The Smart/Empire TIPSTER IR System

Chris Buckley, Janet Walz
Sabir Research, Gaithersburg, MD
chrisb,walz@sabir.com

Claire Cardie, Scott Mardis, Mandar Mitra, David Pierce, Kiri Wagstaff
Department of Computer Science
Cornell University, Ithaca, NY 14853
cardie,mardis,mitra,pierce,wkiri@cs.cornell.edu

1 INTRODUCTION

The primary goal of the Cornell/Sabir TIPSTER Phase III project is to develop techniques to improve the end-user efficiency of information retrieval (IR) systems. We have focused our investigations in four related research areas:

1. **High Precision Information Retrieval.** The goal of our research in this area is to increase the accuracy of the set of documents given to the user.
2. **Near-Duplicate Detection.** The goal of our work in near-duplicate detection is to develop methods for delineating or removing from the set of retrieved documents any information that the user has already seen.
3. **Context-Dependent Document Summarization.** The goal of our research in this area is to provide for each document a short summary that includes only those portions of the document relevant to the query.
4. **Context-Dependent Multi–Document Summarization.** The goal of our research in this area is to provide a short summary for an entire group of related documents that includes only query-related portions.

Taken as a whole, our research aims to increase end-user efficiency in each of the above tasks by reducing the amount of text that the user must peruse in order to get the desired useful information.

We attack each task through a combination of statistical and linguistic approaches. The proposed statistical approaches extend existing methods in IR by performing statistical computations within the context of another query or document. The proposed linguistic approaches build on existing work in information extraction and rely on a new technique for trainable partial parsing. In short, our integrated approach uses both statistical and linguistic sources to identify selected relationships among important terms in a query or text. The relationships are encoded as TIPSTER annotations [7]. We then use the extracted relationships: (1) to discard or reorder retrieved texts (for high-precision text retrieval); (2) to locate redundant information (for near-duplicate document detection); and (3) to generate coherent synopses (for context-dependent text summarization).

An end-user scenario that takes advantage of the efficiency opportunities offered by our research might proceed as follows:

1. The user submits a natural language query to the retrieval system, asking for a high-precision search. This search will attempt to retrieve fewer documents than a normal search, but at a higher quality, so many fewer non-useful documents will need to be examined.

2. The documents in the result set will be clustered so that closely related documents are grouped.

- Duplicate documents will be clearly marked so the user will not have to look at them at all.
- Near-duplicate documents will also be clearly marked. When the user examines a document marked as a near-duplicate to a document previously examined, the new material in this document is emphasized in color so that it can be quickly perused, while the duplicate material can be ignored.

3. Long documents can be automatically summarized, within the context of the query, so that perhaps only 20% of the document will be presented. This 20% summary

would include the material that made the system decide the document was useful, as well as other material designed to set the context for the query-related material.

4. If the user wishes, an entire cluster of documents can be summarized. The user can then decide whether to look at any of the individual documents. This multi-document summary will once again be query-related.

One key result of our TIPSTER efforts is the development of TRUESmart, a Toolbox for Research in User Efficiency. TRUESmart is a set of tools and data supporting researchers in the development of methods for improving user efficiency for state-of-the-art information retrieval systems. TRUESmart allows the integration of system components for high-precision retrieval, duplicate detection, and context-dependent summarization; it includes a simple graphical user interface (GUI) that supports each of these tasks in the context of the end-user scenario described above. In addition, TRUESmart aids system evaluation and analysis by highlighting important term relationships identified by the underlying statistical and linguistic language processing algorithms.

The rest of the paper presents TRUESmart and its underlying IR and NLP components. Section 2 first provides an overview of the Smart IR system and the Empire Natural Language Processing (NLP) system. Section 3 describes the TRUESmart toolbox. To date, we have used TRUESmart to support our work in high-precision retrieval and context-dependent document summarization. We describe our results in these areas in Sections 4–5 using the TRUESmart interface to illustrate the algorithms developed and their contribution to the end-user scenario described above. Section 6 summarizes our work in duplicate detection and describes how the TRUESmart interface will easily be extended to support this task and include linguistic term relationships in addition to statistical term relationships. We conclude with a summary of the potential advantages of our overall approach.

2 THE UNDERLYING SYSTEMS: SMART AND EMPIRE

The two main foundations of our research are the Smart system for Information Retrieval and the Empire system for Natural Language Processing. Both are large systems running in the UNIX environment at Cornell University.

2.1 Smart

Smart Version 13 is the latest in a long line of experimental information retrieval systems, dating back over 30 years, developed under the guidance of G. Salton. The new version is approximately 50,000 lines of C code and documentation.

Smart Version 13 offers a basic framework for investigations of the vector space and related models of information retrieval. Documents are fully automatically indexed, with each document representation being a weighted vector of concepts, the weight indicating the importance of a concept to that particular document. The document representatives are stored on disk as an inverted file. Natural language queries undergo the same indexing process. The query representative vector is then compared with the indexed document representatives to arrive at a similarity and the documents are then fully ranked by similarity.

Smart Version 13 is highly flexible (i.e., its algorithms can be easily adapted for a variety of IR tasks) and very fast, thus providing an ideal platform for information retrieval experimentation. Documents are indexed at a rate of almost two gigabytes an hour, on systems currently costing under $5,000 (for example, a dual Pentium Pro 200 Mhz with 512 megabytes memory and disk). Retrieval speed is similarly fast, with basic simple searches taking much less than a second a query.

2.2 The Empire System: A Trainable Partial Parser

Stated simply, the goal of the natural language processing (NLP) component for the selected text retrieval tasks is to locate linguistic relationships between query terms. For this, we have developed **Empire**[1], a trainable partial parser. The remainder of this section describes the assumptions of our approach and the general architecture of the system.

For the TIPSTER project, we are investigating the role of linguistic relationships in information retrieval tasks. A linguistic relationship between two terms is any relationship that can be determined through syntactic or semantic interpretation of the text that contains the terms. We are focusing on three classes of linguistic relationships that we believe will aid the information retrieval tasks:

1. **noun phrase relationships**. E.g., determine whether two query terms appear in the same (simple) noun phrase; find all places where a query term appears as the head of a noun phrase.

[1] The name refers to our focus on empirical methods for development and evaluation of the system.

2. **subject-verb-object relationships**, including the identification of subjects and objects in gap constructions. These relationships help to identify the functional structure of a sentence, i.e., who did what to whom. Once identified, Smart can assign higher weights to query terms that appear in these topic-indicating verb, object, and especially subject positions.

3. **noun phrase coreference**. Coreference resolution is the identification of all strings in a document that refer to the same entity. Noun phrase coreference will allow Smart to create more coherent summaries, e.g., by replacing pronouns with their referents as identified by Empire. In addition, Smart can use coreference relationships to modify its term weighting function to reflect the implied equality between all elements of a noun phrase equivalence class.

Once identified, the linguistic relationships can be employed in a number of ways to improve the efficiency of end-users: they can be used (1) to prefer the retrieval of documents that also exhibit the relationships; (2) to indicate the presence of redundant information; or (3) to establish the necessary context in automatically generated summaries. Our approach to locating linguistic relationships is based on the following assumptions:

- *The NLP system need recognize only those relationships that are useful for the specific text retrieval application.* There may be no need for full-blown syntactic and semantic analysis of queries and documents.

- *The NLP system must recognize these relationships both quickly and accurately.* The speed requirement argues for a shallow linguistic analysis; the accuracy requirement argues for algorithms that focus on precision rather than recall.

- *The NLP component need only provide a comparative linguistic analysis between a document and a query.* This should simplify the NLP task because individual documents do not have to be analyzed in isolation, but only relative to the query.

Given these assumptions, we have developed Empire, a fast, trainable, precision-based partial parser. As a partial parser, Empire performs only shallow syntactic analysis of input texts. Like many partial parsers and NLP systems for information extraction (e.g., Hobbs *et al.* [9]), Empire relies primarily on finite-state technology [16] to recognize all syntactic and semantic entities as well as their relationships to one another. Parsing proceeds in stages — the initial stages identify relatively simple constituents: simple noun phrases, some prepositional phrases, verb groups, and clauses. All linguistic relationships that require higher-level attachment decisions are identified in subsequent stages and rely on output from earlier stages. Our use of finite-state transducers for partial parsing is most similar to the work of Abney [1], who employs a series of cascaded finite-state machines to build up an increasingly complex linguistic analysis of an incoming sentence.

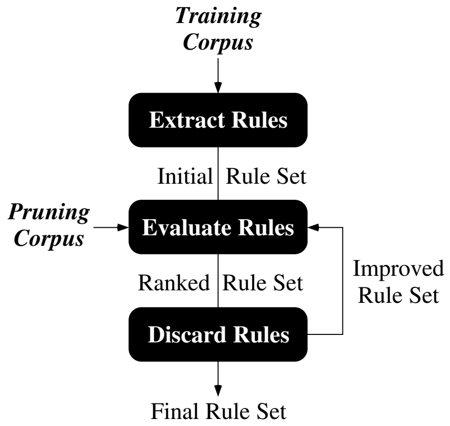

Figure 1: Error-Driven Pruning of Treebank Grammars

Unlike most work in this area, however, we do not use hand-crafted patterns to drive the linguistic analysis. Instead, we rely on corpus-based learning algorithms to acquire the grammars necessary for driving each level of linguistic relationship identification. In particular, we have developed a very simple, yet effective technique for automating the acquisition of grammars through *error-driven pruning of treebank grammars* [6]. As shown in Figure 1, the method first extracts an initial grammar from a "treebank" corpus, i.e., a corpus that has been annotated with respect to the linguistic relationship of interest. Consider the base noun phrase relationship — the identification of simple, non-recursive noun phrases. Accurate identification of base noun phrases is a critical component of any partial parser; in addition, Smart relies on base NPs as its primary source of linguistic phrase information. To extract a grammar for base noun phrase identification, we tag the training text with a part-of-speech tagger (we use Mitre's version of Brill's tagger [3]) and then extract as an NP rule every unique part-of-speech sequence that covers a base NP annotation.

Next, the grammar is improved by discarding rules that obtain a low precision-based "benefit" score when applied to a held out portion of the training corpus, the pruning corpus. The resulting "grammar" can then be used to identify base NPs in a novel text as follows:

1. Run all lower-level annotators. For base NPs, for example, run the part-of-speech annotator.

2. Proceed through the tagged text from left to right, at each point matching the rules against the remaining input. For base NP recognition, match the NP rules against the remaining part-of-speech tags in the text.

3. If there are multiple rules that match beginning at tag or token t_i, use the longest matching rule R. Begin the matching process anew at the token that follows the last NP.

2.2.1 Empire Evaluation

Using this simple grammar extraction and pruning algorithm with the naive longest-match heuristic for applying rules to incoming text, the learned grammars are shown to perform very well for base noun phrase identification. A detailed description of the base noun phrase finder and its evaluation can be found in Cardie and Pierce [6]. In summary, however, we have evaluated the approach on two base NP corpora derived from the Penn Treebank [11]. The algorithm achieves 91% precision and recall on base NPs that correspond directly to non-recursive noun phrases in the treebank; it achieves 94% precision and recall on slightly less complicated noun phrases.[2]

We are currently investigating the use of error-driven grammar pruning to infer the grammars for all phases of partial parsing and the associated linguistic relationship identification. Initial results on verb-object recognition show 72% precision when tested on a corpus derived from the Penn Treebank. Analysis of the results indicates that our context-free approach, which worked very well for noun phrase recognition, does not yield sufficient accuracy for verb-object recognition. As a result, we have used standard machine learning algorithms (i.e., k-nearest neighbor and memory-based learning using the value-difference metric) to classify each proposed verb-object bracketing as either correct or incorrect given a 2-word window surrounding the bracketing. In preliminary experiments, the machine learning algorithm obtains 84% generalization accuracy. If we discard all bracketings it classifies as incorrect, overall precision for verb-object recognition increases from 72% to over 80%. The next section outlines our general approach for using learning algorithms in conjunction with the Empire system.

[2]This corpus further simplifies some of the the Treebank base NPs by removing ambiguities that we expect other components of our NLP system to handle, including: conjunctions, NPs with leading and trailing adverbs and verbs, and NPs that contain prepositions.

2.2.2 The Role of Machine Learning Algorithms

As noted above, Empire's finite-state partial parsing methods may not be adequate for identifying some linguistic relationships. At a minimum, many linguistic relationships are better identified by taking additional context into account. In these circumstances, we propose the use of corpus-based machine learning techniques — both as a systematic means for correcting errors (as done for verb-object recognition above) and for learning to identify linguistic relationships that are more complex than those covered by the finite-state methods above.

In particular, we have employed the **Kenmore** knowledge acquisition framework for NLP systems [4, 5]. Kenmore relies on three major components. First, it requires an **annotated training corpus**, i.e., a collection of online documents, that has been annotated with the necessary bracketing information. Second, it requires a **robust sentence analyzer**, or parser. For this, we use the Empire partial parser. Finally, the framework requires an **inductive learning algorithm**. Although any inductive learning algorithm can be used, we have successfully used case-based learning (CBL) algorithms for a number of natural language learning problems.

There are two phases to the framework: (1) a partially automated training phase, or **acquisition phase**, in which a particular linguistic relationship is learned, and (2) an **application phase**, in which the heuristics learned during training can be used to identify the linguistic relationship in novel texts. More specifically, the goal of Kenmore's training phase (see Figure 2) is to create a *case base*, or memory, of linguistic relationship decisions. To do this, the system randomly selects a set of training sentences from the annotated corpus. Next, the sentence analyzer processes the selected training sentences, creating one case for every instance of the linguistic relationship that occurs. As shown in Figure 2, each case has two parts. The *context* portion of the case encodes the context in which the linguistic relationship was encountered — this is essentially a representation of some or all of the constituents in the neighborhood of the linguistic relationship as denoted in the flat syntactic analysis produced by the parser. The *solution* portion of the case describes how the linguistic relationship was resolved in the current example. In the training phase, this solution information is extracted directly from the annotated corpus. As the cases are created, they are stored in the case base.

After training, the NLP system uses the case base *without the annotated corpus* to identify new occurrences of the linguistic relationship in novel sentences. Given a sentence as input, the sentence analyzer processes the sentence and creates a problem case, automatically filling in its context portion based on the constituents appearing the

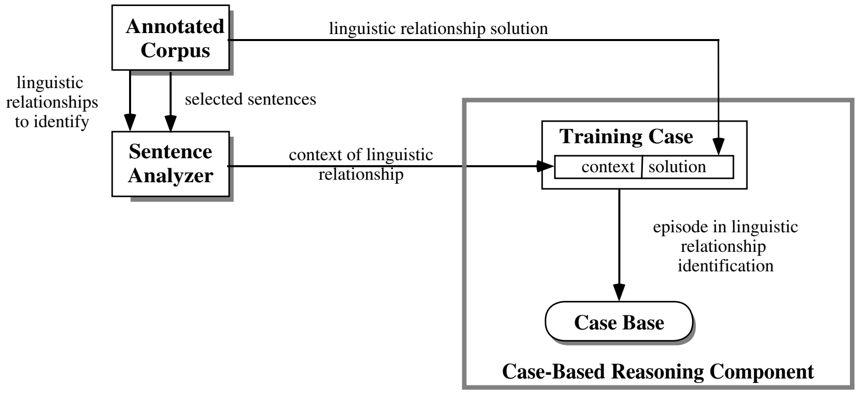

Figure 2: Kenmore Training/Acquisition Phase.

sentence. To determine whether the linguistic relationship holds, Kenmore next compares the problem case to each case in the case base, retrieves the most similar training case, and returns the decision as indicated in the solution part of the case. The solution information lets Empire decide whether the desired relationship exists in the current sentence.

In previous work, we have used Kenmore for part-of-speech tagging, semantic feature tagging, information extraction concept acquisition, and relative pronoun resolution [5]. We expect that this approach will be necessary for coreference resolution, for some types of subject-object identification, and for handling gap constructs (i.e., for determining that "boy" is the subject of "ate" as well as the object of "saw" in "Billy saw the boy that ate the candy"). It is also the approach used to learn the verb-object correction "heuristics" described in the last section.

2.2.3 Coreference Resolution

The final class of linguistic relationship is **noun phrase coreference** — for every entity in a text, the NLP system must locate all of the expressions or phrases that refer to it. As an example, consider the following: "Bill *Clinton*, current *president* of the United States, left Washington Monday morning for China. *He* will return in two weeks." In this excerpt, the phrases "Bill Clinton," "current president (of the United States)," and "he" refer to the same entity. Smart can use this coreference information to treat the associated terms as equivalents. For example, it can assume that all items in the class are present whenever one appears. In conjunction with coreference resolution, we are also investigating the usefulness of providing the IR system with canonicalized noun phrase forms that make use of term invariants identified during coreference.

To date, we have implemented two simple algorithms for coreference resolution to use purely as baselines. Both operate only on base noun phrases as identified by Empire's base NP finder. The first heuristic assumes that two noun phrases are coreferent if they share any terms in common. The second assumes that two noun phrases are coreferent if they have the same head. Both obtained higher scores than expected when tested on the MUC6 coreference data set. The head noun heuristic achieved 42% recall and 51% precision; the overlapping terms heuristic achieved 41% recall and precision.

2.2.4 Empire Annotators

All relationships identified by Empire are made available to Smart in the form of TIPSTER annotations. We currently have the following annotators in operation:

- tokenizer: identifies tokens, punctuation, etc.
- sentence finder: based on Penn's maximum entropy algorithm [15].
- baseNPs: identifies non-recursive noun phrases.
- verb-object: identifies verb-object pairs, either by bracketing the verb group and entire direct object phrase or by noting just the heads of each.
- head noun coreference heuristic: identifies coreferent NPs.
- overlapping terms coreference heuristic: identifies coreferent NPs.

The tokenizer is written in C. The sentence finder is written in Java. All other annotators are implemented in Lucid/Liquid Common Lisp.

3 TRUESmart

To support our research in user-efficient information retrieval, we have developed TRUESmart, a Toolbox for Research in User Efficiency. As noted above, TRUESmart allows the integration, evaluation, and analysis of IR and NLP algorithms for high-precision searches, context-dependent summarization, and duplicate detection. TRUESmart provides three classes of resources that are necessary for effective research in the above areas:

1. **Testbed Collections**, including test queries and correct answers
2. **Automatic Evaluation Tools**, to measure overall how an approach does on a collection.
3. **Failure Analysis Tools**, to help the researcher investigate in depth what has happened.

These tools are, to a large extent, independent of the actual research being done. However, they are just as vital for good research as the research algorithms themselves.

3.1 TRUESmart Collections

The testbed collections organized for TRUESmart are all based on TREC [19] and SUMMAC [10], the large evaluation workshops run by NIST and DARPA respectively. TREC provides a number of document collections ranging up to 500,000 documents in size, along with queries and relevance judgements that tell whether a document is relevant to a particular query.

Evaluation of our high-precision research can be done directly using the TREC collections. The TREC documents, queries, and relevance judgements are sufficient to evaluate whether particular high-precision algorithms do better than others.

For summarization research, however, a different testbed is needed. The SUMMAC workshop evaluated summaries of documents. The major evaluation measured whether human judges were able to judge relevance of entire documents just from the summaries. While very valuable in giving a one-time absolute measure of how well summarization algorithms are doing, human-dependent evaluations are infeasible for a research group to perform on ongoing research since different human assessors are required whenever a given document or summary is judged.

Our summarization testbed is based on the SUMMAC QandA evaluation. Given a set of questions about a document, and a key describing the locations in the document where those questions are answered, the goal is to evaluate how well an extraction-based summary of that document answers the questions. So the TRUESmart summarization testbed consists of

- A small number of queries
- A small number of relevant documents per query
- A set of questions for each query
- Locations in the relevant documents where each question is answered.

Objective evaluation of near-duplicate information detection is difficult. As part of our efforts in this area, we have constructed a small set (50 pairs) of near-duplicate documents of newswire articles. These pairs were deliberately chosen to encompass a range of duplication amounts; we include 5 pairs at cosine similarity .95, 5 pairs at .90, and 10 pairs at each of .85, .80, .75, and .70. In addition, they have been categorized as to exactly what the relationship between the pairs is. For example, some pairs are slight rewrites by the same author, some are followup articles, and some are two articles on the same subject by different authors. We also have queries that will retrieve both of these pairs among the top documents. These articles are tagged: corresponding sections of text from each document pair are marked as identical, semantically equivalent, or different.

Preparing a testbed for multi-document summarization is even more difficult. We have not done this as yet, but our initial approach will take as a seed the QandA evaluation test collections described above. This gives us a query and a set of relevant documents with known answers to a set of common questions. Evaluation can be done by performing a multi-document summarization on a subgroup of this set of relevant documents. The final summary can be evaluated based upon how many questions are answered (a question is answered by a text excerpt in the summary if the excerpt in the corresponding original document was marked as answering the question), and how many questions are answered more than once. If too many questions are answered more than once, then the duplicate detection algorithms may not be working optimally. If too few questions are answered at all, then the summarization algorithms may be at fault. The evaluation numbers produced by the final summary can be compared against the average evaluation numbers for the documents in the group.

3.2 TRUESmart Evaluation

Automatic evaluation of research algorithms is critical for rapid progress in all of these areas. Manual evaluation is valuable, but impractical when trying to distinguish between small variations of a research group's algorithms.

3.2.1 Trec_eval

Automatic evaluation of straight information retrieval tasks is not new. In particular, we have provided the "trec_eval" program to the TREC community to evaluate retrieval in the TREC environment. It will also be an evaluation component in the TRUESmart ToolBox. The trec_eval measures are described in the TREC-4 workshop proceedings [8].

3.2.2 Summ_eval

The QandA evaluation of SUMMAC is very close to being automatic once questions and keys are created. For SUMMAC, the human assessors still judge whether or not a given summary answers the questions. Indeed, for non-extraction-based summaries, this is required. But for evaluation of extraction-based summarization (where the summaries contain clauses, sentences, or paragraphs of the original document), an automatic approximation of the assessor task is possible. This enables a research group to fairly evaluate and compare multiple summaries of the same document, with no additional manual effort after the initial key is determined. Thus we have written the "summ_eval" evaluator. This algorithm for the automatic evaluation of summaries:

1. Automatically finds the spans of the text of the original document that were given as answers in the keys.
2. Automatically finds the spans of the text of the original document that appeared in a summarization of the document.
3. Computes various measures of overlap between the summarization spans and the answer spans.

The effectiveness of two summarization algorithms can be automatically compared by comparing these overlap measures.

We ran summ_eval on the summaries produced by the systems of the SUMMAC workshop. The comparative ranking of systems using summ_eval is very close to the (presumably) optimal rankings using human assessors. This strongly suggests that automatic scoring of summ_eval can be useful for evaluation in circumstances where human scoring is not available

3.2.3 Dup_eval

"Dup_eval" uses the same algorithms as summ_eval to measure how well an algorithm can detect whether one document contains information that is duplicated in another. The key (correct answer) for one document out of a pair will give the spans of text in that document that are duplicated in the other, at three different levels of duplication: exact, semantically equivalent, and contained in. The duplicate detection algorithm being evaluated will come up with similar spans. Dup_eval measures the overlap between the these sets of spans.

3.3 TRUESmart GUI

Automatic evaluation is only the beginning of the research process. Once evaluation pinpoints the failures and successes of a particular algorithm, analysis of these failures must be done in order to improve the algorithm. This analysis is often time-consuming and painful. This motivates the implementation of the TRUESmart GUI. This GUI is not aimed at being a prototype of a user efficiency GUI. Instead, it offers a basic end-user interface while giving the researcher the ability to explore the underlying causes of particular algorithm behavior.

Figure 3 shows the basic TRUESmart GUI as used to support high-precision retrieval and context-dependent summarization. The user begins by typing a query into the text input box in the middle, left frame. The sample query is TREC query number 151: "The document will provide information on jail and prison overcrowding and how inmates are forced to cope with those conditions; or it will reveal plans to relieve the overcrowded condition." Clicking the *SubmitQ* button initiates the search. Clicking the *NewQ* button allows the submission of a new query.[3] Once the query is submitted, Smart initiates a global search in order to quickly obtain an initial set of documents for the user. The document number, similarity ranking, similarity score, source, date, and title of the top 20 retrieved documents are displayed in the upper left frame of the GUI. Clicking on any document will cause its query-dependent summary to be displayed in the large frame on the right. In Figure 3, the summary of the seventh document is displayed. In this run, we have set Smart's target summary length to 25% and asked for sentence- (rather than paragraph-) based summaries. Matching query terms are highlighted throughout the summary although they are not visible in the screen dump. The left, bottom-most frame of the interface lists the most important query terms (e.g., prison, jail,

[3] The "ModQ" and "Mod vec" buttons allow the user to modify the query and modify the query vector, respectively. Neither will be discussed further here.

inmat(e), overcrowd) and their associated weights (e.g., 4.69, 5.18, 7.17, 12.54).

After the initial display of the top-ranked documents, Smart begins a local search in the background: each individual document is reparsed and matched once again against the query to see if it satisfies the particular high-precision restriction criteria being investigated. If it doesn't the document is removed from the retrieved set; otherwise, the document remains in the final retrieved set with a score that combines the global and local score. In addition, the user can supply relevance judgements on any document by clicking *Rel* (relevant), *NRel* (not relevant), or *PRel* (probably relevant). Smart uses these judgements as feedback, updating the ranking after every 5 judgements by adding new documents and removing those already judged from the list of retrieved texts. Figure 4 shows the state of the session after a number of relevance judgements have been made and new documents have been added to the top 20.

The interface, while basic, is valuable in its own right. It was successfully used for the Cornell/SabIR experiments in the TREC 7 High-Precision track. In this task, users were asked to find 15 relevant documents within 5 minutes for each of 50 queries. This was a true test of user efficiency; and Cornell/SabIR did very well.

The most important use of the GUI, though, is to explore what is happening underneath the surface, in order to aid the researcher. Operating on either a single document or a cluster of documents, the researcher can request several different views. The two main paradigms are: (1) the document map view, which visually indicates the relationships between parts of the selected document(s); and (2) the document annotation view, which displays any subset of the available annotations for the selected document(s). Neither view is shown in Figures 3 and 4.

The document annotation view, in particular, is extremely flexible. The interface allows the user to run any of the available annotators on a document (or document set). Each annotator returns the text(s) and the set of annotations computed for the text(s). The GUI, in turn, displays the text with the spans of each annotation type highlighted in a different color. Optionally, the values of each annotation can be displayed in a separate window. Thus, for instance, a document may be returned with one annotation type giving the spans of a document summary, and other annotation types giving the spans of an ideal summary. The researcher can then immediately see what the problems are with the document summary.

There is no limit to the number of possible annotators that can be displayed. Annotators implemented or planned include:

- Query term matches (with values in separate window).
- Statistical and/or linguistic phrase matches.
- Summary vs. model summary.
- Summary vs. QandA answers.
- Two documents concatenated with duplicate information of the second annotated in the first.
- Coreferent noun phrases.
- Subject, verb, or object term matches.
- Verb-object, subject-verb, and subject-object term matches.
- Subjects or objects of gap constructions annotated with the inferred filler if it matches an important term.

Analyzing the role of linguistic relationships in the IR tasks amounts to requesting the display of some or all of the NLP annotators. For example, the user can request to see linguistic phrase matches as well as statistical phrase matches. In the example from Figure 3, the resulting annotated summary would show "27 **inmates**" and "Latino **inmates**" as matches of the query term "**inmates**" because all instances of "inmates" appear as head nouns. Similarly, it would show a linguistic phrase match between "**jail overcrowding**" (paragraph 5 of the summary) and "**jail** and prison **overcrowding**" (in the query) for the same reason. When the output of the linguistic phrase annotator is requested, the lower left frame that lists query terms and weights is updated to include the linguistic phrases from the query and their corresponding weights.

Alternatively, one might want to analyze the role of the "subject" annotator. In the running example, this would modify the summary window to show matches that involve terms appearing as the subject of a sentence or clause. For example, all of the following occurrences of "inmates" would be marked as subject matches with the "inmates" query term, which also appears in the subject position ("inmates are forced"): "**inmates** were injured" (paragraph 1), "**inmates** broke out" (paragraph 2), "**inmates** refused" (paragraph 2), "**inmates** are confined" (paragraph 3), etc. Smart can give extra weight to these "subject" term matches since entities that appear in this syntactic position are often central topic terms. The interface helps the developer to quickly locate and determine the correctness of subject matches. As an aside, if the "subject gap construction" annotator were requested, "**inmates**" would be filled in as the implicit subject of "return" in paragraph 2 and would be marked as a query term match.

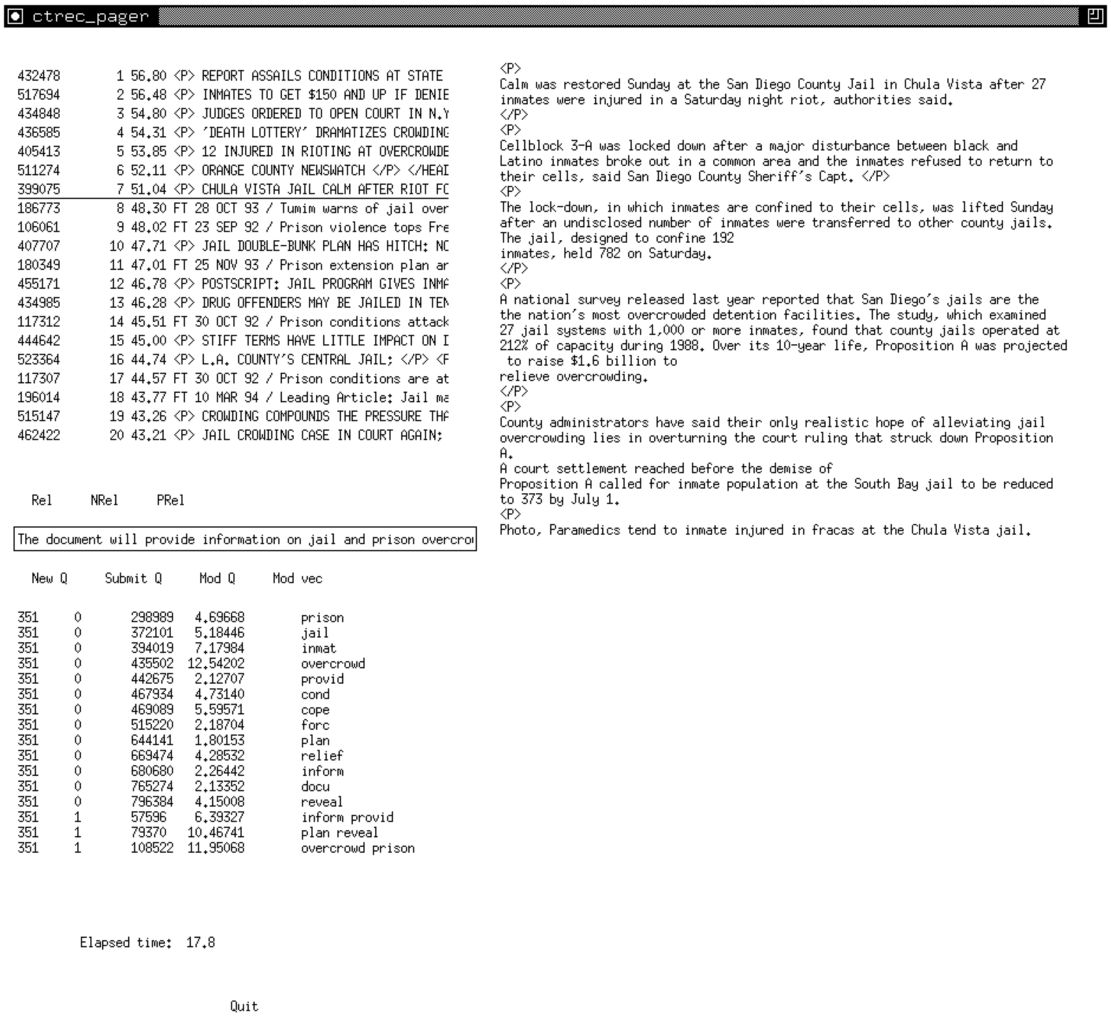

Figure 3: TRUESmart GUI After Initial Query. Note that (other than the text input box) no frame borders, scrolling options, and or button borders are visible in this screendump.

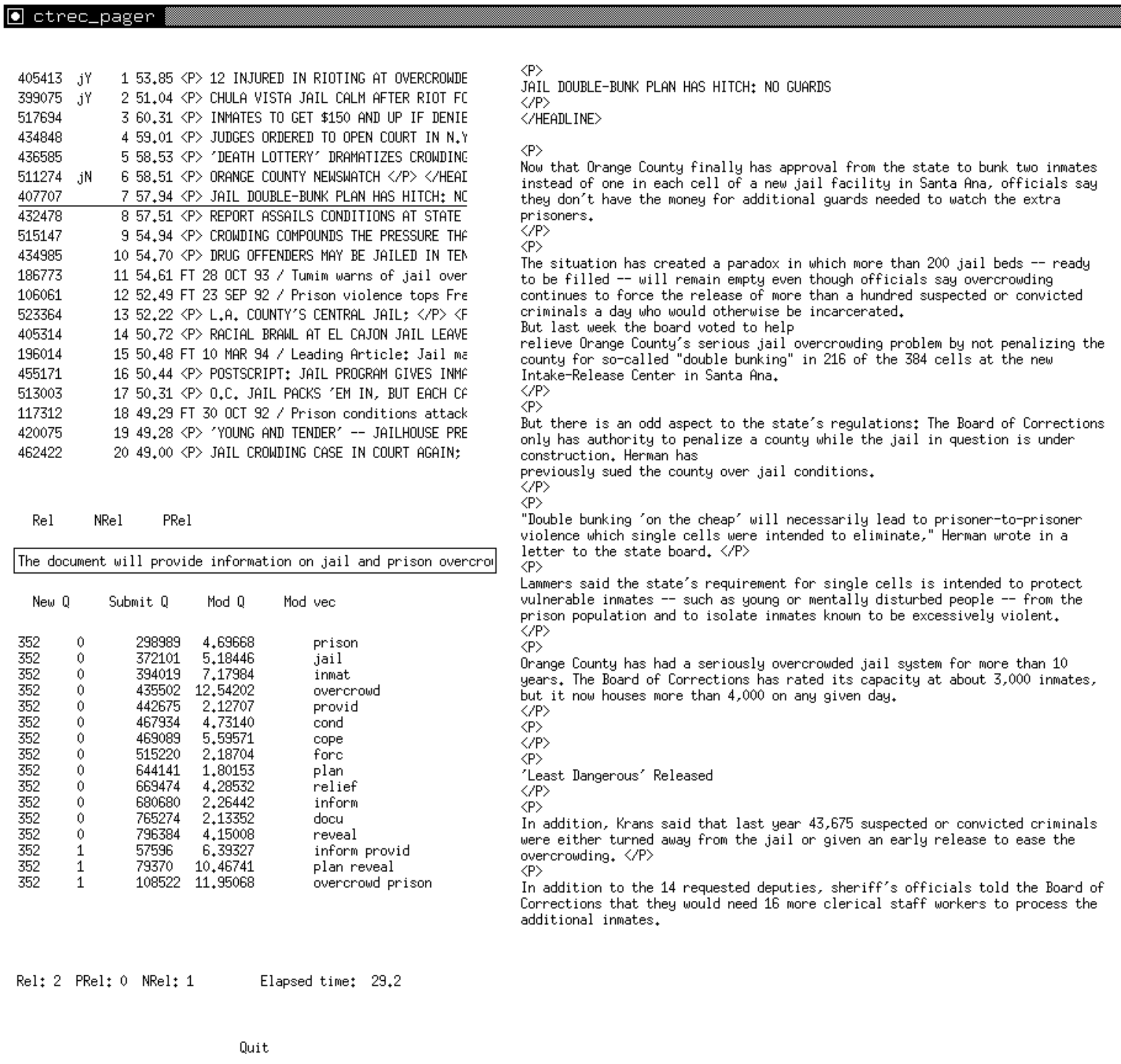

Figure 4: TRUESmart GUI After Relevance Judgements.

Finally, the role of coreference resolution might also be analyzed by requesting to see the output of the coreference annotator. In response to this request, the document text window would then be updated to highlight in the same color all of the entities considered in the same coreference equivalence class. As noted above (see Section 2.2), we currently have two simple coreference annotators: one that uses the head noun heuristic and one that uses the overlapping terms heuristic. In our example, the head noun annotator would assume, among other things, that any noun phrase with "inmates" as its head refers to the same entity: "27 inmates", "black and Latino inmates", "the inmates", etc. (Note that many of these proposed coreferences are incorrect — the heuristics are only meant to be used as baselines with which to compare other, better, coreference algorithms.) A quick scan of the text with all of these occurrences highlighted lets the user quickly determine how well the annotator is working for the current example. After limited pronoun resolution is added to the coreference annotator, "their" in "their cells" (paragraph 2) would also be highlighted as part of the same equivalence class.

4 HIGH-PRECISION INFORMATION RETRIEVAL

In order to maintain general-purpose retrieval capabilities, for example, current IR systems attempt to balance their systems with respect to precision and recall measures. A number of information retrieval tasks, however, require retrieval mechanisms that emphasize precision: users want to see a small number of documents, most of which are deemed useful, rather than being given as many useful documents as possible where the useful documents are mixed in with numerous non-useful documents. As a result, our research in high-precision IR concentrates on improving user time efficiency by showing the user only documents that there is very good reason to believe are useful.

Precision is increased by restricting an already retrieved set of documents to those that meet some additional criteria for relevance. An initial set of documents is retrieved (a global search), and each individual document is reparsed and matched against the query again to see if it satisfies the particular restriction criteria being investigated (local matching). If it does, the document is put into the final retrieved set with a score of some combination of the global and local score. We have investigated a number of re-ranking algorithms. Three are briefly described below: Boolean filters, clusters, and phrases.

4.1 Automatic Boolean Filters

Smart expands user queries by adding terms occurring in the top documents. Maintaining the focus of the query is difficult while expanding; the query tends to drift away towards some one aspect of the query while ignoring other aspects. Therefore, it is useful to have a re-ranking algorithm that emphasizes those top documents which cover all aspects of the query.

In recent work [14], we construct (soft) Boolean filters containing all query aspects and use these for re-ranking. A manually prepared filter can improve average precision by up to 22%. In practice, a user is not going to go to the difficulty of preparing such a filter, however, so an automatic approximation is needed. Aspects are automatically identified by looking at the term-term correlations among the query terms. Highly correlated terms are assumed to belong to the same aspect, and less correlated terms are assumed to be independent aspects. The automatic filter includes all of the independent aspects, and improves average precision by 6 to 13%.

4.2 Clusters

Clustering the top documents can yield improvements from two sources, as we examine in [12]. First, outlier documents (those documents not strongly related to other documents) can be removed. This works reasonably for many queries. Unfortunately, it fails catastrophically for some hard queries where the outlier may be the only top relevant document! Absolute failures need to be avoided, so this approach is not currently recommended. The second improvement source is to ensure that query expansion terms come from all clusters. This is another method to maintain query focus and balance. A very modest improvement of 2 to 3% is obtained; it appears the Boolean filter approach above is to be preferred, unless clustering is being done for other purposes in any case.

4.3 Phrases

Traditionally, phrases have been viewed as a precision enhancing device. In [13] and [12], we examine the benefits of using high quality phrases from the Empire system. We discover that the linguistic phrases, when used by themselves without single terms, are better than traditional Smart statistical phrases. However, neither group of phrases substantially improves overall performance over just using single terms, especially at the high precision end. Indeed, phrases tend to help at lower precisions where there are few clues to whether a document is relevant. At the high precision end, query balance is more important.

There are generally several clues to relevance for the highest ranked documents, and maintaining balance between them is essential. A good phrase match often hurts this balance by over-emphasizing the aspect covered by the phrase.

4.4 TREC 7 High Precision

Cornell/SabIR recently participated in the TREC 7 High Precision (HP) track. In this track, the goal of the user is to find 15 relevant documents to a query within 5 minutes. This is obviously a nice evaluation testbed for user efficient retrieval. We used the TRUESmart GUI and incorporated the automatic Boolean filters described above into some of our Smart retrievals.

Only preliminary results are available now and once again Cornell/SabIR did very well. All 3 of our users did substantially better than the median. One interesting point is that all 3 users are within 1% of each other: The same 3 users participated in the TREC 6 HP track last year with much more varied results. Last year, the hardware speed and choice of query length were different between the users. We attempted to equalize these factors this year. The basically identical results suggest (but the sample is much too small to prove) that our general approach is reasonably user-training independent. The major activity of the user is judging documents, a task for which all users are presumably qualified. The results are bounded by user agreement with the official relevance judgements, and the closeness of the results may indicate we are approaching that upper-bound.

5 CONTEXT-DEPENDENT SUMMARIZATION

Another application area considered to improve end-user efficiency is reduction of the text of the documents themselves. Longer documents contain a lot of text that may not be of interest to the end-user; techniques that reduce the amount of this text will improve the speed at which the end-user can find the useful material. This type of summarization differs from our previous work in that the document summaries are produced within the context of a query. This is done by

1. expanding the vocabulary of the query by related words using both a standard Smart cooccurrence based expansion process, and the output of the standard Smart adhoc relevance feedback expansion process;
2. weighting the expanded vocabulary by importance to the query; and
3. performing the Smart summarization using only the weighted expanded vocabulary.

We participated in both the TIPSTER dry run and the SUMMAC evaluations of summarization. Once again we did very well, finishing within the top 2 groups for the SUMMAC adhoc, categorization, and QandA tasks. Interestingly, the top 3 groups for the QandA task all used Smart for their extraction-based summaries.

Using the summ_eval evaluation tool on the SUMMAC QandA task, we are continuing our investigations into length versus effectiveness, particularly when comparing summaries based on extracting sentences as opposed to paragraphs. As expected, the longer the summary in comparison with the original document, the more effective the summary. For most evaluation measures, the relationship appears to be linear except at the extremes.

For short summaries, sentences are more effective than paragraphs. This is expected; the granularity of paragraphs makes it tough to fit in entire good paragraphs. However, the reverse seems to be true for longer summaries, at least for us at our current level of summarization expertise. The paragraphs tend to include related sentences that individually do not seem to use the particular vocabulary our matching algorithms desire. This suggests that work on coreference becomes particularly crucial when working with sentence based summaries.

Multi-Document Summarization. Our current work includes extending context-dependent summarization techniques for use in multi-document, rather than single-document, summarization. Our work on duplicate information detection will also be critical for creating these more complicated summaries. We have no results to report for multi-document summarization at this time.

6 DUPLICATE INFORMATION DETECTION

Users easily become frustrated when information is duplicated among the set of retrieved documents. This is especially a problem when users search text collections that have been created from several distinct sources: a newswire source may have several reports of the same incident, each of which may vary insignificantly. If we can ensure that a user does not see large quantities of duplicate information then the user time efficiency will be improved.

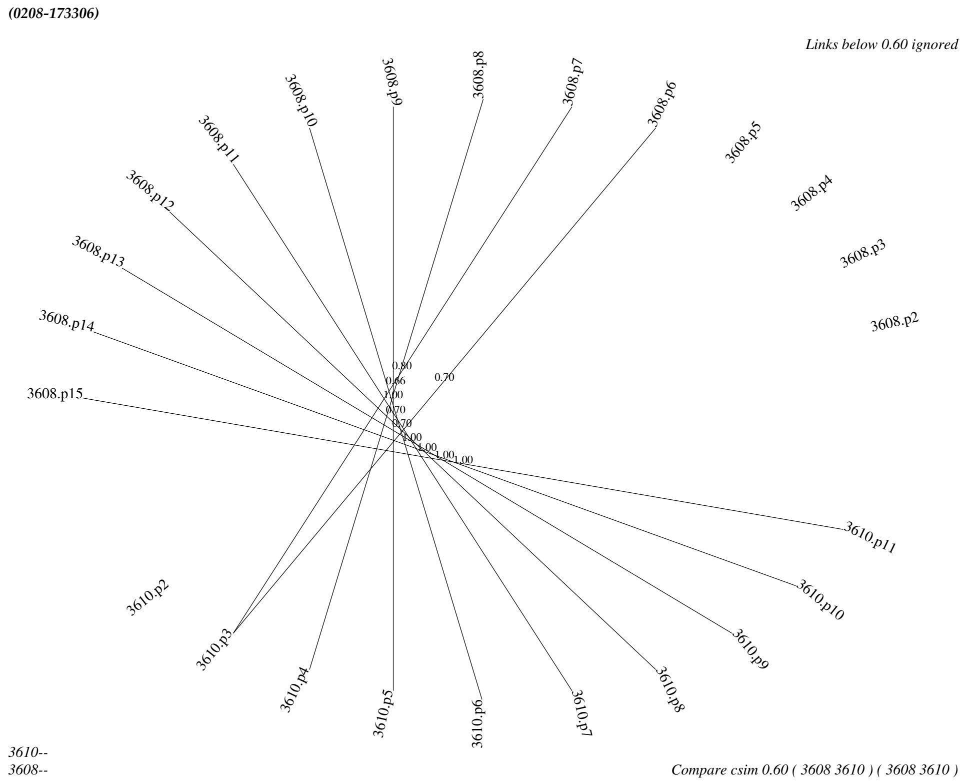

Figure 5: Document-Document Text Relationship Map for Articles 3608 and 3610. A line connects two paragraphs if their similarity is above a predefined threshold.

Exact duplicate documents are very easy to detect by any number of techniques. Documents for which the basic content is exactly the same, but differ in document metadata like Message ID or Time of Message, are also easy to detect by several techniques. We propose to compute a cosine similarity function between all retrieved documents. Pairs of documents with a similarity of 1.0 will be identical as far as indexable content terms.

The interesting research question is how to examine document pairs that are obviously highly related, but do not contain exactly the same terms or vocabulary as each other. For this, *document-document maps* are constructed between all retrieved documents which are of sufficient similarity to each other. These maps (see Figure 5) show a link between paragraphs of one document and paragraphs of the other if the similarity between the paragraphs is sufficiently strong. If all of the paragraphs of a document are strongly linked to paragraphs of a second document, then the content of the first document may be subsumed by the content of the second document. If there are unlinked paragraphs of a document, then those paragraphs contain new material that should be emphasized when the document is shown to the user.

The structure of the document maps is an additional important feature to be used to indicate the type of relationship between the documents: is one document an expansion of another, or are they equivalent paraphrases of each other, or is one a summary document that includes the common topic as well as other topics. All of this information can be used to decide which document to initially show the user.

Document-document maps can be created presently within the Smart system, though they have not been used in the past for detection of duplicate content [2, 17, 18]. Figure 5 gives such a document-document map between two newswire reports, one a fuller version of the other.

7 SUMMARY

In summary, we have developed supporting technology for improving end-user efficiency of information retrieval (IR) systems. We have made progress in three related application areas: high precision information retrieval, near-duplicate document detection, and context-dependent document summarization. Our research aims to increase end-user efficiency in each of the above tasks by reducing the amount of text that the user must peruse in order to get the desired useful information.

As the underlying technology for the above applications, we use a novel combination of statistical and linguistic techniques. The proposed statistical approaches extend existing methods in IR by performing statistical computations within the context of another query or document. The proposed linguistic approaches build on existing work in information extraction and rely on a new technique for trainable partial parsing. The goal of the integrated approach is to identify selected relationships among important terms in a query or text and use the extracted relationships: (1) to discard or reorder retrieved texts, (2) to locate redundant information, and (3) to generate coherent query-dependent summaries. We believe that the integrated approach offers an innovative and promising solution to problems in end-user efficiency for a number of reasons:

- Unlike previous attempts to combine natural language understanding and information retrieval, our approach always performs linguistic analysis relative to another document or query.
- End-user effectiveness will not be significantly compromised in the face of errors by the Smart/Empire system.
- The partial parser is a trainable system that can be tuned to recognize those linguistic relationships that are most important for the larger IR task.

In addition, we have developed TRUESmart, a Toolbox for Research in User Efficiency. TRUESmart is a set of tools and data supporting researchers in the development of methods for improving user efficiency for state-of-the-art information retrieval systems. In addition, TRUESmart includes a simple graphical user interface that aids system evaluation and analysis by highlighting important term relationships identified by the underlying statistical and linguistic language processing algorithms. To date, we have used TRUESmart to integrate and evaluate system components in high-precision retrieval and context-dependent summarization.

In conclusion, we believe that our statistical-linguistic approach to automated text retrieval has shown promising results and has simultaneously addressed four important goals for the TIPSTER program — the need for increased accuracy in detection systems, increased portability and applicability of extraction systems, better summarization of free text, and increased communication across detection and extraction systems.

References

[1] Steven. Abney. Partial Parsing via Finite-State Cascades. In *Workshop on Robust Parsing*, pages 8–15, 1996.

[2] James Allan. *Automatic Hypertext Construction*. Cornell University, Ph.D. Thesis, Ithaca, New York, 1995.

[3] Eric Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21(4):543–565, 1995.

[4] C. Cardie. *Domain-Specific Knowledge Acquisition for Conceptual Sentence Analysis*. PhD thesis, University of Massachusetts, Amherst, MA, 1994. Available as University of Massachusetts, CMPSCI Technical Report 94–74.

[5] C. Cardie. Embedded machine learning systems for natural language processing: A general framework. In Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors, *Symbolic, connectionist, and statistical approaches to learning for natural language processing*, Lecture Notes in Artificial Intelligence Series, pages 315–328. Springer, 1996.

[6] C. Cardie and D. Pierce. Error-Driven Pruning of Treebank Grammars for Base Noun Phrase Identification. In *Proceedings of the 36th Annual Meeting of the ACL and COLING-98*, pages 218–224. Association for Computational Linguistics, 1998.

[7] R. Grishman. TIPSTER Architecture Design Document Version 2.2. Technical report, DARPA, 1996. Available at `http://www.tipster.org/`.

[8] D. K. Harman. Appendix a, evaluation techniques and measures. In D. K. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages A6–A14. NIST Special Publication 500-236, 1996.

[9] J. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. In E. Roche and Y. Schabes, editor, *Finite-State Language Processing*, pages 383–406. MIT Press, Cambridge, MA, 1997.

[10] I. Mani, D. House, G. Klein, L. Hirschman, L. Obrst, T. Firmin, M. Chrzanowski, and B. Sundheim. The tipster summac text summarization evaluation: Final report. Technical report, DARPA, 1998.

[11] M. Marcus, M. Marcinkiewicz, and B. Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[12] Mandar Mitra. *High-Precision Information Retrieval*. PhD thesis, Department of Computer Science, Cornell University, 1998.

[13] Mandar Mitra, Chris Buckley, Amit Singhal, and Claire Cardie. An analysis of statistical and syntactic phrases. In L. Devroye and C. Chrisment, editors, *Conference Proceedings of RIAO-97*, pages 200–214, June 1997.

[14] Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In W. Bruce Croft, Alistair Moffat, C.J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–214. Association for Computing Machinery, 1998.

[15] J. Reynar and A. Ratnaparkhi. A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, San Francisco, CA, 1997. Morgan Kaufmann.

[16] Roche, E. and Schabes, Y., editor. *Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, MA, 1997.

[17] Gerard Salton, James Allan, Chris Buckley, and Mandar Mitra. Automatic analysis, theme generation and summarization of machine-readable texts. *Science*, 264:1421–1426, June 1994.

[18] Gerard Salton, Amit Singhal, Chris Buckley, and Mandar Mitra. Automatic text decomposition using text segments and text themes. Technical Report TR95-1555, Cornell University, 1995.

[19] E. M. Voorhees and D. K. Harman. Overview of the sixth Text REtrieval Conference (TREC-6). In E. M. Voorhees and D. K. Harman, editors, *The Sixth Text REtrieval Conference (TREC-6)*. NIST Special Publication 500-240, 1998.