Cross-lingual C*ST*RD: English access to Hindi information

Cross-Lingual C*ST*RD: English Access to Hindi Information

ANTON LEUSKI, CHIN-YEW LIN, LIANG ZHOU, ULRICH GERMANN,
FRANZ JOSEF OCH, and EDUARD HOVY
Information Sciences Institute, University of Southern California

We present C*ST*RD, a cross-language information delivery system that supports cross-language information retrieval, information space visualization and navigation, machine translation, and text summarization of single documents and clusters of documents. C*ST*RD was assembled and trained within 1 month, in the context of DARPA's Surprise Language Exercise, that selected as source a heretofore unstudied language, Hindi. Given the brief time, we could not create deep Hindi capabilities for all the modules, but instead experimented with combining shallow Hindi capabilities, or even English-only modules, into one integrated system. Various possible configurations, with different tradeoffs in processing speed and ease of use, enable the rapid deployment of C*ST*RD to new languages under various conditions.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**]: Natural Language Processing——*machine translation; text analysis; language generation*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Design, Experimentation, Human factors, Languages, Management, Performance

Additional Key Words and Phrases: Cross-language information retrieval, Hindi-to-English machine translation, information retrieval and information space navigation, single- and multi-document text summarization, headline generation

1. INTRODUCTION

The goal of DARPA's 2003 TIDES Surprise Language Exercise was to test the Human Language Technology community's ability to rapidly create language tools for previously unresearched languages. We focused our attention on the task of providing human access to information that is available only in a language of which the user has little or no knowledge. During 29 days in June, members of ISI's Natural Language Group adapted their Natural Language Processing tools to Hindi and integrated them into C*ST*RD,[1] a single information exploration platform that supports cross-language information retrieval,

[1] Pronounced *custard*, standing for Clustering, Summarization, Translation, Reformatting and Display.

This work was supported by the DARPA TIDES program under contracts nos. N66001-00-1-8914 and N66001-00-1-8916.

information space visualization and navigation, machine translation, and text summarization of single documents and clusters of documents.

A core question in such integration is when in the information delivery pipeline to deploy machine translation (MT): one can translate the full source collection and then perform English-only retrieval, summarization, and so on, or one can perform foreign-language operations and translate only the minimum required to show the user for information space navigation. The optimal system configuration for this tradeoff—the computational expense of MT versus the programming expense of creating foreign language capabilities for the other modules—has not yet been determined.

Whatever one decides, MT obviously plays a pivotal role in this endeavor—the language barrier must be crossed at some point. While it is desirable in any case to shield the user from nonrelevant information and minimize the amount of text he or she has to read in order to obtain the information needed, this is especially true for MT output. For example, in an exercise on rapid development of MT for Tamil in 2001 [Germann 2001], evaluators were asked to extract information from approximately 10 pages of the MT output. They experienced this task as extremely tedious, tiring, and frustrating. Despite encouraging progress in the MT quality over the past years, MT output is still, for the most part, ungrammatical and quite hard to read. Limiting the amount of text the user has to scan to obtain information is therefore crucial. Coupled with the fact that higher-quality MT tends to be slow and computationally expensive, one would prefer to perform as little MT as possible, as late as possible.

Our model of the cross-lingual information access task is therefore based on two assumptions. First, the user is not familiar with the Hindi language and thus needs the system to translate the text. In Section 2, we describe our MT technique, present some evaluation results, and show that we have created an effective system that produces readable, albeit not quite fluent, text.

Second, we want to minimize the amount of translated text the user has to read to find the relevant information. For this purpose we developed C*ST*RD, an interactive information access system that integrates various language technologies, including information retrieval (IR), document space exploration, and single- and multi-document summarization. Our aim is to provide an integrated solution where the user begins by typing a query into a search system, receives back a set of documents, and uses several document organization and visualization tools to locate relevant documents quickly. In Section 3, we describe *Lighthouse*, one of two main components of C*ST*RD that handles IR, clustering, and document space exploration.

Lighthouse operates at the granularity of single documents. This means that, once Lighthouse has potentially retrieved relevant documents, the user has to open and read a whole document at a time to locate the interesting information. Therefore, we include *iNeATS*,[2] the second main component of C*ST*RD, which is an interactive multi-document summarization tool that allows the user to focus on the most interesting parts of the retrieved texts, ignoring nonrelevant content. iNeATS can summarize either individual documents or clusters of

[2] Interactive Next generation Automatic Text Summarization.

documents. We describe the iNeATS component of C*ST*RD in Section 4. iNeATS produces paragraph-sized summaries, that is, texts of approximately 100–400 words long. While adequate for exploring one or more documents, this length is cumbersome when the system is displaying many clusters of documents. Therefore, in Section 5, we introduce another summarization technology, also included in C*ST*RD, that compresses text even further to produce single- and multi-document headlines. These headlines are sentence-sized, that is, 10–15 words long, and define the main topics of the retrieved documents.

In Section 6, we discuss the implications of different architectural decisions regarding performing MT early or late, and of performing IR and summarization on the source Hindi or translated English.

2. MACHINE TRANSLATION

MT is central to the system's cross-lingual capabilities. The Surprise Language experiment was, among other things, also a test of the promise of statistical MT to allow the rapid development of robust MT systems for new languages.

Statistical MT systems use statistical models of translation relations to assess the likelihood of a, say, English string being the translation of some foreign input. Three factors determine the quality of a statistical MT system: (1) the quality of the model; (2) the accuracy of parameter estimation (training); and (3) the quality of the search.

Our statistical translation model is based on the alignment template approach [Och et al. 1999] embedded in a log-linear translation model [Och and Ney 2002] that uses discriminative training with the BLEU score [Papineni et al. 2001] as an objective function [Och 2003]. In the alignment template translation model, a sentence is translated by segmenting the input sentence into phrases, translating these phrases, and reordering the translations in the target language. A major difference of this approach from the often used single-word-based translation models of Brown et al. [1993] is that local word context is explicitly taken into account in the translation model.

The main training data used to train the system comes from a large set of different web sources that were assembled by a variety of participating sites throughout the course of the surprise language experiment. The final sentence-aligned training data included about 4.2 million English and 4.7 million Hindi words. In order to obtain reference translations for discriminative training and for evaluation to monitor development progress, we commissioned human translations of about 1,000 sentences (20,000 words of Hindi) from Hindi news agency reports into English. The hope is that by using news-related 'tuning' corpora, the training procedure adapts the system to the domain we are actually interested.

We use a dynamic programming beam-search algorithm to explore a subset of all possible translations [Och et al. 1999], and extract *n*-best candidate translations using A* search [Ueffing et al. 2002]. These *n*-best candidate translations are the basis for discriminative training of the model parameters with respect to translation quality.

More details on this system can be found in Oard and Och [2003], where the adaptation of the same core alignment template MT system to Cebunao is described.

During translation, word-reordering operations are the most time-consuming. At the same time, their payoff is often low [Germann 2003]. It is possible to forgo this step, producing slightly lower quality output in return for significant speedup in translation time. Since we needed to translate entire document collections for subsequent processing, we performed these translations with *monotone* decoding, that is, while word reorderings were possible locally within the scope of the alignment templates, entire templates were not reordered. This decision was based on two considerations:

(1) Word order is not important for IR.

(2) A more thorough search was impractical given the computing resources required for high-quality, high-volume translations.

The outcome of MT, even within a single month, was acceptable. In the Hindi MT evaluation organized by NIST at the end of the Surprise Language Exercise, our system obtained better results than all competing systems. It obtained a NIST score of 7.43 (on input retaining upper and lower case) and 7.80 (uncased) on the 452 sentence test corpus with four reference translations. The following text is an example output from this test data:

```
Indonesian City of Bali in October last year in the bomb blast in
the case of imam accused India of the sea on Monday began to be
averted. The attack on getting and its plan to make the charges and
decide if it were found guilty, he death sentence of May. Indonesia
of the police said that the imam sea bomb blasts in his hand claim
to be accepted. A night Club and time in the bomb blast in more
than 200 people were killed and several injured were in which most
foreign nationals.
```

A preliminary error analysis shows that major error sources are unknown words (due to incomplete lexicon coverage of the training corpus) and wrong word order in the English output produced.

ISI's approach to MT is generally language-independent. Language-specific components come into play only during pre and postprocessing and are not tightly integrated into the core MT technology. This allows us to set up MT engines rather quickly. In fact, the first MT system was available via the web within 24 h after the surprise language had been announced, albeit of very limited utility—it was based on a Hindi encoding that is used exclusively for the Bible, and trained only on a parallel English–Hindi Bible.

In addition to our web interface, we also provided bulk translations on demand and via a TCP/IP translation socket. This allowed at least two other sites (New York University and Alias-I) to integrate ISI MT technology into their systems. By-products of our training, such as word alignments and probabilistic lexicons, were made available to other sites via our resource page whenever they became available. The bottom line of our experience with MT is that within 3 weeks we were able to provide the community with MT services good enough to serve certain purposes, such as cross-lingual IR (with search on the English side) and gisting.

Even though we did not implement it, we could use the TCP/IP translation socket to provide high(er)-quality translations of selected documents or sections of documents on demand to other modules within the C*ST*RD. We discuss this in Section 6.

3. LIGHTHOUSE

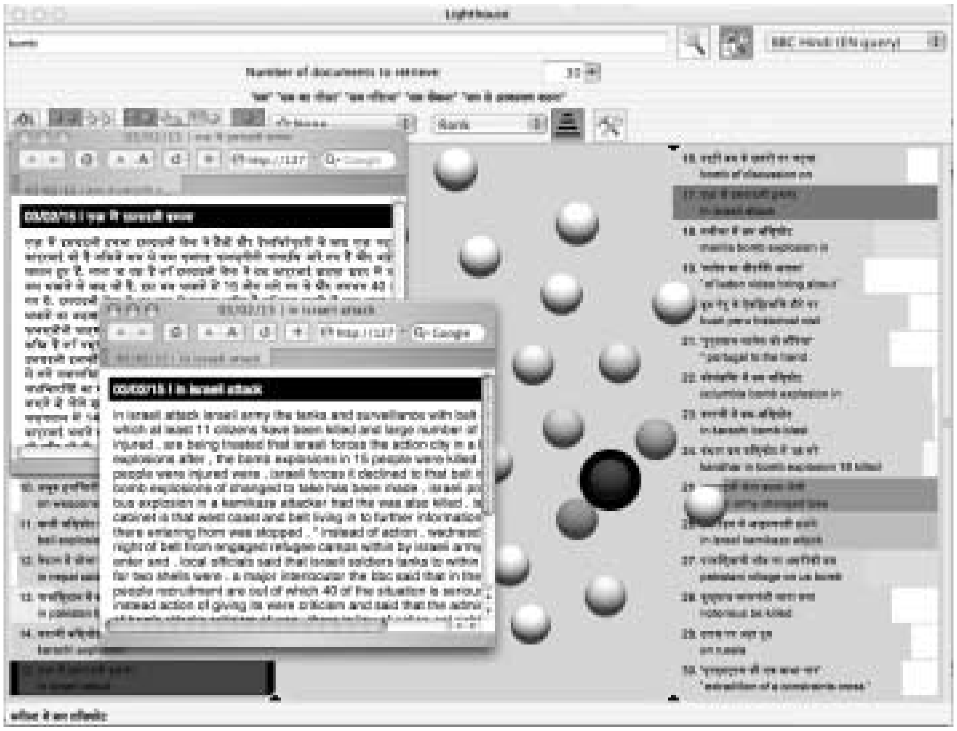
Given the very short period of the Surprise Language Exercise, we could not develop adequate training data for the IR and summarization modules. As mentioned above, we therefore decided to abbreviate the MT process and place MT early in the information delivery pipeline. (Once the raw material of the exercise had become available, however, we also could translate it fully, and deploy the remaining modules in English-only mode. Therefore, we can, in principle, configure C*ST*RD in various ways, deploying MT earlier or later; for a discussion of the possibilities see Section 6.)

For IR, display, and information space navigation, we embedded Lighthouse into C*ST*RD. Lighthouse supports full-text search and presents retrieved documents to the user in a way that facilitates effective browsing and exploration of the information. In contrast to traditional search engines (e.g., such as Google, AltaVista, etc.) that arrange the retrieved documents in a linear list by their similarity to the query, Lighthouse uses interdocument similarity in three ways to organize the retrieved document set: clustering, spatial visualization, and user-directed categorization. These tools jointly present a visual summary of the document set content to help the user locate interesting information, significantly decreasing the amount of nonrelevant material the user has to examine. In this section, we briefly describe the main components of the system and discuss how it has been adapted to take into account the cross-lingual nature of the Surprise Language experiments. A full description of Lighthouse and its features can be found in Leuski [2001b] and Leuski and Allan [2003].

3.1 Cross-Language Retrieval

The version of Lighthouse used in the Surprise Language Experiment is built on top of the Lucene search engine [Lucene 2003]. Lucene is an open source search engine written in Java. It supports full text indexing and searching using techniques similar to the best research retrieval systems. It has a powerful query language and it is easily expandable.

The default distribution of Lucene handles only European languages. We adapted the search engine to Hindi by implementing a word tokenizer for the language that breaks the input stream of text into individual terms or tokens. The tokenizer uses the standard Java 1.4 international text breaking subroutines developed at IBM [Gillam 1999]. We also implemented a stopword removal filter that consults the word list provided by the University of California, Berkeley. The list contains 543 Hindi stopwords. No stemming was


Lighthouse
bomb
Number of documents to retrieve:
30
BBC Hindi (EN query)
Rank
03/03/15 | in israeli attack

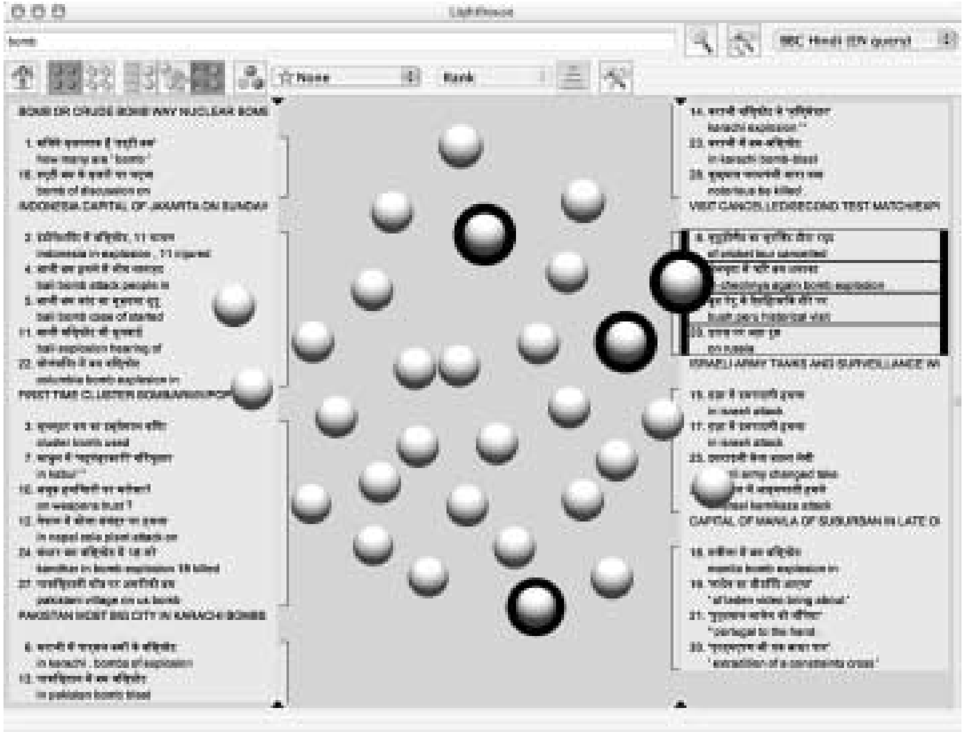
In israeli attack israeli army the tanks and surveillance with bell
which at least 11 citizens have been killed and large number of
injured , are being treated that israeli forces the action city in a
explosions after , the bomb explosions in 15 people were killed
people were injured were , israeli forces it declined to that bell
bomb explosions of changed to take has been made , israeli
bus explosion in a kamikaze attacker had the was also killed .
cabinet is that west coast and bell living in to further information
there entering from was stopped . " instead of action , wednesd
night of bell from engaged refugee camps within by israeli army
order and . local officials said that israeli soldiers tanks to within
for two shells were . a major interlocutor the bbc said that in the
people recruitment are out of which 40 of the situation is seriou
instead action of giving its were criticism and said that the admi

Fig. 1. The Lighthouse interface with open document windows.

done because no Java-based Hindi stemmer was available to us during the Surprise Language Exercise, and there was no time to implement one ourselves.

Our test corpus was a small (16 MB) collection of 3,000 BBC news stories collected and preprocessed to remove extra HMTL formatting by the Surprise Language team at the University of Maryland at College Park (UMD). The news stories covered world events from June 2001 to May 2003. We indexed the Hindi collection using Lucene's indexing functions.

The user can search the collection by entering a query in either Hindi or English (specified using a pop-up menu next to the query field, see Figure 1, where the user-selected corpus and language with "BBC Hindi (EN query)"). The search mechanism is the same for both options, except that Lighthouse translates an English query into Hindi first, using a query translation algorithm based on an English–Hindi dictionary provided by UMD. The algorithm performs a greedy search on the English part of the lexicon using the query string and returns the corresponding Hindi parts, which are then joined into the resulting query string. Matching long phrases in the query is preferred over matching individual words. If no match is found in the first pass, the search is repeated using stemmed query words. We used the Porter stemmer [Porter 1980] to stem both the query and the lexicon. If an English term had multiple Hindi translations, we used all translation variants. Each translation variant can be weighted in proportion to its translation "likeliness," assuming the



BOMB OR CRUDE BOMB WAY NUCLEAR BOMB

1. [illegible] how many are "bomb"
18. [illegible] bomb of discussion on

INDONESIA CAPITAL OF JAKARTA ON SUNDAY

2. [illegible] indonesia in explosion , 11 injured
4. [illegible] bali bomb attack people in
5. [illegible] bali bomb case of started
11. [illegible] bali explosion hearing of
22. [illegible] colombia bomb explosion in

FIRST TIME CLUSTER BOMB

3. [illegible] cluster bomb used
7. [illegible] in kabul"
16. [illegible] on weapons trust ?
12. [illegible] in nepal solar plant attack on
24. [illegible] kandhar in bomb explosion 18 killed
27. [illegible] pakistani village on us bomb

PAKISTAN MOST BIG CITY IN KARACHI BOMBS

6. [illegible] in karachi , bombs of explosion
13. [illegible] in pakistan bomb blast

14. [illegible] karachi explosion"
23. [illegible] in karachi bomb-blast
28. [illegible] notorious be killed

VISIT CANCELLED SECOND TEST MATCH

8. [illegible] all united tour cancelled
[illegible] chechnya again bomb explosion
[illegible] bush para historical visit
20. [illegible] on russia

ISRAELI ARMY TANKS AND SURVEILLANCE

15. [illegible] in israeli attack
17. [illegible] in israeli attack
25. [illegible] army changed [illegible] kamikaze attack

CAPITAL OF MANILA OF SUBURBAN IN LATE

18. [illegible] manila bomb explosion in
19. [illegible] "al-laden video bring about"
21. [illegible] "portugal to the hand
30. [illegible] "extradition of a constituents cross"

Fig. 2. The Lighthouse interface.

likeliness information is present in the dictionary [Pirkola 1998]; since, however, the UMD dictionary does not contain such information, all translations were treated as equivalent . English words not found in the lexicon were copied to the result unchanged.

An alternative to dictionary-based query translation is to use the abovementioned statistical MT module. However, several factors lead us to adopt the former approach: (1) it proved easier to integrate dictionary-based translation into Lighthouse, (2) the access latency for the local dictionary-based translation is smaller than for the TCP-socket-based service the MT system provides, and (3) the statistical MT engine was designed to translate whole sentences and not queries, which are generally disfluent.

Lighthouse displays the translated Hindi query to the user (e.g., in Figure 1, the user's query "bomb" is shown, with its translation variants below, centered).

3.2 Ranked List

Lighthouse presents to the user the top portion of the retrieved document set. The size of the retrieved set is defined by the user. In Figure 2, the limit is set to 30.

The retrieved documents are shown as the ranked list of document headlines and a set of spheres arranged in either 2- or 3-dimensional space. Each sphere corresponds to a document in the retrieved set, and the spheres are positioned in proportion to the interdocument similarity: a pair of similar document will be represented by two spheres that are close to each other. We describe the details of the latter presentation based on the spring-embedding visualization algorithm in Section 3.4.

The ranked list is broken into two columns of equal length, flanking the configuration of spheres left and right. The columns flow starting from top left down and again from the top right. The pages are ranked by the search engine in the order they are presumed to be relevant to the query. A rank number precedes each document in the list. For each document we show both the original Hindi headline and the English headline produced by MT, as described in Section 2. The documents in the list can be ordered by their rank or alphabetically by either version of the headline.

A click on the document title (or sphere) with the right mouse button brings up a pop-up menu that allows the user select either the original Hindi document text or the translated English text to be opened in the web browser (see Figure 1). While developing the C*ST*RD system we pretranslated all documents in the collection so the translated headlines and document texts were available locally. An alternative approach would be to support on-the-fly document translation. Lighthouse implements document text-viewing requests as HTTP requests. We did not implement this option due to time limitations.

3.3 Unsupervised Clustering

The clustering subsystem of Lighthouse partitions the retrieved documents in a set of nonoverlapping clusters and groups the corresponding headlines in the list. The main assumption behind using this technique is the Cluster Hypothesis of IR: “closely associated documents tend to be relevant to the same requests” [van Rijsbergen 1979, p. 45]. Using this hypothesis, the clustering will place them together in the same cluster. Once the user finds one relevant document, he or she is likely to find more relevant documents by examining the rest of the cluster. The Cluster Hypothesis has been studied in the context of improving search and browsing performance by preclustering the entire collection [Willett 1988; Cutting et al. 1992; Cutting et al. 1993]. Croft [1978] and more recently Hearst and Pedersen [1996] showed that the Cluster Hypothesis holds in a retrieved set of documents.

Figure 2 shows the 30 retrieved documents partitioned into seven clusters. Each cluster is represented by a rectangular bracket or “handle” that runs parallel to the cluster. We order the documents in the clusters using their rank and sort the clusters using the rank of the highest-ranked document in each cluster. Scanning the document headlines we can tell that, for example, the second cluster talks about an explosion in the Indonesian city of Bali and the fourth cluster contains documents that deal with an explosion in Karachi. Our monolingual English experiments show that such a document organization can be much more effective in helping the user to locate the relevant information than the ranked list [Leuski 2001a].

A cluster headline precedes each cluster. The headline is produced by the *GOSP*[3] multi-document headline generation system of C*ST*RD (see Section 5) [Zhou and Hovy 2003]. The GOSP system is implemented as a Perl script, which is called from the main Lighthouse Java code on demand. The cluster headlines are generated from the English translations of the Hindi documents produced by the MT module of C*ST*RD.

Lighthouse uses the Ward hierarchical agglomerative clustering algorithm to generate the document set partition [Mirkin 1996]. On input, the algorithm receives a set of objects and a matrix of interobject distances. It starts by assigning each object to its unique cluster. The algorithm iterates through the current cluster set by selecting the closest pair of clusters and merging them together, forming a new cluster that replaces them in the cluster set. We terminate the clustering process as soon as the distance between the closest pair of clusters exceeds a predefined threshold. This threshold is set to a value that generally produces good clusters on a standard collection of documents [Leuski 2001a]. Using the clustering algorithm on another collection may require adjustments of the threshold value. Also, the user's task may require a cluster granularity level that is different from the default setting. Lighthouse thereby provides the user with controls for threshold adjustments.

To compute interdocument distances, we employ the vector-space model for document representation [Salton 1989]. Each document j is defined as vector V_j, where $v_{i,j}$ is the weight in this document of the ith term in the vocabulary. The term weight is determined by an ad-hoc formula [Allan et al. 1998] that combines Okapi's tf score [Robertson et al. 1995] and INQUERY's normalized idf score:

$$v_{i,j} = \frac{tf_{i,j}}{tf_{i,j} + 0.5 + 1.5\frac{doclen_j}{avgdoclen}} \frac{\log(\frac{colsize+0.5}{docf_i})}{\log(colsize+1)},$$

where $v_{i,j}$ is the weight of the ith term in the vocabulary in the jth document; $tf_{i,j}$, the number of times the term occurs in the document; $docf_i$, the number of documents the term occurs in; $doclen_j$, the number of terms in the document; $avgdoclen$, the average number of terms per document in the collection; and $colsize$, the number of documents in the collection. The similarity between a pair of documents is computed as the cosine of the angle between the corresponding vectors ($\cos\theta$) [Salton 1989]. In this study, we use the inverse of the cosine ($1/\cos\theta$) to define the distance between a pair of documents.

3.4 Spatial Visualization

Partitioning the document set into nonoverlapping clusters reduces interdocument relationships and hence simplifies display and manipulation of the retrieved set. It is very easy from a user's point of view to tell a pair of similar documents from a pair of dissimilar ones, since they are assigned to the same cluster. This simplification comes at the cost of losing the intricate details of interdocument similarity that might otherwise be useful for locating relevant

[3] Global Word Selection with Localized Phrase Clustering.

information. Additionally, the clustering algorithm is a parametric approach. Determining the best value for the threshold that defines the final partition of the document set is a very hard question that one would like to avoid in a real-world system.

To position the spheres in 2- or 3-dimensional space in proportion to interdocument similarity, Lighthouse uses the same interdocument distance matrix described in the previous section. However, the visualization discussed in this section does not require any clusters—when a user sees spheres arranged in groups, that is just an artifact of the interdocument similarity. Simply put, it just draws the documents, illustrating any structure that is already present in the data. Assigning any meaning to the structure is the user's concern. The reason for the difference with clustering is that the clustering threshold parameter introduces a discontinuity that partitions the space, while the visualization display is performed in a continuum.

Figure 2 illustrates one advantage of visualization over clustering. Here we selected the fifth cluster. It contains four documents, which the clustering algorithm deemed to be similar enough. The document titles and spheres are drawn with the black outline. The first document in the cluster discusses an explosion in Karachi, the second and the fourth documents discuss Chechnya, and the third document talks about President Bush's visit to Peru. These four documents are in fact different from each other and this fact can be quickly noted by looking at the visualization—the corresponding spheres lie in different parts of the screen. Clearly the clustering algorithm made a mistake in bringing together these documents. Probably we need to adjust the clustering threshold.

In contrast, spatial visualization does not have any parameters to adjust. In this continuum, the Cluster Hypothesis still applies: relevant documents tend to group together. Once the user finds a relevant document, the spheres for other relevant documents will be located nearby.

Our experiments with spatial visualization in monolingual settings showed that such a presentation can be used effectively to direct the user's search for relevant information in the top ranked portion of the retrieved set [Allan et al. 2000]. We have experimentally shown that this approach significantly exceeds the initial performance of the ranked list and rivals in its effectiveness the traditional Rocchio relevance feedback methods [Rocchio 1971; Buckley and Salton 1995; Allan et al. 1997; Leuski 2000].

To generate a set of spheres that represents the multidimensional document vectors we use a Multidimensional Scaling algorithm called spring-embedding [Fruchterman and Reingold 1991]. The spring-embedding algorithm models each document vector as an object in 2- or 3-dimensional visualization space. It is assumed that the objects repel each other with a constant force. They are connected with springs, where the strength of each spring is proportional to the cosine similarity between the corresponding document vectors. This "mechanical" model begins with a random arrangement of objects and oscillates due to existing tension forces in the springs until it reaches a state with "minimum energy", namely when the constraints imposed on object placement best satisfied. The result of the algorithm is a set of points in space, where each point represents a document and the interpoint distances closely mimic the interdocument dissimilarity.

Each sphere in the visualization is linked to the corresponding document title in the ranked list, so clicking on the sphere will select the title and vice versa. The user can examine the clustering structure and place it in the best viewing angle by rotating, zooming, and sliding the whole structure while dragging the mouse pointer. (Only the spheres can be manipulated in this fashion; the ranked list remains in place.)

3.5 Supervised Classification

What if the user has difficulty with the spatial navigation and the clustering algorithm produces unsatisfactory partitions of the retrieved document set? Lighthouse can also categorize the documents based on the user's examples: The user first selects one or more documents and assigns them to a particular category. Lighthouse then dynamically computes the likelihood of the other documents being assigned to the category and presents this information to the user.

For clarity of display, each category is assigned a color. The category assignments are indicated by painting the corresponding document titles and spheres with the category color. The user starts with two categories: "relevant" and "nonrelevant." He or she can introduce new categories at will.

The title and the sphere of the document that was assigned to the category by the user is filled with a bright shade of the category color. In contrast, the automatically assigned documents are indicated with a less intense shade of color, where the intensity of the shading is proportional to the computed likelihood. The length of the document title background shading is proportional the likelihood as well.

Figure 1 shows that we selected the document ranked 15 and assigned it to the relevant category. The document title and the sphere are shown in dark gray.[4] Lighthouse estimates that the documents 17, 25, and 26 are likely to be relevant because they are similar to the selected document. The corresponding spheres and titles are drawn in various shades of gray. For example, document 17 is very likely to be relevant—it has dark gray shading and the title background is long. Meanwhile, document 26 is less likely to be relevant, the corresponding title background is approximately half as long. In this example, the nonrelevant category is represented by the white color. Most of the other documents are estimated to be nonrelevant, they are represented as white spheres and the corresponding title backgrounds are painted as white bars.

This category assignments are computed using a supervised classification "wizard" based on a neural network [Leuski 2000]. The wizard takes into account the number of documents the user assigned to each category and the average distances between them and each unassigned document. If the user confirms or rejects the Lighthouse category assignments by marking a

[4] In this example we use different shades of gray to indicate categories due to black and white limitations of the publishing media. Lighthouse paints categories in bright colors making the markings much clearer.

document, the system dynamically recomputes its estimates for other documents and directs the user to the most interesting information. Our experiments showed that wizard-directed browsing of the retrieved document set can be significantly more effective then using the state-of-the-art relevance feedback method of IR [Leuski 2000].

3.6 Lighthouse Summary

Lighthouse is primarily a document organization and presentation system. It uses interdocument similarity information to focus the user's attention on the groups of similar documents in order to locate relevant information much more effectively than traditional ranked list presentations do. It has been shown to be very successful in decreasing the amount of nonrelevant information the user has to examine before locating the relevant documents. This is especially important in the cross-lingual settings, where the user has to read the MT output. All the technologies used in Lighthouse are language-neutral. The main challenge in adapting Lighthouse for the Surprise Language Exercise was to provide the Hindi IR system.

4. INEATS

Once the user has retrieved and identified possibly relevant cluster(s) of documents, summarization can help in various ways to reduce the amount of browsing required to find the most useful information. At the crudest granularity, cluster-based headlines (produced here by GOSP; see Section 5) characterize the whole cluster in 15 words. At the next level of granularity, document-based headlines characterize the contents of one document. At the next level, paragraph-length summaries of a cluster (produced by iNeATS; this section) and of a single document provide more detail. Most finely, paragraph-length summaries of a single document, tailored to topic word(s) input by the user, provide the most accurate drill-down into document content.

In contrast to the mostly language-neutral technologies used in Lighthouse, the summarization algorithm used in iNeATS is more language-dependent. We did not have enough time to create the training data required for building versions of the summarizer properly trained for Hindi. Our main focus was to integrate the summarizers into C*ST*RD. We show the summarization results obtained on the machine-translated versions of the Hindi documents.

iNeATS helps the user summarize and examine single or small sets of documents in more detail than allowed by Lighthouse. It can be invoked from within Lighthouse by selecting a group of documents on the Lighthouse screen and choosing "Summarize" from the pop-up menu.

Most automatic summarization systems work by assigning one or more scores to each sentence, combining the scores, ranking the sentences, extracting the top-scoring sentences from the documents, and arranging them in coherent order [McKeown et al. 2001; Over 2001]. The system has to make decisions on the summary's length, inclusion of redundant material, and focus. Any of these decisions may have a significant impact on the quality of the output.

We believe a system that directly involves the user in the summary generation process and adapts to his or her input will produce better summaries. Additionally, it has been shown that users are more satisfied with a system that visualizes its decisions and gives the user a sense of control over the process [Koenemann and Belkin 1996]. We identify three ways in which interactivity and visualization can be incorporated into the multi-document summarization process:

(1) give the user direct control over the summarization parameters such as summary length, redundancy, and focus;
(2) support rapid browsing of the document set using the summary as the starting point and combining a multi-document summary with summaries of individual documents;
(3) incorporate alternative formats for organizing and displaying the summary, for example, a set of news stories can be summarized by placing the stories on a world map based on the locations of the events described in the stories.

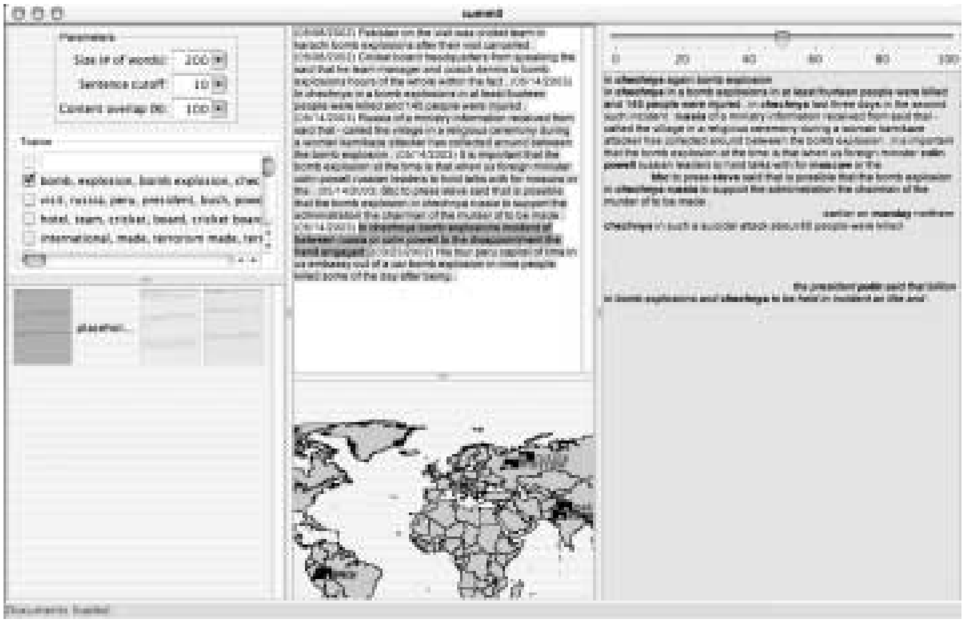
The iNeATS part of C*ST*RD addresses all three directions. It is built on top of the NeATS multi-document summarization system.

4.1 NeATS

NeATS [Lin and Hovy 2002] is an extraction-based multi-document summarization system. It is among the top two performers in the international DUC 2001 and 2002 summarization evaluation competitions organized by NIST [Over 2001]. It consists of three principal components:

Content selection. The goal of content selection is to identify important concepts mentioned in a document set. Given two sets of documents, relevant and nonrelevant, NeATS computes the likelihood ratio [Dunning 1993] to identify key concepts as unigrams, bigrams, and trigrams, and clusters these concepts, called *topic signatures* [Lin and Hovy 2000], to identify major subtopics within the main topic. (In the C*ST*RD system the relevant document set is the document set the user wants to summarize. We implement the nonrelevant document set by randomly sampling the whole collection. The size of the sample is set equal to the size of the relevant document set.) Each sentence in the document set is then ranked, using the topic signatures.

Content filtering. NeATS uses three different filters to assign importance scores to each sentence: sentence position, stigma words, and word redundancy. Sentence position has been used as an effective content filter since the late 1960s [Edmundson 1969]. NeATS applies a simple sentence filter that only retains the *N* lead sentences. Stigma words are defined as follows. Some sentences start with conjunctions, quotation marks, pronouns, or are headed by the verb “say” and its derivatives (e.g., “Joe said . . . ”). Since these stigma words usually cause discontinuities in summaries, the second filter reduces the scores of sentences containing stigma words to demote their ranks and avoid including them in summaries of small sizes. Third, to address the redundancy problem, NeATS uses a simplified version of CMU’s



Parameters

Size (# of words): 200

Sentence cutoff: 10

Content overlap (%): 100

Topics

☑ bomb, explosion, bomb explosion, chec

□ visit, russia, peru, president, bush, powe

□ hotel, team, cricket, board, cricket boar

□ international, made, terrorism made, terr

Fig. 3. The *iNeATS* interface.

MMR [Goldstein et al. 1999] algorithm. A sentence is added to the summary if and only if its content has less than X percent stemmed word overlap with the summary. NeATS uses the Porter stemmer to stem the tokens and X is a parameter of the summarization process that can assume values between 0 and 100. Once every sentence has been scored, an automatically trained function combines the scores into a single value for each sentence.

Content presentation. To ensure coherence of the summary, NeATS pairs each sentence with a lead sentence, which tends to introduce the contents of its document. It then outputs the final sentences in their chronological order.

4.2 Interactive Summarization

Figure 3 shows a screenshot of the iNeATS system. We divide the screen into three parts corresponding to the three directions outlined at the beginning of the section. The *control* panel displays the summarization parameters on the left side of the screen. The *document* panel shows the document text on the right side. The *summary* panel presents the summaries in the middle of the screen.

4.3 Controlling the Summarization Process

The top of the control panel provides the user with control over the summarization process. The first set of widgets contains controls for the summary size, sentence position, and redundancy filters. The second row of parameters displays the set of topic signatures identified by the iNeATS engine. The selected subset of the topic signatures defines the content focus for the summary. If the user enters a new value for one of the parameters or selects a different subset of the topic signatures, iNeATS immediately regenerates and redisplays the summary text in the top portion of the summary panel.

4.4 Browsing the Document Set

iNeATS facilitates browsing of the document set by providing (1) an overview of the documents, (2) linking the sentences in the summary to the original documents, and (3) using sentence blending to highlight the most relevant sentences in the documents [Leuski et al. 2003].

The bottom part of the control panel is occupied by document thumbnail displays. The thumbnails are arranged in chronological order and each document is assigned a unique color to paint the text background for the document. The same color is used to draw the document thumbnail in the control panel, to fill up the text background in the document panel, and to paint the background of those sentences in the summary that were collected from the document. For example, the screenshot shows that a user selected the second document which was assigned the orange color.[5] The document panel displays the document text on orange background. iNeATS selected summary sentences 3, 4, 5, and 6 from this document, so these four sentences are shown in the summary panel with orange background.

The sentences in the summary are linked to the original documents in two ways. First, the document can be identified by the color of the sentence. Second, each sentence is hyperlinked to the document—if the user moves the mouse over a sentence, the sentence is underlined in the summary and highlighted in the document text. For example, the first sentence of the summary is the document sentence highlighted in the document panel. If the user clicks on the sentence, iNeATS brings the source document into the document panel and scrolls the window to make the sentence visible.

The relevant parts of the documents are illuminated using the technique that we call *sentence blending*. We make the text color intensity of each sentence proportional to the relevance score computed by the NeATS engine and a blending parameter, which can be controlled by the user with a slider widget at the top of the document panel. The higher the sentence score, the darker the text is. Conversely, sentences that blend into the background have a very low sentence score. The blending parameter controls the proportion of the top ranked sentences visible on the screen at each moment. This blending affects both the full-text and the thumbnail document presentations. Combining the sentence blending with the document set overview, the user can quickly see which document contains most of the relevant material and where approximately in the document this material is placed.

For example, in Figure 3 the blending parameter is set to 50%. The first line of the document is the headline and it is ignored in summarization. The first and sixth sentences are very important and they are shown in black. These are sentences that start with “in chechnya. . . ” and “bbc to press. . . ” The second,

[5] The distinction between the colors may not be apparent on a black and white reproduction of the screenshot but it is very noticeable on a computer screen. The actual color is not crucial for this discussion.

third, and fourth sentences are less important and they are shown in gray. Apparently, the second sentence has slightly lower score than the other two because it was not selected for the summary.

Some sentences may have a very high score but do not appear in the summary because of their position in the document. We illustrate this by rendering such sentences in italics. For example, the document in Figure 3 has one such sentence, which starts with "the president bush. . . ," at the end of the document. It is shown in black italics, indicating that it has a high relevance score and its position is higher than the sentence cutoff, which is set to 10.

4.5 Alternative Summaries

The bottom part of the summary panel is occupied by a map-based visualization. We use BBN's IdentiFinder [Bikel et al. 1997] to detect the names of geographic locations in the document set. We highlight the most frequent ones on a world map, determining the geographic locations by querying the Getty Gazetteer [Getty 2003].

Each location is identified by a black dot followed by a frequency chart and the location name. The frequency chart is a bar chart where each bar corresponds to a document. The bar is painted using the document color and the length of the bar is proportional to the number of times the location name is used in the document. For example, the set used in earlier figures contains documents about explosions in Chechnya, Karachi, and Lima. iNeATS identified and displayed six locations: Russia, Moscow, Pakistan, Karachi, Peru, and Lima. We notice that the first two locations appear only in the first two documents, the next two are discussed only in the third document, and the last two locations are mentioned in the fourth document.

This panel of the display can of course be used for other multimedia displays, including graphs or bar charts for numerical information or pictures of relevant objects and places.

4.6 Multilingual Summarization

NeATS is based on SUMMARIST, an older single-document summarization system developed at ISI. Despite containing additional sentence scoring algorithms, SUMMARIST has been applied to several other languages, including Spanish, French, Italian, Chinese, and Bahasa Indonesia. In general, adapting an extractive summarization system such as SUMMARIST or NeATS to new language requires adapting or retraining the individual sentence scoring modules for the conditions of the new language. The position module, for example, exploits the fact that certain text genres exhibit a stereotypical text structure, for example in news articles the most important information appears early in the text. To the extent this is true in the new language (and genre), the position module can be employed without change. Other modules of course have to be retrained, or at least their resources have to be rebuilt. English stigma word lists, for example, are of no use in Hindi documents.

Having enough training material (examples of full texts and abstracts, or extracts), one can fairly easily adapt the individual modules. In the time allowed for the Surprise Language Exercise, we could not build or find the requisite materials, and hence had to deploy NeATS on the English produced by the MT module. While not ideal, this solution did not harm the summary quality too much; the relative crudeness of the three scoring algorithms of NeATS did not suffer from the slightly degraded quality of the interrupted MT process. We conclude that one can, without much harm, show the user summaries based on slightly degraded MT, and then only request better quality translation when the user starts exploring a particular document (see Section 6 for discussion).

4.7 Implementation and Efficiency Issues

iNeATS is implemented as a Java module that places calls to the external NeATS implementation, collects the results and presents them on the screen. The NeATS engine is written in Perl and C. It loads the documents, parses them, extracts key concepts, computes sentence scores, and generates the summaries. It takes approximately 10 s to preprocess a set of documents and create sentence ranking information on a 800 MHz PowerPC laptop. Once the preprocessing stage is complete, NeATS can compute the final summary almost instantaneously. When the user adjusts the summarization parameters, iNeATS takes approximately 1 s to regenerate and redisplay the summary.

Another time consuming operation is querying the web-based gazetteer. We addressed this problem by caching the most widely used locations locally on the hard drive.

5. HEADLINE GENERATION

Cluster headlines, displayed in the Lighthouse interface, help the user decide which document clusters are worth further examination. Since cluster headlines need to be short, sentence extraction using NeATS is not an option.

Our multi-document headline generation module, a Perl implementation of the GOSP algorithm, generates headlines for document clusters in two stages: First, it generates a headline, composed of phrases, for each document in the cluster. Then it selects among the individual document headlines the most "informative" phrases until it meets a headline length limit criterion (see Section 5.2). The length threshold of a headline is generally set to 15 words. We discuss the single-document headline system first.

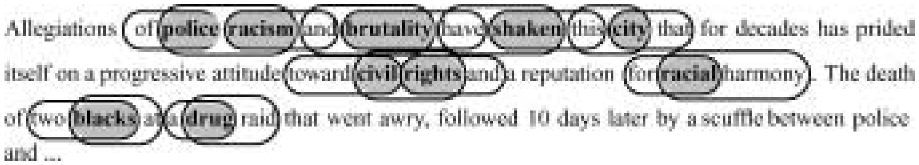
5.1 Single-Document Headline Generation

Single-document headline generation is performed as follows:

(1) *Select headline-worthy words from the document body*: Potential headline candidates are determined by statistical model trained on a collection of documents and their headlines. The scoring function combines two models of "headline worthiness":

$$Score(w) = P_{fo}(w) \times P_{tf}(w).$$

$P_{fo}(w)$ is the probability of a word w occurring in the headline given the position (measured by sentence number) of its first occurrence in the document body. It is estimated as follows.



Allegiations of police racism and brutality have shaken this city that for decades has prided itself on a progressive attitude toward civil rights and a reputation for racial harmony. The death of two blacks at a drug raid that went awry, followed 10 days later by a scuffle between police and ...

Fig. 4. GOSP forming bigram chains around headline-worthy words in the initial portion of the document body. The headline generated by the system is *police racism and brutality have shaken this city*.

Let $fo(w)$ be a function that returns the position (in terms of the sentence number) of the first occurrence of the word w in the document body of a given document, and let

$$Count\_Pos_i = \sum_{k=1}^{M} \sum_{j=1}^{N_k} \delta(fo_k(h_{k,j}) = i)$$

be the number of times a headline word has its first occurrence in the document body in position i in a document collection, where M is the number of documents in the collection; N_k, the number of words in the headline of document k; fo_k, the "first occurrence" function with respect to document k; $h_{k,j}$, the jth word in the headline of document k; and δ an evaluation function that returns 1 if the argument expression is true, 0 otherwise.

Then

$$P_{fo}(w) = \frac{Count\_Pos_{fo}}{\sum_{k=1}^{Q} Count\_Pos_Q},$$

where Q is the highest sentence number in the training collection.

An evaluation of this measure in Zhou and Hovy [2003] showed that roughly 40% (310 out of 808) of the words in headlines also occur within the first 50 words of the document body. Similar observations can be found in Zajic et al. [2002] and Lin and Hovy [1997].

The same evaluation in Zhou and Hovy [2003] also indicated that when the length of the headline is restricted, predictions are best if the sentence position model is combined with a lexicalized model based on the correlation of a word's occurrence in a document's body and its occurrence in the headline [Jin and Hauptmann 2001]:

$$P_{tf}(w) = \frac{\sum_{j=1}^{M}(TF_{body}(w, j) \times TF_{headline}(w, j))}{\sum_{j=1}^{M} TF_{body}(w, j)},$$

where $TF_{body}(w, j)$ is the number of occurrences of the word w in the document body of the jth document in the collection, and $TF_{headline}(w, j))$ the number of occurrences of w in the document's headline.

(2) *Extract phrases from the initial 50 words of the document body*: Next, the GOSP algorithm forms "bigrams chains" around each occurrence of the 10 highest-scoring words within the first 50 words of the document body (see Figure 4). This restriction is based on the aforementioned observation that the more important words in the document tend to have their first

occurrence early in the text. These bigram chains form candidate headline phrases.

The headline candidate phrases are then sorted by their length in decreasing order. Starting with the longest phrase, candidate phrases are added to the set of final headline phrases until the length threshold is met.

(3) *Cleanup*: Finally, dangling verbs, particles, conjunctions at the beginning and the end of the final headline phrases are removed. In order to do so, a part-of-speech tagger is run on all input texts. Using a set of hand-written rules, dangling words and words in the stop list are removed.

5.2 Multi-Document Headline Assembly

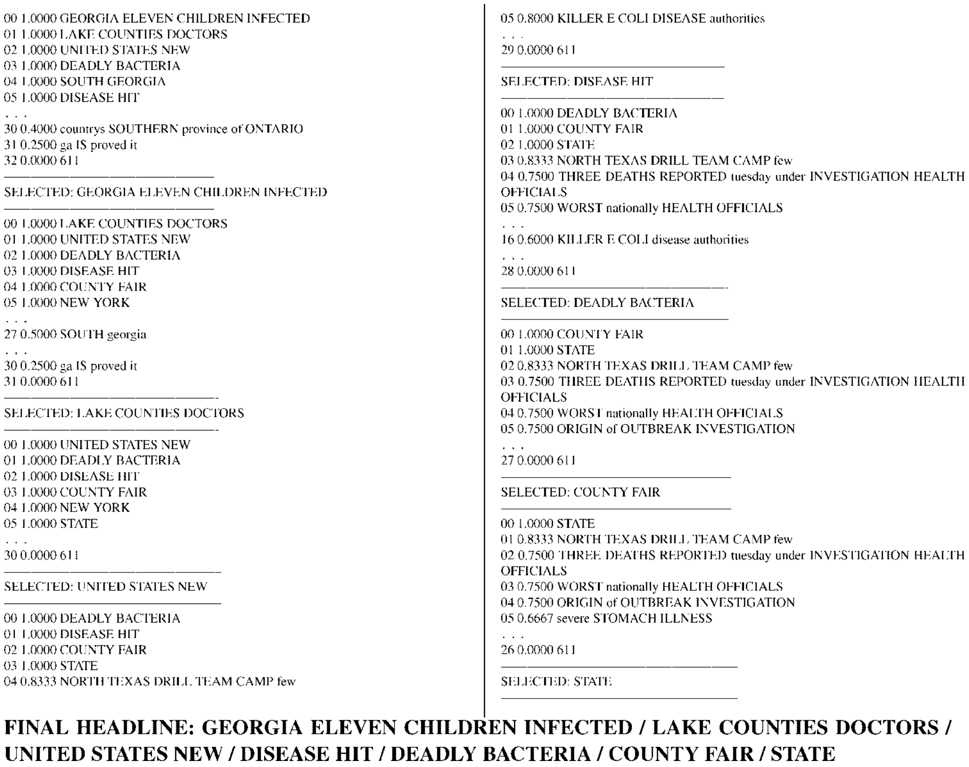
The procedure described so far generates sets of phrases for single-document headlines, resulting in a fairly large set of overlapping candidate phrases for the entire collection. From this set of candidate phrases, we must now extract the ones with the highest "information value," as measured in the ratio of headline-worthy words to the total number of words in the phrase, and the least overlap between the phrases selected.

We use a greedy strategy to select the most informative phrases. The selection process works as follows. First, all phrases in the collection are ranked by the ratio of keywords (headline-worthy words) and the total number of words in the phrase. The highest ranking one is selected. (In the sample in Figure 5, we prefer longer phrases over shorter ones if they have the same keyword ratio. Whether this is the best strategy has yet to be determined.) Once a phrase has been selected, all keywords in it lose their value as keywords, and the remaining phrases are reranked. Note, for example, that the phrase SOUTH GEORGIA drops from rank 4 to rank 27 after GEORGIA ELEVEN CHILDREN INFECTED has been selected. This is because GEORGIA has lost its value as a keyword, so that the keyword ration drops from 100% to 50%. This procedure is repeated until the headline is shorter than a predefined length threshold. In our experiments the length threshold was set to 15 words.

5.3 GOSP Before MT or MT Before GOSP?

When generating headlines for document clusters in a cross-lingual application, an important decision must be made: Should the cluster headline be generated from the source language and then translated, or is it better to generate the headline from the document translations?

In order to answer this question, we computed single-document headlines for Hindi documents and then translated the headlines using our MT system. We also computed headlines for the machine-translated versions of the same documents. There were 26 documents in the set. We compared the performance of both approaches using the automated evaluation-scoring algorithm ROUGE (Recall-Oriented Understudy for Gisting Evaluation). ROUGE, introduced by Lin and Hovy [2003] (then still under the name RED), is a measure of *n*-gram recall between candidate summaries (or headlines) and a set of reference summaries/headlines.



```
00 1.0000 GEORGIA ELEVEN CHILDREN INFECTED
01 1.0000 LAKE COUNTIES DOCTORS
02 1.0000 UNITED STATES NEW
03 1.0000 DEADLY BACTERIA
04 1.0000 SOUTH GEORGIA
05 1.0000 DISEASE HIT
. . .
30 0.4000 countrys SOUTHERN province of ONTARIO
31 0.2500 ga IS proved it
32 0.0000 611
----------------------------------------
SELECTED: GEORGIA ELEVEN CHILDREN INFECTED
----------------------------------------
00 1.0000 LAKE COUNTIES DOCTORS
01 1.0000 UNITED STATES NEW
02 1.0000 DEADLY BACTERIA
03 1.0000 DISEASE HIT
04 1.0000 COUNTY FAIR
05 1.0000 NEW YORK
. . .
27 0.5000 SOUTH georgia
. . .
30 0.2500 ga IS proved it
31 0.0000 611
----------------------------------------
SELECTED: LAKE COUNTIES DOCTORS
----------------------------------------
00 1.0000 UNITED STATES NEW
01 1.0000 DEADLY BACTERIA
02 1.0000 DISEASE HIT
03 1.0000 COUNTY FAIR
04 1.0000 NEW YORK
05 1.0000 STATE
. . .
30 0.0000 611
----------------------------------------
SELECTED: UNITED STATES NEW
----------------------------------------
00 1.0000 DEADLY BACTERIA
01 1.0000 DISEASE HIT
02 1.0000 COUNTY FAIR
03 1.0000 STATE
04 0.8333 NORTH TEXAS DRILL TEAM CAMP few
05 0.8000 KILLER E COLI DISEASE authorities
. . .
29 0.0000 611
----------------------------------------
SELECTED: DISEASE HIT
----------------------------------------
00 1.0000 DEADLY BACTERIA
01 1.0000 COUNTY FAIR
02 1.0000 STATE
03 0.8333 NORTH TEXAS DRILL TEAM CAMP few
04 0.7500 THREE DEATHS REPORTED tuesday under INVESTIGATION HEALTH
OFFICIALS
05 0.7500 WORST nationally HEALTH OFFICIALS
. . .
16 0.6000 KILLER E COLI disease authorities
. . .
28 0.0000 611
----------------------------------------
SELECTED: DEADLY BACTERIA
----------------------------------------
00 1.0000 COUNTY FAIR
01 1.0000 STATE
02 0.8333 NORTH TEXAS DRILL TEAM CAMP few
03 0.7500 THREE DEATHS REPORTED tuesday under INVESTIGATION HEALTH
OFFICIALS
04 0.7500 WORST nationally HEALTH OFFICIALS
05 0.7500 ORIGIN of OUTBREAK INVESTIGATION
. . .
27 0.0000 611
----------------------------------------
SELECTED: COUNTY FAIR
----------------------------------------
00 1.0000 STATE
01 0.8333 NORTH TEXAS DRILL TEAM CAMP few
02 0.7500 THREE DEATHS REPORTED tuesday under INVESTIGATION HEALTH
OFFICIALS
03 0.7500 WORST nationally HEALTH OFFICIALS
04 0.7500 ORIGIN of OUTBREAK INVESTIGATION
05 0.6667 severe STOMACH ILLNESS
. . .
26 0.0000 611
----------------------------------------
SELECTED: STATE
----------------------------------------
```

FINAL HEADLINE: GEORGIA ELEVEN CHILDREN INFECTED / LAKE COUNTIES DOCTORS / UNITED STATES NEW / DISEASE HIT / DEADLY BACTERIA / COUNTY FAIR / STATE

Fig. 5. Multi-document headline assembly. Candidate phrases are ranked (1st column) by the ratio (2nd column) of keywords (headline-worthy words; displayed in upper case) and total number of words in the phrase. After each phrase selection, the keywords in it become "downgraded" to non-keywords (displayed in lower case), and the remaining phrases are reranked. For example, SOUTH GEORGIA drops from rank 4 to rank 27 after GEORGIA ELEVEN CHILDREN INFECTED has been selected, because GEORGIA loses its value as a keyword. The process stops when the headline length limit has been reached.

ROUGE is computed as follow:

$$ROUGE_n = \frac{\sum_{C \in \{Reference\ \ Summary\}} \sum_{ngram \in C} Count_{match}(ngram)}{\sum_{C \in \{Reference\ \ Summary\}} \sum_{ngram \in C} Count(ngram)},$$

where n stands for the length of the n-gram and $Count_{match}(ngram)$ is the maximum number of n-grams co-occurring in a candidate summary and a reference summary and $Count(ngram)$ is the number of n-grams in the reference summary. It is clear that $ROUGE_n$ is a recall-related metric since the denominator of the equation is the total sum of the number of n-grams occurring at the reference summary side.

The data in Table I indicate that generating English headlines from translations is better than translating Hindi headlines generated from the original documents. This result is statistically significant for unigrams and bigrams with the confidence level of 95% ($\alpha = 0.95$).

Table I. Comparative ROUGE (*n*-Gram Overlap (recall)) Score for Multi-Document Headline Generation

System	Unigrams	Bigrams	Trigrams	4-grams
HH	0.43 (±0.07)	0.16 (±0.06)	0.06 (±0.04)	0.02 (±0.02)
Trans	0.19 (±0.06)	0.02 (±0.02)	0.00 (±0.00)	0.00 (±0.00)
Gen	0.29 (±0.07)	0.07 (±0.04)	0.01 (±0.02)	0.01 (±0.01)
Gen10	0.27 (±0.08)	0.08 (±0.05)	0.03 (±0.03)	0.01 (±0.01)
Gen15	0.32 (±0.07)	0.08 (±0.04)	0.02 (±0.02)	0.01 (±0.01)

HH	overlap among (human) reference translations
Trans	headline generated from Hindi originals, then translated
Gen	headline generated from MT output of full text
Gen10	same as Gen; optimized to achieve an average headlines length of 10
Gen15	same as Gen; optimized to achieve an average headlines length of 15

Notes: Only the first 10 content words in the headlines were considered in the evaluation in order to favor short headlines. Confidence intervals (with $\alpha = 0.95$) were calculated by jackknifing (systematic resampling by selecting 3 out of 4 references for scoring).

Table II. Examples of Generated Single-Document Headlines

Translate then generate	Generate then translate
Defense minister threat of legal action New Delhi/Fernandes/Supremo/food	The legal action threatened Delhi/party Mr./threat/accused
Summit of site with demonstration violence June/police/city/French	Summit conference with/as a countries/ Centre
Trade agreement/pressure/be in business concessions/countries Bangladesh in Dhaka/Asian	The/officers it/the concessions/the/the discussion

Two factors contribute to this phenomenon:

(1) Translation of whole documents increases the sample size and therefore it increases the chances of the translation engine “hitting the right words.” The impact of mistranslation of generated headlines is stronger than of occasional mistranslations in large amounts of text.

(2) The translation engine was designed to translate whole sentences, not phrases. It does not perform as well when the input is a list of separate phrases.

Table II gives a few examples of the generated headlines. Each row in the table corresponds to single document. The left column shows the English headline produced from the translated English document. The right column presents the automatic English translation of the headline generated from the original Hindi document.

5.4 Integration Efficiency

The present Lighthouse/GOSP implementation requires approximately 1 s to compute a cluster headline in a 800 MHz Power PC machine in the interactive settings. Three factors affect the speed of the headline generation: (1) the latency of a Java call to an external command, (2) the time required by GOSP to load and parse the document text, and (3) the time required by GOSP to generate the headline. The first two factors significantly outweigh the last one. Both of them can be eliminated by rewriting the GOSP algorithm in Java and merging it into the Lighthouse framework. Lighthouse already parses the documents and generates document vectors to compute interdocument similarity.

6. TRADEOFFS: MT EARLY OR LATE?

A principal underlying question is architectural: should we translate the Hindi first and then apply the various language modules such as retrieval, clustering, and summarization, or first apply the various modules on the Hindi source and then translate the results? Is it better to develop foreign capabilities for the various technologies (knowing that, given resource limitations, their performance will almost certainly be lower than the English original)? Or is it better to try to perfect and speed up MT, and then focus on improving just the English versions of the other modules? And how should one handle the IR?—certainly one cannot translate all the world's text into English!

Since these questions involve user performance and satisfaction measures, it is impossible to study all the tradeoffs involved in 1 month. However, we can learn some important lessons. Clearly, in a real-world setting, especially when operating over the World Wide Web, one would prefer to minimize translation, since it is an expensive process. However, our results in Section 5.3 suggest that this approach is not ideal within the system: once retrieval is complete, translating and then summarizing the documents can produce significantly better summaries than creating headlines for the Hindi documents and then translating the headlines. Until we have time and the training data required for properly building a Hindi headliner, this will be the case. But for a rapid-deployment scenario one has to follow the first option, using a standard English-trained summarizer.

Similarly, as described in Section 2, we can speed up MT by forgoing some word reordering and hence compromising on the output quality. Since IR operates at the isolated word level, this compromise does not impact IR. We can then introduce this portion of the MT process at a later stage, for example after headline browsing but before paragraph-sized summarization, allowing the user to see an improved version of just the sentences selected for the summary.

As one decomposes each module into submodules, additional opportunities for interweaving them appear. For example, the *N* best alternative translations of a sentence generally are probably not equivalent when it comes to composing a headline from parts of them; GOSP might actually be able to create a more fluent headline from the second- or third-best translation.

As the component technologies mature, and as evaluation metrics for such integrated systems as C*ST*RD are perfected, we can look forward to increasingly refined configurations of modules, parameterized depending on the user's task, the time available for system construction, and the amount of training material at hand. This kind of system integration provides an exciting new arena for research in Human Language Technology.

7. CONCLUSION

During DARPA's June 2003 Surprise Language Exercise, we worked on creating technology solutions for providing access to information in Hindi for English speakers. Our focus in this exercise was twofold. First, we created a MT system that is very successful in translating Hindi documents into English. Second, we built an interactive information access system using various technologies developed at ISI. This system, called C*ST*RD, integrates cross-lingual information retrieval, document organization and visualization, multi-document summarization and headline generation. The goal of C*ST*RD is to provide an English speaking user with information search capabilities in Hindi document collections and minimize the user's exposure to the MT output by means of effective document organization and various text-based and non-text-based summarization approaches.

Most of the document organization technologies employed in C*ST*RD such as clustering and spatial visualization are language-neutral and were easy to adapt for Hindi. Several studies in monolingual English settings showed that these techniques are very effective in helping a user in locating relevant documents among the retrieved document set. We expect this claim to be valid in the cross-lingual settings.

Other C*ST*RD technologies, including IR and document summarization, depend on the target language and either require implementation effort to adapt them to Hindi or can applied to the English output of the MT system. We handled this problem differently for IR and summarization. We implemented English–Hindi query translation and Hindi document analysis such as tokenization and stopword removal. On the other hand, we applied multi-document summarization to the machine-translated output. This strategy proved to be very successful for document headline generation. Our experiments showed that summarizing MT output creates better headlines than translating headlines produced from the original text.

The time limitation of the Surprise Language Exercise precluded us from conducting any extensive study in the same time frame. A subjective cursory evaluation of C*ST*RD indicates that it is indeed good and effective tool for searching Hindi documents for English speakers.

REFERENCES

Allan, J., Callan, J., Croft, B., Ballesteros, L., Broglio, J., Xu, J., and Shu, H. 1997. Inquery at TREC-5. In *Fifth Text REtrieval Conference (TREC-5)* (Gaithersburg, MD, USA). 119–132.

Allan, J., Callan, J., Croft, W. B., Ballesteros, L., Byrd, D., Swan, R., and Xu, J. 1998. Inquery does battle with TREC-6. In *Sixth Text REtrieval Conference (TREC-6)* (Gaithersburg, MD, USA). 169–206.

Allan, J., Leuski, A., Swan, R., and Byrd, D. 2000. Evaluating combinations of ranked lists and visualizations of inter-document similarity. *Information Processing and Management (IPM) 37*, 435–458.

Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*. 194–201.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics 19*, 2, 263–311.

Buckley, C. and Salton, G. 1995. Optimization of relevance feedback weights. In *Proceedings of ACM SIGIR* (Seattle, Washington, USA). 351–357.

Croft, W. B. 1978. Organising and searching large files of documents. Ph.D. thesis, University of Cambridge.

Cutting, D. R., Karger, D. R., and Pedersen, J. O. 1993. Constant interaction-time Scatter/Gather browsing of very large document collections. In *Proceedings of ACM SIGIR*. 126–134.

Cutting, D. R., Pedersen, J. O., Karger, D. R., and Tukey, J. W. 1992. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of ACM SIGIR* (Copenhagen, Denmark). 318–329.

Dunning, T. E. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics 19,* 1, 61–74.

Edmundson, H. P. 1969. New methods in automatic extraction. *Journal of the ACM 16,* 2, 264–285.

Fruchterman, T. M. J. and Reingold, E. M. 1991. Graph drawing by force-directed placement. *Software–Practice and Experience 21,* 11, 1129–1164.

Germann, U. 2001. Building a statistical machine translation system from scratch: How much bang for the buck canwe expect? In *ACL 2001 Workshop on Data-Driven Machine Translation* (Toulouse).

Germann, U. 2003. Greedy decoding for statistical machine translation in almost linear time. In *HLT-NAACL 2003: Main Proceedings*, M. Hearst and M. Ostendorf, Eds. Association for Computational Linguistics, Edmonton, AB, Canada, 72–79.

Getty. Getty Thesaurus of Geographic Names. Available at http://www.getty.edu/research/conducting_research/vocabularies/tgn/ as of September 2003.

Gillam, R. 1999. Finding Text Boundaries in Java. Available at http://www-106.ibm.com/developerworks/java/library/j-boundaries/boundaries.html as of March 2004.

Goldstein, J., Kantrowitz, M., Mittal, V. O., and Carbonell, J. G. 1999. Summarizing text documents: Sentence selection and evaluation metrics. In *Research and Development in Information Retrieval*. 121–128.

Hearst, M. A. and Pedersen, J. O. 1996. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of ACM SIGIR* (Zurich, Switzerland). 76–84.

Jin, R. and Hauptmann, A. 2001. Headline generation using a training corpus. In *Proceedings of the Second International Conference on Intelligent Text Processing and Computational Linguistics (CICLing01)*. Lecture Notes in Computer Science. Springer, Mexico City, Mexico, 208–215.

Koenemann, J. and Belkin, N. J. 1996. A case for interaction: A study of interactive information retrieval behavior and effectivness. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada). 205–212.

Leuski, A. 2000. Relevance and reinforcement in interactive browsing. In *Proceedings of Ninth International Conference on Information and Knowledge Management (CIKM'00),* A. Agah, J. Callan, and E. Rundensteiner, Eds. ACM Press, McLean, Virginia, USA, 119–126.

Leuski, A. 2001a. Evaluating document clustering for interactive information retrieval. In *Proceedings of Tenth International Conference on Information and Knowledge Management (CIKM'01)*, H. Paques, L. Liu, and D. Grossman, Eds. ACM Press, Atlanta, Georgia, USA, 41–48.

Leuski, A. 2001b. Interactive Information Organization: Techniques and Evaluation. Ph.D. thesis, University of Massachusetts at Amherst.

Leuski, A. and Allan, J. 2003. Interactive information retrieval using clustering and spatial proximity. *User Modeling and User Adapted Interaction (UMUAI)*. In Press.

Leuski, A., Lin, C.-Y., and Hovy, E. 2003. iNeATS: Interactive multi-document summarization. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)* (Sapporo, Japan). 125–128.

Lin, C.-Y. and Hovy, E. 1997. Identifying topics by position. In *Proceedings of the 5th Conference on Applied Natural Language Processing* (Washington, DC).

Lin, C.-Y. and Hovy, E. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)* (Saarbrücken, Germany).

LIN, C.-Y. AND HOVY, E. 2002. From single to multi-document summarization: a prototype system and it evaluation. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL-02)* (Philadelphia, PA, USA).

LIN, C.-Y. AND HOVY, E. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL 2003: Main Proceedings*, M. Hearst and M. Ostendorf, Eds. Association for Computational Linguistics, Edmonton, AB, Canada, 150–157.

Lucene 2003. Lucene Search Engine. Available at http://jakarta.apache.org/lucene/ as of January 2003.

MCKEOWN, K. R., BARZILAY, R., EVANS, D., HATZIVASSILOGLOU, V., SCHIFFMAN, B., AND TEUFEL, S. 2001. Columbia multi-document summarization: Approach and evaluation. In *Proceedings of the Workshop on Text Summarization, ACM SIGIR Conference 2001*. DARPA/NIST, Document Understanding Conference.

MIRKIN, B. 1996. *Mathematical Classification and Clustering*. Kluwer, Boston.

OARD, D. W. AND OCH, F. J. 2003. Rapid-response machine translation for unexpected languages. In *Proceedings of the MT Summit IX* (New Orleans, LA).

OCH, F. J. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)* (Sapporo, Japan).

OCH, F. J. AND NEY, H. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)* (Philadelphia, PA).

OCH, F. J., TILLMANN, C., AND NEY, H. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora* (University of Maryland, College Park, MD). 20–28.

OVER, P. 2001. Introduction to duc-2001: an intrinsic evaluation of generic news text summarization systems. In *Proceedings of the Workshop on Text Summarization, ACM SIGIR Conference 2001*. DARPA/NIST, Document Understanding Conference.

PAPINENI, K. A., ROUKOS, S., WARD, T., AND ZHU, W.-J. 2001. Bleu: a method for automatic evaluation of machine translation. Tech. Rep. RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY.

PIRKOLA, A. 1998. The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*. 55–63.

PORTER, M. F. 1980. An algorithm for suffix stripping. *Program 14*, 3, 130–137.

ROBERTSON, S. E., WALKER, S., JONES, S., HANCOCK-BEAULIEU, M. M., AND GATFORD, M. 1995. Okapi at TREC-3. In *Third Text REtrieval Conference (TREC-3)*, D. Harman and E. Voorhees, Eds. NIST, Gaithersburg, Maryland, USA.

ROCCHIO, JR., J. J. 1971. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, G. Salton, Ed. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 313–323.

SALTON, G. 1989. *Automatic Text Processing*. Addison-Wesley.

UEFFING, N., OCH, F. J., AND NEY, H. 2002. Generation of word graphs in statistical machine translation. In *Proceedings Conference on Empirical Methods for Natural Language Processing* (Philadelphia, PA). 156–163.

VAN RIJSBERGEN, C. J. 1979. *Information Retrieval*, 2nd ed. Butterworths, London.

WILLETT, P. 1988. Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management 24*, 5, 577–597.

ZAJIC, D., DORR, B., AND SCHWARTZ, R. 2002. Automatic headline generation for newspaper stories. In *Proceedings of the ACL-02 Workshop on Text Summarization* (Philadelphia, PA).

ZHOU, L. AND HOVY, E. 2003. Headline summarization at ISI. In *Document Understanding Conference (DUC-03)* (Edmonton, AB, Canada).

Received August 15, 2003; revised September 29, 2003; accepted October 24, 2003