, M., ed., Intelligent Multimedia Information Retrieval, AAAI/MIT Press, 1997.
29. I. Mani and E. Bloedorn. “Summarizing Similarities and Differences Among Related Documents”, Proceedings of RIAO-97, Montreal, Canada, June 25-27, 1997, pp. 373-387.
30. I. Mani and E. Bloedorn. “Multi-document Summarization by Graph Search and Merging”, Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), Providence, RI, July 27-31, 1997, pp. 622-628.
31. W.C. Mann and S.A. Thompson. Rhetorical Structure Theory: Toward a functional theory of text organization. Text, 8, 3, 1988, pp. 243-281.
32. D. Marcu. “From discourse structures to text summaries”, in Mani, I., and Maybury, M., eds., Proceedings of the ACL/EACL’97 Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, 11 July 1997, pp. 82-88.
33. M. Maybury. “Generating Summaries from Event Data”, Information Processing and Management, 31, 5, 1995, pp. 735-751.
34. K. McKeown and D. Radev. “Generating Summaries of Multiple News Articles”, Proceedings of ACM-SIGIR’95, Seattle, WA.
35. S. Miike, E. Itoh, K. Ono and K. Sumita. “A Full-Text Retrieval System with a Dynamic Abstract Generation Function”, Proceedings of ACM-SIGIR’94, Dublin, Ireland.
36. M. Mitra, A. Singhal and C. Buckley. “Automatic Text Summarization by Paragraph Extraction”, in Mani, I., and Maybury, M., eds., Proceedings of the ACL/EACL’97 Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, 11 July 1997.
37. J. Morris and G. Hirst. “Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text”, Computational Linguistics, 17, 1, pp. 21-43, 1991.
38. G. Miller. “WordNet: A Lexical Database for English”, Communications of the ACM, 38, 11, pp. 39-41, 1995.
39. MUC-6, Proceedings of the Sixth Message Understanding Conference (MUC-6), Columbia, Maryland, November 1995.
40. C. Paice. “Constructing Literature Abstracts by Computer: Techniques and Prospects, Information Processing and Management, 26, 1, pp. 171-186, 1990.
41. C. Paice and P. Jones. “The Identification of Important Concepts in Highly Structured Technical Papers”, Proceedings of ACM-SIGIR’93, Pittsburgh, PA.
42. W. Paik, E. Liddy, E. Yu and M. McKenna. “Categorizing and Standardizing Proper Nouns for Efficient Information Retrieval”, Proceedings of the ACL Workshop on Acquisition of Lexical Knowledge from Text, Ohio State University, 1993.

43. C. Pearce and C. Nicholas. “TELLTALE: Experiments in a dynamic hypertext environment for degraded and multilingual data”, JASIS, 47, 4, 263-275, 1996.
44. M.F. Porter. “An Algorithm For Suffix Stripping”, Program, 14, 3, July 1980, pp. 130-137.
45. G.J. Rath, A. Resnick and T.R. Savage. “The formation of abstracts by the selection of sentences”, American DOcumentation, 12, 2, 1961, pp. 139-143.
46. L. Rau. ”Knowledge Organization and Access in a Conceptual Information System,” Information Processing and Management, 23, 4, 269-283, 1987.
47. P. Resnick. “Selection and Information: A Class-Based Approach to Lexical Relationships”, Ph.D. Dissertation, 1993, University of Pennsylvania, Philadelphia, PA.
48. G. Salton. “Automatic text processing - the transformation, analysis, and retrieval of information by computer”, Addison-Wesley, Reading, MA, 1989.
49. G. Salton, J. Allan, C. Buckley and A. Singhal. “Automatic Analysis, Theme Generation, and Summarization of Machine-Readable Texts”, Science, 264, June 1994, pp. 1421-1426.
50. G. Salton and C. Buckley. “On the Use of Spreading Activation Methods in Automatic Information Retrieval”, Technical Report 88-907, Department of Computer Science, Cornell University, 1988.
51. G. Salton, A. Singhal, C. Buckley and M. Mitra. “Automatic Text Decomposition Using Text Segments and Text Themes”, Cornell University Technical Report TR 95-1555, Nov. 17, 1995.
52. H.M. Schutze and J.O. Pedersen. “A Cooccurrence-Based Thesaurus and Two Applications to Information Retrieval”, Proceedings of RIAO’97.
53. A.F. Smeaton and I. Quigley. “Experiments on Using Semantic Distances Between Words in Image Caption Retrieval”, Proceedings of ACM-SIGIR’96, Zurich, Switzerland.
54. K. Sparck-Jones. “A Statistical Interpretation of Term Specificity and Its Application in Retrieval”, Journal of Documentation, 28, 1, 11-20, 1972.
55. K. Sparck-Jones. “Summarizing: Where are we now? Where should we go?”, in Mani, I., and Maybury, M., eds., Proceedings of the ACL/EACL’97 Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, 11 July 1997.
56. T. Strzalkowski. “Natural Language Information Retrieval: TIPSTER-2 Final Report”, TIPSTER Text Program (Phase II), 1996, pp. 143-148.
57. T.A. Van Dijk. “News as Discourse”, Lawrence Erlbaum, Hillsdale, NJ, 1988.
58. E.M. Voorhees. “Using WordNet to Disambiguate Word Senses for Text Retrieval”, Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, PA, June, 1993, pp. 171-180.