Multi-document Summarization by Visualizing Topical Content

Rie Kubota Ando
Department of Computer Science, Cornell University, Ithaca, NY 14853-7501
kubotar@cs.cornell.edu

Branimir K. Boguraev, Roy J. Byrd, Mary S. Neff
IBM T.J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532
{bkb, byrd, neff}@watson.ibm.com

Abstract

This paper describes a framework for multi-document summarization which combines three premises: coherent themes can be identified reliably; highly representative themes, running across subsets of the document collection, can function as multi-document summary surrogates; and effective end-use of such themes should be facilitated by a visualization environment which clarifies the relationship between themes and documents. We present algorithms that formalize our framework, describe an implementation, and demonstrate a prototype system and interface.

1 Introduction: multi-document summarization as an enabling technology for IR

The rapid growth of electronic documents has created a great demand for a navigation tool to traverse a large corpus. Information retrieval (IR) technologies allow us to access the documents presumably matching our interests. However, a traditional hit list-based architecture, which returns linearly organized single document summaries, no longer suffices, given the size of a typical hit list (e.g. submitting the query "summarization workshop" to a search engine Altavista (http://altavista.com) gave us more than ten million hits).

To allow a more comprehensive and screen space-efficient presentation of query results, we propose in this paper a technology for summarizing collections of multiple documents. In our work, we focus on identifying themes, representative of a document, and possibly running across documents. Even if we are unable to 'embody' a theme in coherently generated prose, we start with the assumption that a mapping exists between a theme and a tightly connected (and therefore intuitively interpretable) set of coherent linguistic objects, which would act as a 'prompting' device when presented to the user in an appropriate context. As will become clear in the rest of the paper, we refer to such themes as *topics*.

Our view of multi-document summarization combines three premises: coherent topics can be identified reliably; highly representative topics, running across subsets of the document collection, can function as multi-document summary surrogates; and effective end-use of such topics should be facilitated by a visualization environment which clarifies the relationship between topics and documents. The work specifically addresses the following considerations.

- **Multiple general topics** We regard the ability to respond to multiple topics in a document collection — in contrast to a prevailing trend in multi-document summarization, seeking to present the single, possibly pre-determined, topic (see below) — to be crucial to applications such as summarization of query results. In this work we choose not to narrow the topic detection process by the given query, since in IR it is a well-known concern that user-specified queries do not necessarily convey the user's real interests thoroughly. Thus, we need to deal with multiple general topics.

- **Textual and graphical presentation** Since our multi-document summaries will, by definition, incorporate multiple topics, the question arises of optimal representation of the relationships among the topics, the linguistic objects comprising each topic, and the documents associated with (possibly more than one) topic. In particular, for IR, we want to show the relationships between topics and documents so that a user can access documents in the context of the topics. A topic by itself can clearly be represented largely by a set of text objects. However, we need also to present arbitrary number of such topics as part of the same summary. We believe that, for adequate representation of

the resulting many-to-many relationships (which is crucial for the end-user fully understanding the summary), additional graphical components are needed in the interface.

To our knowledge, the existing studies of multi-document summarization do not place emphasis on these considerations. Radev and McKeown (1998) have shown a methodology for 'briefing' news articles reporting the same event. Barzilay et al. (1999) have proposed a method for summarizing "news articles presenting different descriptions of the same event". These studies focus on a single topic in a document collection. Mani and Bloedorn (1999) have addressed summarizing of similarities and differences among related documents with respect to a specified query or profile. In their study, several presentation strategies are suggested. Although they mention a graphical strategy, such as plotting documents sharing more terms closer together, no implementation is reported.

There are a number of different studies that address graphical presentation of multi-document (or document corpus visualization) – The VIBE System (Olsen et al., 1993; Korfhage and Olsen, 1995), Galaxy (Rennison, 1994), SPIRE Themescapes (Wise et al., 1995), LyberWorld (Hemmje et al., 1994), and applications of self-organizing map utilizing neural network technique (Kohonen, 1997; Lin, 1993; Lagus et al., 1996). In general, these studies consider documents as objects in a model space (document space, typically high-dimensional) and provide 2-D or 3-D representation of this document space. Their focus is on detecting and presenting structural relationships among documents in a corpus.

From our viewpoint, these two fields of research address two different perspectives on the multi-document analysis problem: multi-document summarization efforts largely deliver their results in textual form, while document corpus visualization research, which focuses on means for graphical representation of a document space, does not perform any summarization work. While we believe that both textual and graphical representations are essential in the context of IR, the technologies from the two fields, in general, cannot be easily combined because of methodological differences (such as differences in modeling the document set, calculating similarity measures, and choosing linguistic objects in terms of which a summary would be constructed).

Motivated by these observations, we propose one uniform framework that provides both textual and graphical representations of a document collection. In this framework, topics underlying a document collection are identified, and described by means of linguistic objects in the collection. Relationships, typically many-to-many, among documents and topics are graphically presented, together with the topic descriptions, by means of a graphical user interface specifically designed for this purpose. We focus on relatively small document collections (e.g. 100 or so top-ranked documents), observing that in a realistic environment users will not look much beyond such a cut-off point. Our approach maps linguistic objects onto a multi-dimensional space (called *semantic space*). As we will see below, the mapping is defined in a way that allows for topics with certain properties to be derived and for linguistic objects at any granularity to be compared as semantic concepts.

The rest of this paper is organized as follows. The next section describes the multi-dimensional space for the document collection. Section 3 demonstrates our prototype system and illustrates the interplay between textual and graphical aspects of the multi-document summary. Section 4 highlights the implementation of the prototype system. We will conclude in Section 5.

2 Mapping a document collection into semantic space

Semantic space is derived on the basis of analyzing relationships among linguistic objects — such as terms, sentences, and documents — in the entire collection. A *term* can be simply a 'content word', in the traditional IR sense, or it can also be construed as a phrasal unit, further representative of a concept in the document domain. In our implementation, we do, in fact, take that broader definition of terms, to incorporate all types of non-stop lexical items as well as phrasal units such as named entities, technical terminology, and other multi-word constructions (see Section 4 below).

We map linguistic objects (such as terms, sentences, and documents) to vectors in a multi-dimensional space. We construct this space so that the vectors for the objects behaving statistically similarly (and therefore presumed to be semantically similar) point in similar directions. The vectors are called *document vectors*, *sentence vectors*, and *term vectors*, according to the original linguistic

objects they are derived from; however, all vectors hold the same status in the sense that they represent some concepts. In this work, we call this multi-dimensional space *semantic space* (Ando, 2000) to distinguish it from a traditional vector space (Salton and McGill, 1983). In essence, in our semantic space, the terms related to each other are mapped to the vectors having similar directions, while a traditional vector space model treats all terms as independent from each other.

Our motivation for using semantic space is at least twofold. First, we believe that we need the high representational power of a multi-dimensional space since natural language objects are intrinsically complicated, as Deerwester et al. (1990) argued. Secondly, our definition of semantic space allows us to measure similarities among concepts and linguistic units at any granularity. Single-word terms, multi-word terms, sentences, and topics – all can be equally treated as objects representing some concept(s) when they are mapped to vectors in this space. From the viewpoint of a summarization task, this is an advantage over a traditional vector space in which terms are assumed to be independent of one another.

To detect topics underlying the document collection, we create a set of vectors in the semantic space so that every document vector is represented by (or close to) at least one vector (called *topic vector*). In other words, we provide viewpoints in the semantic space so that every document can be viewed somewhat closely from some viewpoint. Given such vector representations for topics, we can quantitatively measure the degree of associations between topics and linguistic objects by using a standard cosine similarity measure between topic vectors and linguistic object vectors. The linguistic objects with the strongest association would represent the topic most appropriately.

The algorithm we use for semantic space construction (see Figure 5 in Section 4) is closely related to singular value decomposition (SVD) used in Latent Semantic Indexing (LSI) (Deerwester et al., 1990). As in SVD, this algorithm finds statistical relationships between documents and terms by computing eigenvectors, and it performs dimensional reduction that results in a better statistical modeling. The advantages of the semantic space we described above are shared with similar approaches (such as SVD-based and Riemannian SVD-based (Jiang and Berry, 1998)). The algorithm we adopt, however, differs from others in that it achieves a high precision of similarity measurement among *all* the documents by capturing information more *evenly from every document* while, with other approaches, the documents whose statistical behaviors are different from the others tend to be less well represented. This algorithm fits well in our framework since we want to find topics by referring the similarities of *all* pairs of documents (shown later), and also we want to assume *all the documents are equal*. Full details of the semantic space construction algorithm may be found in (Ando, 2000), including evaluation results compared with SVD.

3 Visual presentation of a semantic space: combining text and graphics

In this section, to illustrate how we combine textual and graphical presentation, we demonstrate a summary that our prototype system created from 50 documents (TREC documents relevant to 'non-proliferation treaty').

The document set is presented in one full screen in relation to the underlying topics. The prototype system detected six[1] topics in this document set (see Figure 1). For each topic, three types of information are presented: a list of terms (*topic terms*), a list of sentences (*topic sentences*), and a visual representation of relevance of each document to the topic (*document map*).

Below we highlight some essential features of the interface.

Topic terms and topic sentences: The topic presented at the upper right corner of Figure 1 has the topic terms "Iraq", "Iraqi", "Kuwait", "Saddam Hussein", "embargo", "invasion", "disarm", and so on. (The frame is scrollable, thus accommodating all topic terms.) A topic typically will be addressed by more than one sentence, presented in a closely associated scrollable frame. The first topic sentence for this topic is "Israel's Air Force bombed Iraq's Osirak ...". Together, the sets of topic terms and sentences describe the topic, i.e. one 'thread' discussed in possibly several documents.

Document proxy – a "dot" represents a document: In a document map, a dot image represents each document (i.e. *document proxy*). A dot before a topic sentence is also a document proxy representing the document containing that sentence.

[1] The number of topics detected depends on the document set and the parameter setting adjusting the granularity.

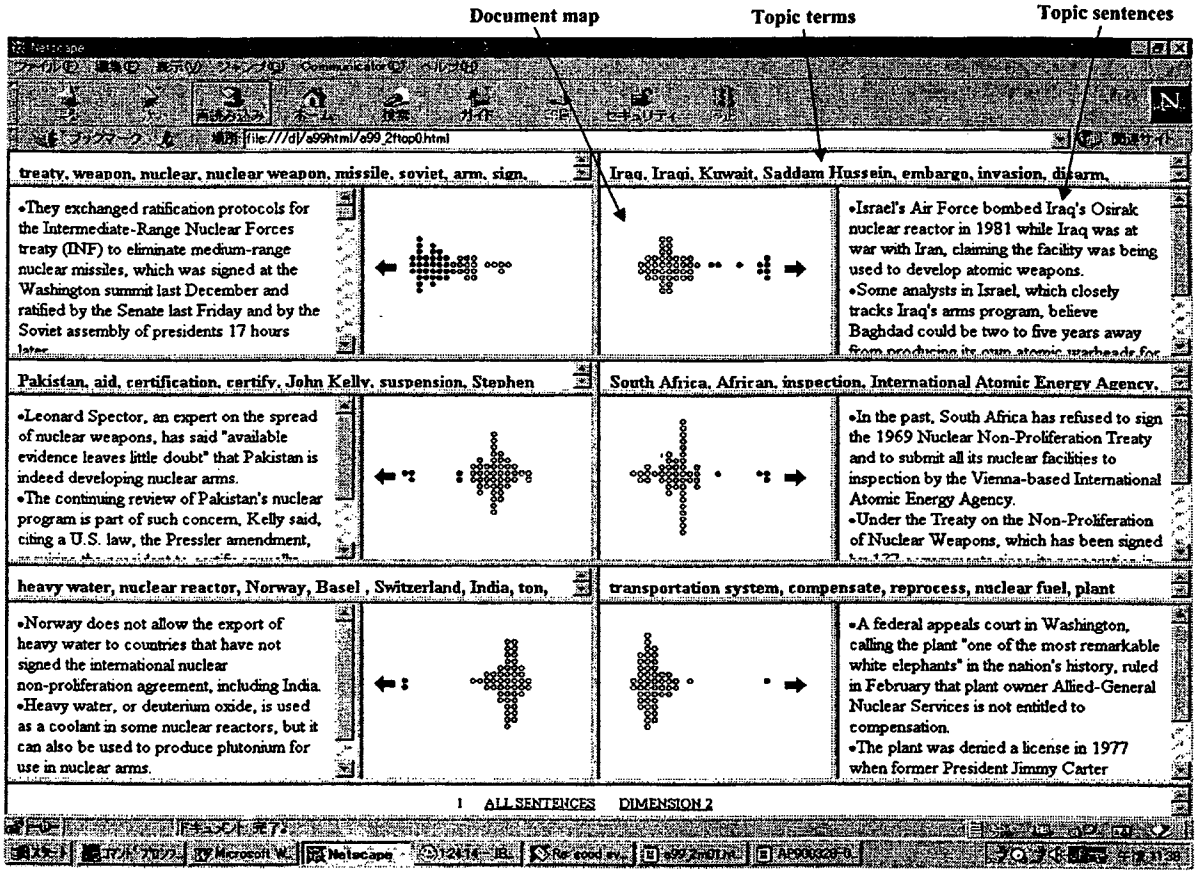


Figure 1: Example of the final output.

Document maps – topic-document relevance shown by document proxy placement and color gradation: In a document map, the horizontal placement of each dot represents the degree of relevance of the corresponding document to the topic. Documents closer to the direction of the arrow are more relevant to the topic. The color intensity of the dot also represents the degree of relevance. For instance, in the document map at the upper right corner of Figure 1, we see that there are six documents closely related to this 'Iraq-topic'. These six dots are placed on the right (the direction of the arrow), and their colors are more intense than the other document proxies. We see one more document to the left to the six documents, also with a relatively strong connection to this topic. Two documents, represented by dots almost at the center of the map, are only somewhat related to this topic. The rest of the documents, having dots that are almost transparent and placed on the left, are not very related to this topic. Thus, users can tell, at a glance, how many documents are related to each topic and how strongly they are related. Note that each document map contains proxies for all the documents. Unlike a typical clustering approach, we do not divide documents into groups. Clusters of documents, if any, are naturally observed in the document map. A document map is a projection of document vectors onto a topic vector. The semantic space allows us to detect and straightforwardly present the structural relationships among the documents.

Highlighting of document proxies – the relationships between a document and multiple topics: When a mouse rolls over a dot, the title of the document appears, and the color of the dots representing the same document in all the document maps changes (from blue to red) (see Figure 2). This color change facilitates understanding the relationships between a document and multiple topics.

A hot-link from a document proxy to full text:

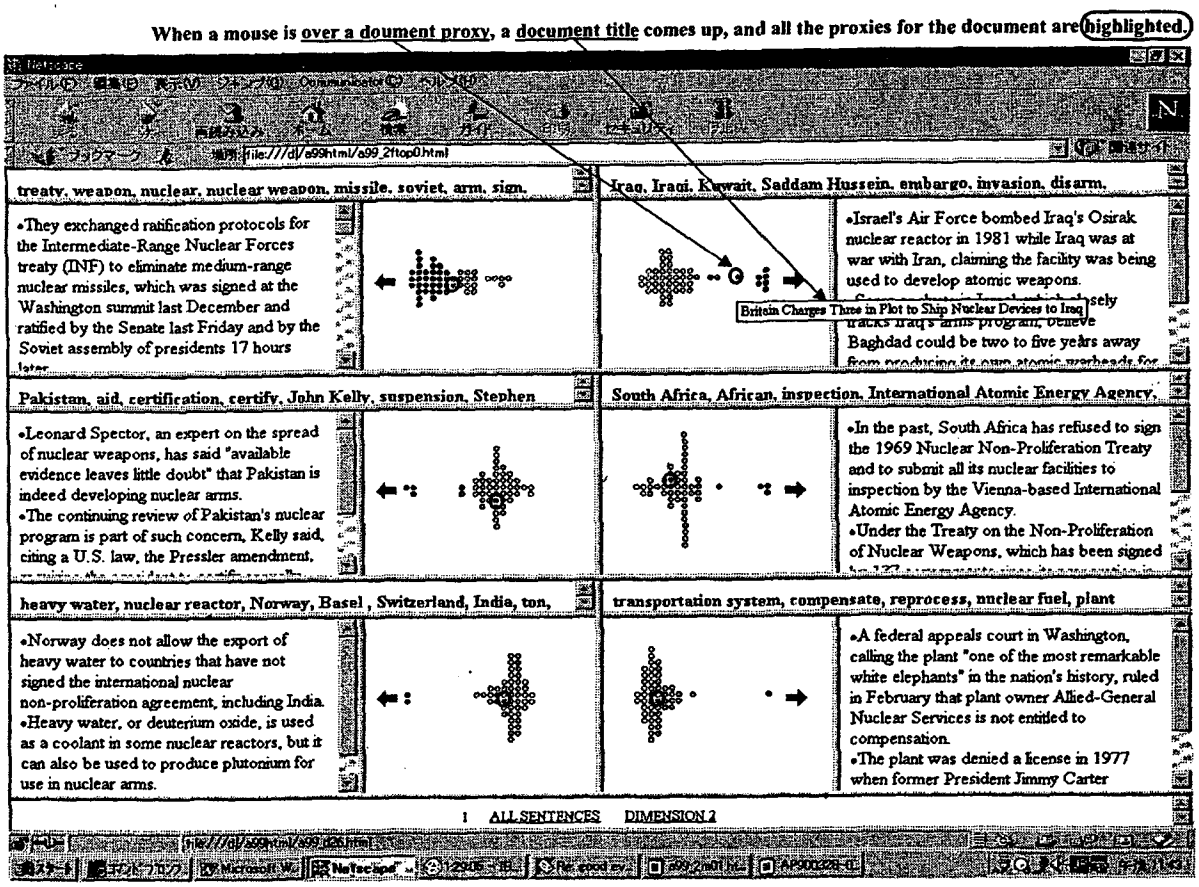


Figure 2: When a mouse rolls over a dot:

When a dot is clicked, the full text of the corresponding document is displayed in a separate window. This allows us to browse documents in the context of document-topic relationships.

Highlighting a topic sentence in the full text: When the clicked dot is associated with a topic sentence, the full text is displayed in a separate window, with the topic sentence highlighted. This highlighting helps the user to understand the context of the sentence quickly, and thus further facilitates focusing on the information of particular interest.

Topic sentences: Finally, we illustrate some of the topic sentences extracted by our system below. For each topic, the two sentences related to the topic most closely are shown.

'Iraq-topic':

• *Israel's Air Force bombed Iraq's Osirak nuclear reactor in 1981 while Iraq was at war with Iran, claiming the facility was being used to develop atomic weapons.*

• *Some analysts in Israel, which closely tracks Iraq's arms program, believe Baghdad could be two to five years away from producing its own atomic warheads for missiles or nuclear bombs to be dropped from jets.*

'Pakistan-topic':

• *Leonard Spector, an expert on the spread of nuclear weapons, has said "available evidence leaves little doubt" that Pakistan is indeed developing nuclear arms.*

• *The continuing review of Pakistan's nuclear program is part of such concern, Kelly said, citing a U.S. law, the Pressler amendment, requiring the president to certify annually that Pakistan does not possess a nuclear weapon.*

'South Africa-topic':

• *In the past, South Africa has refused to sign the 1969 Nuclear Non-Proliferation Treaty and to submit all its nuclear facilities to inspection by the Vienna-based International Atomic Energy Agency.*

• *Under the Treaty on the Non-Proliferation of Nuclear*

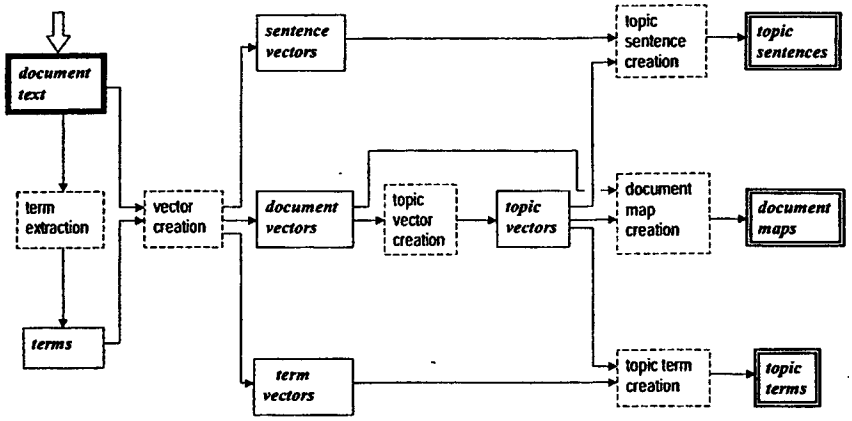


Figure 3: Overview of the process.
A block arrow indicates the input to the process, and rectangles with double-line border are the output. Rectangles with dashed line border are sub-processes. Other rectangles represent data.

Weapons, which has been signed by 137 governments since its preparation in 1969, countries without such weapons open their nuclear facilities to inspection by experts from the International Atomic Energy Agency, a U.N. agency based in Vienna.

Both for 'Iraq-' and 'Pakistan-topic', the two topic sentences address two different aspects of the similar "doubt" or "concern". For 'South Africa-topic', the second topic sentence gives background knowledge of the specific fact described in the first topic sentence. We find it interesting that, despite the fact that the two topic sentences are extracted from different documents, they appear to be consecutive sequences from a uniform source.

In essence, the design seeks to facilitate quick appreciation of the contents of a document space by supporting browsing through a document collection with easily switching between different views: topic highlights (terms), topical sentences, full document text, and inter-document relationships. At present, there is no attempt to handle redundancy between topic sentences.

4 Implementation

In this section, we describe the implementation of our prototype system. The overall process flow of this system is shown in Figure 3. Our description omits the process of creating graphical presentation that is straightforwardly understood from Section 3. The system takes, as its input, the text of a given set of documents. Throughout this section, we use the three small 'documents' shown below as an illustrative example. The data flow from these three documents to the final output is shown in Figure 4.

Document #1:
Mary Jones has a little lamb. The lamb is her good buddy.

Document #2:
Mary Jones is a veterinarian for ABC University.
ABC University has many lambs.

Document #3:
Mike Smith is a programmer for XYZ Corporation.

4.1 Term extraction

First, we extract all terms contained in the documents, using an infrastructure for document processing and analysis, comprising a number of interconnected, and mutually enabling, linguistic filters; which operates without any reference to a predefined domain. The whole infrastructure (hereafter referred to as TEXTRACT) is designed from the ground up to perform a variety of linguistic feature extraction functions, ranging from straightforward, single pass, tokenization, lexical look-up and morphological analysis, to complex aggregation of representative (salient) phrasal units across large multi-document collections (Boguraev and Neff, 2000). TEXTRACT combines functions for linguistic analysis, filtering, and normalization; these focus on morphological processing, named entity identification, technical terminology extraction, and other multi-word phrasal analysis; and are further enhanced by cross-document aggregation, resulting in some nor-

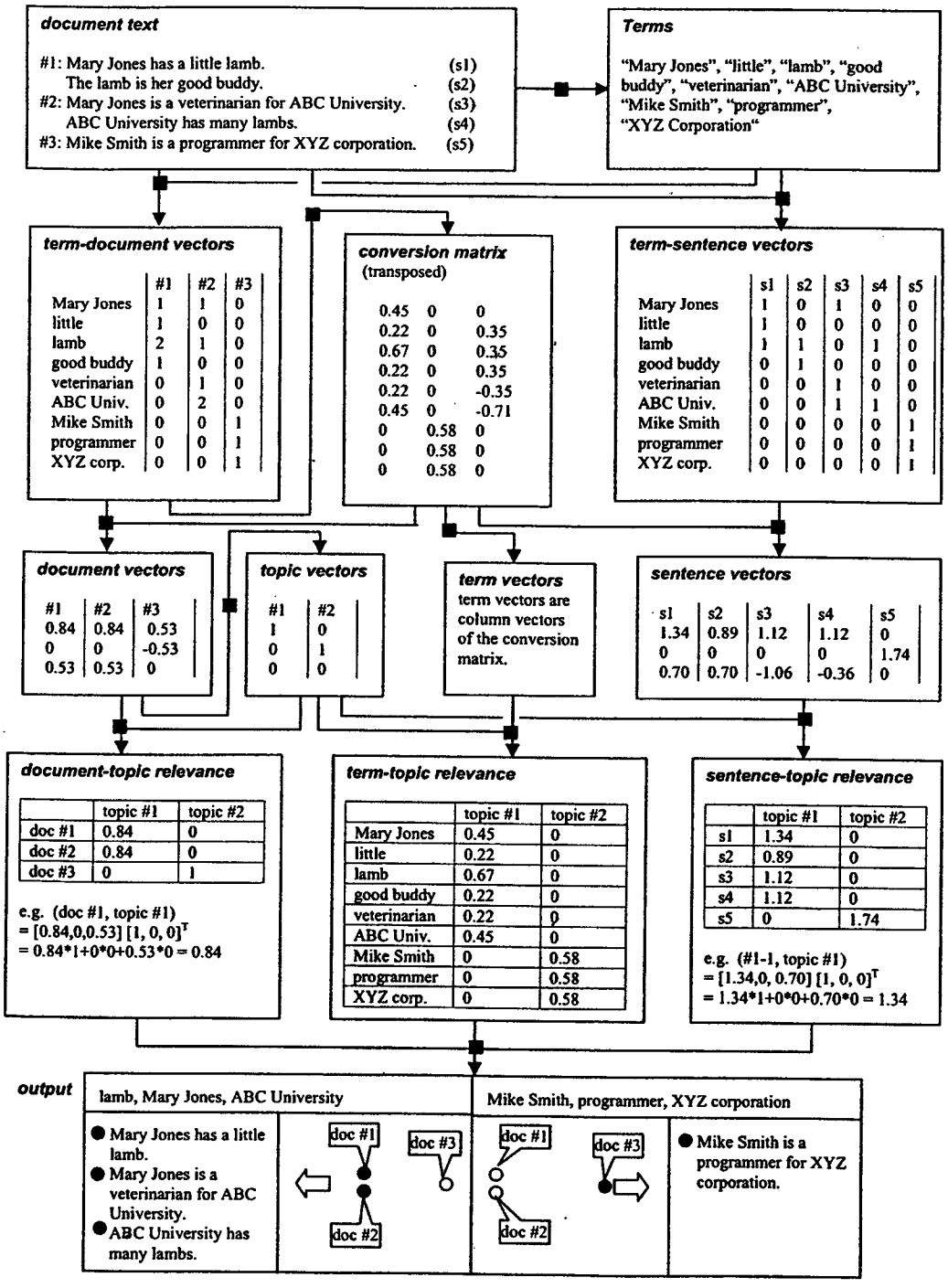


Figure 4: Example of data flow.

Procedure ConstructSemanticSpace
Input: term-document vectors $\mathbf{d}_1, ..., \mathbf{d}_n$
Output: conversion matrix $\mathbf{C}$

$\mathbf{D} = [\mathbf{d}_1 ... \mathbf{d}_n]$ /* Term-document matrix */
$\mathbf{R} = \mathbf{D}$ /* Initialize a residual matrix with the term-document matrix */
For $i = 1$ to k
  $\mathbf{R}_s = [|\mathbf{r}_1|^q \mathbf{r}_1 ... |\mathbf{r}_n|^q \mathbf{r}_n]$ /* Scale each of $\mathbf{R}$'s column vectors by a power of its own length */
  $\mathbf{c}_i$ = the eigenvector of $\mathbf{R}_s \mathbf{R}_s^T$ with the largest eigenvalue
  $\mathbf{R} = \mathbf{R} - [(\mathbf{c}_i^T \mathbf{r}_1)\mathbf{c}_i ... (\mathbf{c}_i^T \mathbf{r}_n)\mathbf{c}_i]$ /* Eliminate the direction of $\mathbf{c}_i$ from $\mathbf{R}$'s column vectors */
End for

$\mathbf{C} = [\mathbf{c}_1 ... \mathbf{c}_k]^T$ /* Conversion matrix */

Figure 5: Semantic space creation. Scaling factor q and the dimensionality k are experimentally determined.

malization to canonical forms, and simple types of co-reference resolution.

For the example mini-documents above, after removal of common stop words, the terms remaining as linguistic objects for the algorithm to operate on are listed at top of Figure 4.

4.2 Vector creation

We construct the semantic space from term-document relationships by a procedure[2] shown in Figure 5. In the semantic space, each of vector elements represents a linear combination of terms. The conversion matrix returned by the semantic space creation procedure keeps the information of these linear combinations. For instance, the conversion matrix for our example (see Figure 4) shows that the first element of a vector in the semantic space is associated with 0.45*"Mary Jones"+0.22*"little"+0.67*"lamb"+0.22*"good buddy"+0.22*"veterinarian"+0.45*"ABC University".

To map the documents to the vectors in the semantic space, we create the *term-document vectors* each of whose elements represents the degree of relevance of each term to the document. Our implementation uses term frequency as the degree of relevance. We create document vectors of the semantic space by multiplying term-document vectors and the conversion matrix. Sentences and terms can also be mapped to the vectors in the same way by treating them as "small documents".

[2]We do not describe the details of this procedure in this paper. See Section 2.

4.3 Identifying topics

Ultimately, our multi-document summaries rely crucially on identifying topics representing all the documents in the set. This is done by creating topic vectors so that *each document vector is close to (i.e. represented by) at least one topic vector*. We implement this topic vector creation process as follows. First, we create a document graph from the document vectors. In the document graph, each node represents a document vector, and two nodes have an edge between them if and only if the similarity between the two document vectors is above a threshold. Next, we detect the connected components in the document graph, and we create the topic vectors from each connected component by applying the procedure 'DetectTopic' (Figure 6) recursively.

'DetectTopic' works as follows. The unit eigenvector of a covariance matrix of the document vectors in a set S is computed as $\mathbf{v}$. It is a representative direction of the document vectors in S. If the similarity between $\mathbf{v}$ and any document vector in S is below a threshold, then S is divided into two sets S_1 and S_2 (as in Figure 7), and the procedure is called for S_1 and S_2 recursively. Otherwise, $\mathbf{v}$ is returned as a topic vector. The granularity of topic detection can be adjusted by the setting of threshold parameters.

Note that such a topic vector creation procedure essentially detects "cluster centroids" of document vectors (not sentence vectors), although grouping documents into clusters is not our purpose. This indicates that general vector-based clustering technologies could be integrated into our framework if

it brings further improvement.

4.4 Associations between topics and linguistic objects

The associations between topics and linguistic objects (documents, sentences, and terms) are measured by computing the cosine (similarity measurement) between the topic vectors and linguistic object vectors. The degree of association between topics and documents is used to create document maps. The terms and sentences with the strongest associations are chosen to be the topic terms and the topic sentences, respectively.

As a result, for our example we get the output shown at the bottom of Figure 4.

4.5 Computational complexity

Let m be the number of different terms in the document set (typically around 5000), and let n be the number of documents (typically 50 to 100)[3]. Given that $m > n$, the semantic space is constructed in $O(mn^2)$ time. The topic vectors are created in $O(n^3)$ time by using a separator tree for the computation of all-pairs minimum cut[4], assuming that the document vector set is divided evenly[5]. Let k be the dimensionality of the semantic space, and let h be the number of detected topics. Note that k and h are at most n, but are generally much smaller than n in practice. Regarding the number of terms contained in one sentence as a constant, topic sentences are extracted in $O(skh)$ time where s is the total number of sentences in the document set. Topic terms are extracted in $O(mkh)$ time. We note that the prototype system runs efficiently enough for an interactive system.

5 Conclusion and further work

This paper proposes a framework for multiple document summarization that leverages graphical elements to present a summary as a 'constellation' of topical highlights. In this framework, we detect topics underlying a given document collection, and we describe the topics by extracting related terms and sentences from the document text. Relationships among topics and documents are graphically presented using gradation of color and placement of image objects. We illustrate interactions with

```
Procedure DetectTopic(S)
Input: a set of document vectors S
Output: topic vectors

v = the unit eigenvector of a covariance matrix of
document vectors in S
Loop for each document vector d in S
   if similarity between d and v is below a threshold
   then begin
      divide S into S1 and S2
      Call DetectTopic(S1)
      Call DetectTopic(S2)
      Exit the procedure
   End if
End loop
Return v as a topic vector
```

Figure 6: Topic vector creation.

our prototype system, and describe its implementation. We re-emphasize that the framework presented here derives its strength in equal part from two components: the results of topical analysis of the document collection are displayed by means of a multi-perspective graphical interface specifically designed to highlight this analysis. Within such a philosophy for multi-document summarization, sub-components of the analysis technology can be modularly swapped in and replaced, without contradicting the overall approach.

The algorithms and subsystems comprising the document collection analysis component have been implemented and are fully operational. The paper described one possible interface, focusing on certain visual metaphors for highlighting collection topics. As this is work in progress, we plan to experiment with alternative presentation metaphors. We plan to carry out user studies, to evaluate the interface in general, and to determine optimal features, best suited to representing our linguistic object analysis and supporting navigation through query results.

Other future work will focus on determining the effects of analyzing linguistic objects to different level of granularity on the overall results. Questions to consider here, for instance, would be: what is the optimal definition of a term for this application; does it make sense to include larger phrasal units in the semantic space; or do operations over sentences, such as sentence merging or reduction, offer alternative ways of visualizing topical content.

[3] In this work, we focus on relatively small document collections; see Section 1.

[4] See (Ahuja et al., 1993) for all-pairs min cut problem.

[5] Note that Step 3 in the document vector division procedure (Figure 7) seeks for this.

It is therefore worthwhile investigating whether combining automatic summarization with intelligent multimedia presentation techniques can make the briefing generation amenable to full automation. In other words, the author should be able to use a computer program to generate an initial briefing, which she can then edit and revise as needed. The briefing can then be presented by the author if desired, or else directly by the computer (particularly useful if the briefing is being sent to someone else). The starting point for this process would be a high-level outline of the briefing on the part of the author. The outline would include references to particular information sources that had to be summarized in particular ways. If a program were able to take such outlines and generate briefings which didn't require extensive post-editing to massage into a state deemed acceptable for the task at hand, the program could be regarded as a worthwhile time saving tool.

2 Approach

Our work forms part of a larger DARPA-funded project aimed at improving analysis and decision-making in crisis situations by providing tools that allow analysts to collaborate to develop structured arguments in support of particular conclusions and to help predict likely future scenarios. These arguments, along with background evidence, are packaged together as briefings to high-level decision-makers. In leveraging automatic methods along the lines suggested above to generate briefings, our approach needs to allow the analyst to take on as much of the briefing authoring as she wants to (e.g., it may take time for her to adapt to or trust the machine, or she may want the machine to present just part of the briefing). The analyst's organisation usually will instantiate one of several templates dictating the high-level structure of a briefing; for example, a briefing may always have to begin with an executive summary. The summarization methods also need to be relatively domain-independent, given that the subject matter of crises are somewhat unpredictable; an analyst in a crisis situation is likely to be inundated with large numbers of crisis-related news and intelligence reports from many different sources. This means that we cannot require that a domain knowledge base be available to help the briefing generation process.

Given these task requirements, we have adopted an approach that is flexible about accommodating different degrees of author involvement, that is relatively neutral about the rhetorical theory underlying the briefing structure (since a template may be provided by others), and that is domain-independent. In our approach, the author creates the briefing outline, which is then fleshed out further by the system based on information in the outline. The system fills out some content by invoking specified summarizers; it also makes decisions, when needed, about output media type; it introduces narrative elements to improve the coherence of the briefing; and finally, it assembles the final presentation, making decisions about spatial layout in the process.

A briefing is represented as a tree. The structure of the tree represents the rhetorical structure of the briefing. Each node has a label, which offers a brief textual description of the node. Each leaf node has an associated goal, which, when realized, provides content for that node. There are two kinds of goals: *content-level* goals and *narrative-level* goals. Content-level goals are also of two kinds: *retrieve* goals, which retrieve existing media objects of a particular type (text, audio, image, audio, video) satisfying some description, and *create* goals, which create new media objects of these types using programs (called *summarization filters*). Narrative-level goals introduce descriptions of content at other nodes: they include captions and running text for media objects, and *segues*, which are rhetorical moves describing a transition to a node.

Ordering relations reflecting temporal and spatial layout are defined on nodes in the tree. Two coarse-grained relations, *seq* for precedence, and *par* for simultaneity, are used to specify a temporal ordering on the nodes in the tree. As an example, temporal constraints for a (tiny) tree of 9 nodes may be expressed as:

```
<ordering> <seq>
        <par>7</par>
        <par>8</par>
        <par>3</par>
        <par>4 5</par>
        <par>6</par>
```

<par>1 9</par>
<par>2</par>
</seq> </ordering>

The tree representation, along with the temporal constraints, can be rendered in text as XML; we refer to the XML representation as a *script.*

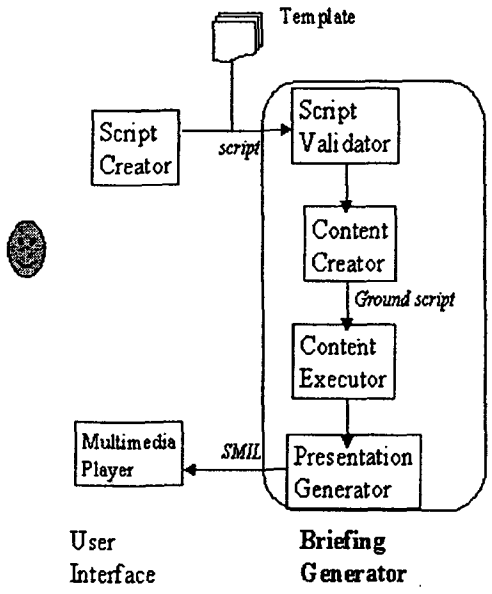


Figure 1: System Architecture

The overall architecture of our system is shown in Figure 1. The user creates the briefing outline in the form of a script, by using a GUI. The briefing generator takes the script as input. The *Script Validator* applies an XML parser to the script, to check for syntactic correctness. It then builds a tree representation for the script, which represents the briefing outline, with temporal constraints attached to the leaves of the tree.

Next, a *Content Creator* takes the input tree and expands it by introducing narrative-level goals including segues to content nodes, and running text and captions describing media objects at content nodes. Running text and short captions are generated from meta-information associated with media objects, by using shallow text generation methods (canned text). The end result of content selection (which has an XML representation called a *ground* script) is that the complete tree has been fully specified, with all the *create* and *retrieve* goals fully specified, with all the output media types decided. The Content Creator is thus responsible for both content selection and creation, in terms of tree structure and node content.

Then, a *Content Executor* executes all the *create* and *retrieve* goals. This is a very simple step, resulting in the generation of all the media objects in the presentation, except for the audio files for speech to be synthesized. Thus, this step results in realization of the content at the leaves of the tree.

Finally, the *Presentation Generator* takes the tree which is output from Content Execution, along with its temporal ordering constraints, and generates the spatial layout of the presentation. If no spatial layout constraints are specified (the default is to not specify these), the system allocates space using a simple method based on the temporal layout for nodes which have spatial manifestations. Speech synthesis is also carried out here. Once the tree is augmented with spatial layout constraints, it is translated by the Presentation Generator into SMIL[2] (Synchronized Multimedia Integration Language) (SMIL 99), a W3C-developed extension of HTML that can be played by standard multimedia players (such as Real[3] and Grins[4]. This step thus presents the realized content, synthesizing it into a multimedia presentation laid out spatially and temporally.

This particular architecture, driven by the above project requirements, does not use planning as an overall problem-solving strategy, as planning requires domain knowledge. It therefore differs from traditional intelligent multimedia presentation planners, e.g., (Wahlster et al. 93). Nevertheless, the system does make a number of intelligent decisions in organizing and coordinating presentation decisions. These are discussed next, after which we turn to the main point of the paper, namely the leveraging of summarization in automatic briefing generation.

[2] http://www.w3.org/AudioVideo/

[3] www.real.com

[4] www.oratrix.com

3 Intelligent Multimedia Presentation Generation

The author of a briefing may choose to flesh out as little of the tree as desired, with the caveat that the temporal ordering relations for non-narrative nodes need to be provided by her. When a media object is generated at a node by a *create* goal, the running text and captions are generated by the system. The motivation for this is obvious: when a summarization filter (which is a program under our control) is generating a media object, we can often provide sufficient meta-information about that object to generate a short caption and some running text. By default, all segues and spatial layout relations are also specified by the system, so the author does not have to know about these unless she wants to. Finally, the decision as to when to produce audio, when not specified by the author, is left to the system.

When summarization filters are used (for *create* goals), the media type of the output is specified as a parameter to the filter. This media type may be converted to some other type by the system, e.g., text to speech conversion using Festival (Taylor et al. 98). By default, all narrative nodes attempt to realize their goals as a speech media type, using rules based on text length and truncatability to less than 250 bytes to decide when to use text-to-speech. The truncation algorithm is based on dropping syntactic constituents, using a method similar to (Mani et al. 99). Captions are always realized, in addition, as text (i.e., they have a text realization and a possible audio realization).

Spatial layout is decided in the Presentation Generator, after all the individual media objects are created along with their temporal constraints by the Content Executor. The layout algorithm walks through the temporal ordering in sequence, allocating a segment to each set of objects that is designated to occur simultaneously (grouped by *par* in the temporal constraints). Each segment can have up to 4 frames, in each of which a media object is displayed (thus, no more than 4 media objects can be displayed at the same time). Since media objects declared to be simultaneous (using *par*) in the temporal constraints will go together in a separate segment, the temporal constraints determine what elements are grouped together in a segment. The layout within a segment handles two special cases. Captions are placed directly underneath their associated media object. Running text, when realized as text, is placed beside the media object being described, so that they are paired together visually. Thus, coherence of a segment is influenced mainly by the temporal constraints (which have been fleshed out by the Content Creator to include narrative nodes), with further handling of special cases. Of course, an individual summarization filter may choose to coordinate component multimedia objects in particular ways in the course of generating a composite multimedia object.

Details such as duration and onset of particular frames are specified in the translation to SMIL. Duration is determined by the number of frames present in a segment, unless there is an audio media object in the segment (this media object may have a spatial representation, e.g., as an audio icon, or it may not). If an audio media object occurs in a frame, the duration of all media objects in that frame is equal to the length of all the audio files in the segment. If there is no audio present in a segment, the duration is α seconds (α has a default value of 5) times the number of frames created.

4 Summarization Filters

As mentioned above, *create* goals are satisfied by summarization filters, which create new media objects summarizing information sources. These programs are called *summarization filters* because in the course of condensing information, they take input information and turn it into some more abstract and useful representation, filtering out unimportant information. Such filters provide a novel way of carrying out content selection and creation for automated presentation generation.

Our approach relies on component-based software composition, i.e., assembly of software units that have contractually specified interfaces that can be independently deployed and reused. The idea of assembling complex language processing programs out of simpler ones is

hardly new; however, by employing current industry standards to specify the interaction between the components, we simultaneously increase the robustness of the system, ensure the reusability of individual components and create a more fully plug-and-play capability. Among the core technology standards that support this plug-and-play component assembly capability are (a) Java interfaces, used to specify functions that all summarization components must implement in order to be used in the system, (b) the JavaBeans standard, which allows the parameters and methods of individual components to be inspected by the system and revealed to the users (c) the XML markup standard, which we have adopted as an inter-component communication language. Using these technologies, legacy or third-party summarizers are incorporated into the system by "wrapping" them so as to meet the interface specification of the system. These technologies also make possible a graphical environment to assemble and configure complex summarization filters from individual summarization components.

Among the most important wins over the traditional "piping" approach to filter assembly is the ability to impose build-time restrictions on the component assembly, disallowing "illegal" compositions, e.g. component X cannot provide input to component Y unless X's output type corresponds to Y's input type. Build-time restrictions such as these play a clear role in increasing the overall robustness of the run-time summarization system. Another build-time win lies in the ability of JavaBeans to be serialized, i.e., written to disk in such a way as to preserve the state of its parameters settings, ensuring that every component in the system can be configured and run at different times independently of whether the component provides a parameter file facility.

Establishing the standard functions required of a summarization filter is challenging on several fronts. One class of functions required by the interface is necessary to handle the technicalities of exchanging information between otherwise discrete components. This set includes functions for discovering a component's input and output types, for handling messages, exceptions and events passed between components and for interpreting XML based on one or more system-wide document type definitions (DTDs). The other, more interesting set of functions gets to the core of summarization functionality. Selecting these functions involves identifying parameters likely to be broadly applicable across most or all summarizers and finding ways to group them and/or to generalize them. This is desirable in order to reduce the burden on the end user of understanding the subtle differences between the various settings in the summarizers available to her.

An example of the difficulty inherent in this endeavor is provided by the *compression* (summary length divided by source length) vs. *reduction* (1's complement of compression) vs. *target length* paradigm. Different summarizers will implement one or more of these. The wrapper maps from the high-level interface function, where the application/user can specify either compression or target length, but not both, to the individual summarizer's representation. Thus, a user doesn't need to know which representation(s) a particular summarizer uses for reduction/compression.

A vanilla summarization Bean includes the following functionality, which every summarizer must be able to provide methods for:

> *source*: documents to be summarized (this can be a single document, or a collection)
> *reduction-rate*: either summary size/source size, or target length
> *audience*: user-focused or generic (user-focused requires the specification of a bag of terms, which can be of different types)
> *output-type*: specific data formats (specified by DTDs)

The above are parameters which we expect all summarizers to support. More specialized summarizer beans can be constructed to reflect groupings of summarizers. Among other parameters are *output-fluency*, which specifies whether a textual summary is to be made up of passages (sentences, paras, blocks), named entities, lists of words, phrases, or topics, etc. Given that definitions of summarization in more

theoretical terms have not been entirely satisfactory (Mani 2000), it is worth noting that the above vanilla Bean provides an *operational definition* of what a summarizer is.

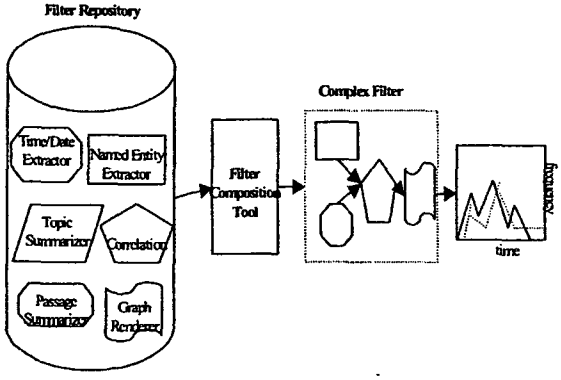


Figure 2: Summarization Filter Composition

In addition to its practical utility in the ability to assimilate, combine and reuse components in different combinations, and to do so within a GUI, this approach is interesting because it allows powerful summarization functions to be created by composing together simpler tools. (Note that this is different from automatically finding the best combination, which our system does not address). For example, Figure 2 illustrates a complex filter created by using a GUI to compose together a named entity extractor, a date extractor, a component which discovers significant associations between the two and writes the result to a table, and a visualizer which plots the results as a graph. The resulting summarizer takes in a large collection of documents, and produces as a summary a graph (a jpeg) of salient named entity mentions over time. Each of its components can be easily reused within the filter composition system to build other summarizers.

5 Narrative Summarization

As mentioned above, the system can construct a narrative to accompany the briefing. Narrative nodes are generated to cover captions, running text, and segues. The captions and running text, when not provided by the filters, are provided by the script input. In the case of *retrieve* goals, the objects may not have any meta-information, in which case a default caption and running-text is generated. Clearly, a system's explanatory narrative will be enhanced by the availability of rich meta-information.

The segues are provided by the system. For example, an item with a label "A biography of bin Laden" could result in a generated segue *"Here is a biography of bin Laden"*. The Content Creator, when providing content for narrative nodes, uses a variety of different canned text patterns. For the above example, the pattern would be *"Here is @6.label"*, where *6* is the number of a non-narrative node, with *label* being its label.

Peru Action Brief
1 Preamble
2 Situation Assessment
2.1 Chronology of Events
2.1.2 Latest document summary
create("*summarize -generic -compression .1 /peru/p32*")
2.2 Biographies
2.2.1 Biography of Victor Polay
2.2.1.1 Picture of *@2.2.2.person*
retrieve("D:\rawdata\polay.jpg")
2.2.1.2 Biography of *@2.2.2.person*
create("*summarize -bio -length 350 -span multi -person @2.2.2.person -out table /peru/**")
3 Coda
"This briefing has assessed aspects of the situation in Peru. Overall, the crisis appears to be worsening."

Figure 3: Input Script

```
Peru Action Brief
  1 Preamble
    audio = "In this briefing, I will go over
             the @2.label. This will cover
             @2.1.label and @2.3.1.label"
  2 Situation Assessment
    2.1 "An overview of the @2.2.label"
        (Meta-2.2)
    2.2 Chronology of Events
      2.2.1 audio = "Here is the @2.2.2.label"
            (Meta-2.2.2)
      2.2.2 text = "Latest document summary"
        audio = text =
        create("summarize -generic
                -compression .1 /peru/p32")
    2.3 Biographies
      2.3.1 audio =
            "A profile of @2.3.2.person"
            (Meta-2.3.2)
      2.3.2 Biography of Victor Polay
        2.3.2.1 audio = text =
                "A file photo of
                @2.3.2.person"
                (Meta-2.3.2.2)
        2.3.2.2 Picture of @2.3.2.person
          image =
          retrieve("D:\rawdata\polay.jpg")
        2.3.2.3 audio = text =
                "Profile of @2.3.2.person"
                (Meta-2.3.2.3)
        2.3.2.4 Biography of @2.3.2.person
          audio = text =
          create("summarize -bio -length 350
                  -span multi -person
                  @2.2.2.person -out table
                  /peru/*")

  3 Coda
    audio = "This briefing has assessed
    aspects of the situation in Peru. Overall,
    the crisis appears to be worsening."
```

```
<seq>
      <par>1</par>
      <par>2.2.1 2.2.2</par>
      <par>2.3.1</par>
      <par>2.3.2.1  2.3.2.2
           2.3.2.3  2.3.2.4</par>
      <par>3</par>
</seq>
```

Figure 4: Ground Script

All segue nodes are by default generated automatically by the system, based on node labels. We always introduce a segue node at the beginning of the presentation (called a preamble node), which provides a segue covering the "crown" of the tree, i.e., all nodes upto a particular depth d from the root (d=2) are marked with segue nodes. A segue node is also produced at the end (called a coda). (Both preamble and segue can of course be specified by the author if desired).

For introducing intervening segue nodes, we use the following algorithm based on the distance between nodes and the height in the tree. We traverse the non-narrative leaves of the tree in their temporal order, evaluating each pair of adjacent nodes A and B where A precedes B temporally. A segue is introduced between nodes A and B if either (a) the maximum of the 2 distances from A and B to their least common ancestor is greater than 3 nodes or (b) the sum of the 2 distances from A and B to the least common ancestor is greater than 4 nodes. This is less intrusive than introducing segues at random or between every pair of successive nodes, and appears to perform better than introducing a segue at each depth of the tree.

6 An Example

We currently have a working version of the system with a variety of different single and multi-document summarization filters. Figure 3 shows an input script created by an author (the scripts in Figure 3 and 4 are schematic representations of the scripts, rather than the raw XML). The script includes two create goals, one with a single-document generic summarization filter, the other with a multi-document user-focused summarization filter. Figure 4 shows the ground script which was created automatically by the Content Creator component. Note the addition of media type specifications, the introduction of narrative nodes, and the extension of the temporal constraints. The final presentation generated is shown in Figure 5. Here we show screen dumps of the six SMIL segments produced, with the audio if any for each segment indicated in this paper next to an audio icon.

7 Status

The summarization filters have incorporated several summarizers, including some that have been evaluated in the DARPA SUMMAC conference (Mani et al. 99-1). These carry out both single-document and multi-document summarization, and include a preliminary biographical summarizer we have developed. The running text for the biography table in the second-last segment of Figure 5 is produced from meta-information in the table XML generated by the biographical summarizer. The production method for running text uses canned text which should work for any input table conforming to that DTD.

The summarization filters are being tested as part of a DARPA situated test with end-users. The briefing generator itself has been used internally to generate numerous briefings, and has been demonstrated as part of the DARPA system. We also expect to carry out an evaluation to assess the extent to which the automation described here provides efficiency gains in briefing production.

8 Related Work

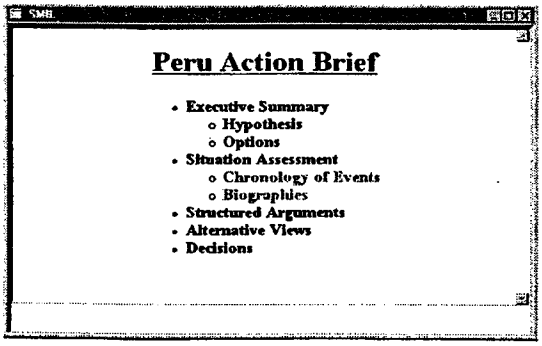
There is a fair amount of work on automatic authoring of multimedia presentations, e.g., (Wahlster et al. 93), (Dalal et al. 96), (Mittal et al. 95), (Andre and Rist 97)[5]. These efforts differ from ours in two ways: first, unlike us, they are not open-domain; and, second, they don't use summarization components. While such efforts are extremely sophisticated compared to us in multimedia presentation planning and fine-grained coordination and synchronization capabilities, many of the components used in those efforts are clearly applicable to our work. For example, (Andre and Rist 96) include methods for leveraging lifelike characters in this process; these characters can be leveraged in our work as well, to help personify the computer narrator. In addition, our captions, which are very short, rely on canned text based on node labels in the initial script, or based on shallow meta-information generated by the summarization filter (in XML) along with the created media object. (Mittal et al. 95) describe a variety of strategies for generation of longer, more explanatory captions, some of which may be exploited in our work by deepening the level of meta-information, at least for summarization components developed by us.

In our ability to leverage automatic summarization, our work should be clearly distinguished from work which attempts to format a summary (from an XML representation) into something akin to a Powerpoint briefing, e.g., (Nagao and Hasida 98). Our work, by contrast, is focused on using summarization in generating briefings from an abstract outline.

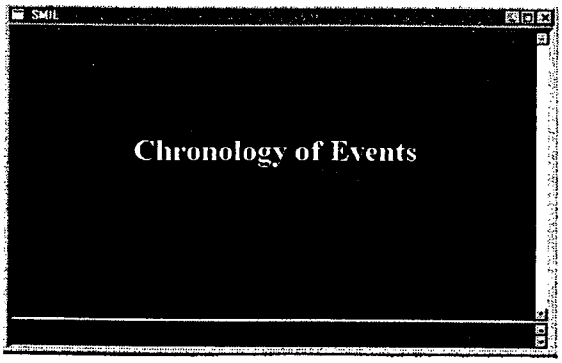
9 Conclusion

We have described methods for leveraging automatic summarization in the automatic generation of multimedia briefings. This work has taken an open-domain approach, in order to meet the requirements of the DARPA application we are involved with. We believe there is a stronger role that NL generation can play in the narrative aspects of our briefings, which currently rely for the most part on canned text. Our future work on description merging in biographical summaries, and on introducing referring expressions into the narrative nodes, would in effect take advantage of more powerful generation methods, without sacrificing open-domain capabilities. This may require much richer meta-information specifications than the ones we currently use.

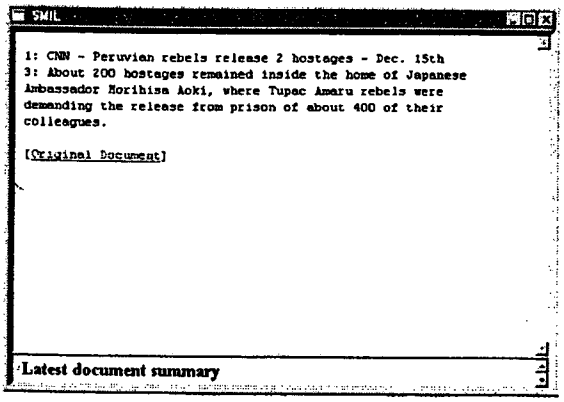
Finally, we have begun the design of the Script Creator GUI (the only component in Figure 1 remaining to be built). This will allow the author to create scripts for the briefing generator (instead of editing templates by hand), by laying out icons for media objects in temporal order. A user will be able to select a "standard" briefing template from a menu, and then view it in a briefing/template structure editor. The user can then provide content by adding annotations to any node in the briefing template. The user will have a choice of saving the edit version in template form, or in SMIL or possibly Microsoft Powerpoint format.



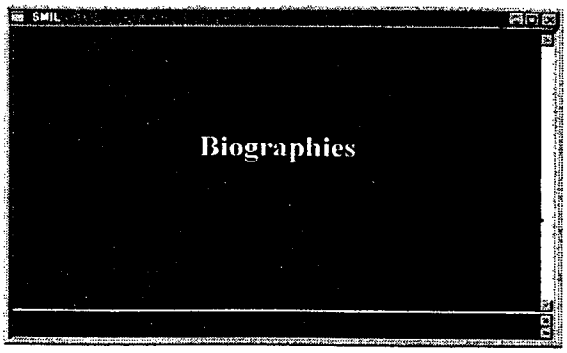
- In this briefing I will go over the situation assessment. This will cover an overview of the chronology of events and a profile of Victor Polay.



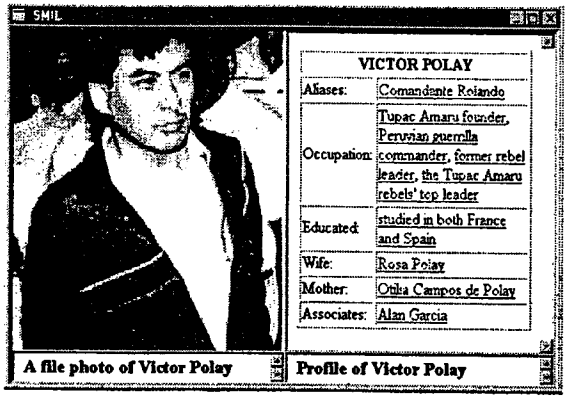
- Here is an overview of the chronology of events.



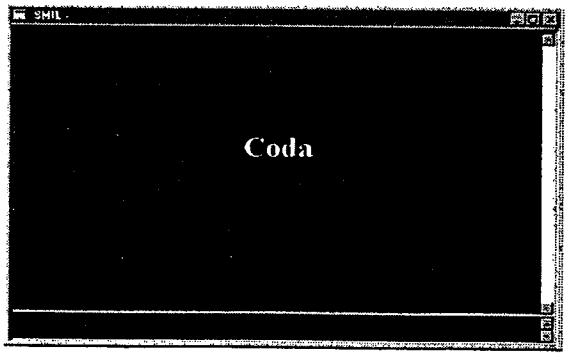
- Here is the latest document summary.



- Next, a biography of Victor Polay.



- Victor Polay, also known as Comandante Rolando, is the Tupac Amaru founder, a Peruvian guerrilla commander, a former rebel leader, and the Tupac Amaru rebels' top leader. He studied in both France and Spain. His wife is Rosa Polay and his mother is Otilia Campos de Polay. His associates include Alan Garcia.



- This briefing has assessed aspects of the situation in Peru. Overall, the crisis appears to be worsening.

Figure 5: Presentation

References

Andre, E. and Rist, T. (1997) *Towards a New Generation of Hypermedia Systems: Extending Automated Presentation Design for Hypermedia.* L. Dybkjaer, ed., Proceedings of the Third Spoken Dialogue and Discourse Workshop, Topics in Natural Interactive Systems 1. The Maersk Mc-Kinney Moller Institute for Production Technology, Odense University, Denmark, pp. 10-27.

Dalal, M., Feiner, S., McKeown, K., Pan, S., Zhou, M., Hollerer, T., Shaw, J., Feng, Y., and Fromer, J. (1996) *Negotiation for Automated Generation of Temporal MultimediaPresentations.* Proceedings of ACM Multimedia '96.

Mani, I., Gates, B., and Bloedorn, E. (1999) *Improving Summaries by Revising Them.* Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, College Park, MD, pp. 558-565.

Mani, I., Firmin, T., House, D., Klein, G., Sundheim, B., and Hirschman, L. (1999) *The TIPSTER SUMMAC Text Summarization Evaluation.* Proceedings of EACL'99, Bergen, Norway, pp. 77-85.

Mani, I. (2000) *Automatic Text Summarization.* John Benjamins Publishing Company. To appear.

Mittal, V., Roth, S., Moore, J., Mattis, J., and Carenini, G. (1995) *Generating Explanatory Captions for Information Graphics.* Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95), pp. 1276-1283.

Nagao, K. and K. Hasida, K. (1998) *Automatic Text Summarization Based on the Global Document Annotation.* Proceedings of COLING'98, Montreal, pp. 917-921.

Power, R. and Scott, D. (1998) *Multilingual Authoring using Feedback Texts.* Proceedings of COLING'98, Montreal, pp. 1053-1059.

Taylor, P., Black, A., and Caley, R. (1998) *The architecture of the Festival Speech Synthesis System.* Proceedings of the Third ESCA Workshop on Speech Synthesis, Jenolan Caves, Australia, pp. 147-151.

Wahlster, W., Andre, E., Finkler, W., Profitlich, H.-J., and Rist, T. (1993) *Plan-Based Integration of Natural Language and Graphics Generation.* AI Journal, 63.