A Literature Survey on Information Extraction and Text Summarization

Paper for Directed Reading (Fall 1996)

Klaus Zechner
Computational Linguistics Program
Advisor: Alex Waibel

April 14, 1997

Contents

1 Introduction

While the science of automatic information retrieval (IR) has been around for quite a long time now — basically, as long as databases of documents have been existing —, the scientific community only quite recently has been moving to the field of automatic information extraction (IE), maybe for as long as a decade now. There are at least two reasons for that: first, the amount of on-line (and off-line) textual data has been increasing exponentially, without a limit in sight (keywords are electronic newswires, World Wide Web (WWW)); secondly, the field of IE has been pushed forward by the ARPA funded evaluation series of the MUC conferences in the past ten years (see section 5.3). It has been demonstrated that fully automatic IE systems can be built with state-of-the-art technology, and that, for some selected tasks, their performance is as good as the performance of a human expert. Taking into account the enormous difference in throughput, machines can particularly serve as "rough preprocessors" of the Terabytes of information out there, which then can be further analyzed by human experts.

For text summarization, the picture is a bit ambivalent: Early ideas and systems of automatically condensing and/or summarizing documents date back into the fifties (Luhn, 1958); there have been many different attempts to that goal, combining low level (e.g., statistical) and high level knowledge (e.g. domain models), with the rough conclusion that while it is quite feasible to produce a short "indicative" abstract (or: extract) from a text, it is still very hard to produce also a sufficiently coherent and "readable" one — if one aims at applying it to general domain real data.

The goal of this paper is to give a survey mainly on these two latter fields of research, i.e. information extraction and text summarization. I will focus mainly on approaches for domain-independent purposes which require little effort of knowledge engineering and domain modeling since I believe that first, the latter is very labor intensive and therefore expensive, and secondly, that successful systems should be (and are) built in a way which *can* be customized for narrow-domain applications if needed. Other approaches will also be mentioned and discussed here, but to a lesser extent.

2 Shifting Paradigms in NLP: Chomsky-style vs. MUC-style

It is quite clear from within and from without the field of Computational Linguistics that the field has been undergone a significant change in the past decade. While some might call it a mere "extension" to areas of more applied research (let us call it "language technology"), others feel about that more in terms of a shifting *paradigm* in a Kuhnian sense (Kuhn, 1970). This can be seen on indicators such as: contributions to journals and conferences, focus of public and private funding, development of academic curricula, and changes in the "job market" for computational linguists. The trend is, I claim, ubiquitous: Moving more away from research that is motivated by linguistic theory and moving towards applied technology. E.g., people are less concerned about whether *green ideas sleep furiously* or not, but, e.g., more how to handle repetitions or false starts in processing of spontaneous speech, or how to parse "Mr. Smith, the CEO of NNC Corp." in a newswire article.

It may be the case that what happens now in NLP and CL is similar to what happened in other sciences as well[1], in their developmental history: from being an "descriptive" and "classificatory" enterprise, to becoming a science which readily interacts with society in that it provides the basis for the creation of "useful products". It is probably to early to tell at this point whether our conjecture holds or not, but at least the changing trends can be observed quite clearly, and nobody within the community would dispute it. The dispute rather arises about the subject matter of the "old" vs. the "new" paradigm — or, to put it bluntly: how much funding and support is needed for "purely" theoretical research within NLP, compared to application oriented research and development of systems with a potential economic benefit in the end.

But these (sometimes heavy) disputes are even more of an indicator for the possibility that we are in the midst of a Kuhnian scientific paradigm shift, comparable to that in the intrinsically related field of linguistics in the sixties, triggered by Chomsky's writings and arguments.

What were (and are) some of the contributing "external" factors for this change?

- **Machine Power**: Computers in the fifties or sixties were simply to weak to handle reasonable amounts of data in a reasonable amount of time, even if the algorithms were simple; memory was measured in kilobytes (or: in kilo-words) at that time.

- **Amount of Data**: As already mentioned in the introduction, it was not until the late eighties, that corpora of a significant size were made publicly available and accessible. (Certainly, this is also linked to the first point about machine power, of RAM and disk storage sizes.) The hugest

[1]E.g., in Biology or Chemistry, maybe one hundred years ago.

corpus has been building up for about three years now and will quite likely continue to grow in an exponential way: the World Wide Web (and, to be more comprehensive: all data accessible via the Internet).

- **Rebirth of the Belief in Statistics**: After the shift from (behavioristic) empiricism to (introspective) mentalism in linguistics in the sixties, people in that field have practically abandoned statistics about language as a field of potential interest or insights. This changed not insignificantly after the publication of IBM's seminal work on statistical machine translation (MT) (Brown et al., 1990). Contrary to predictions, their system which employed essentially no linguistic knowledge (at that time) was not only working, but it did so surprisingly well, given the difficulty of the MT task. At the same time, a lot of corpus based methods and approaches were developed and implemented, such as automatic part-of-speech (POS) tagging, formerly also believed to be "doomed" without sophisticated handcrafted linguistic knowledge (Kupiec, 1992; Brill, 1994).

- **Theory-Driven Systems Remain Lab-Toys**: People in NLP frequently had to realize that although they had put lots of efforts and man-years into their sophisticated systems, these were in the end never useful other than as being research prototypes. As long as the funding agencies were willing to appreciate that, this was fine, but then came a change:

- **Competitive Evaluations**: It was some time in summer of 1989, when people in the ARPA language group were moving from thinking aloud to taking action in what turned out to be one if not *the* most influential initiative in the field of NLP: they founded TIPSTER, a huge project centered around building textual corpora and hosting various competitive conferences where participating sites had to develop systems for specifically designed tasks. The two main stands of conferences are TREC (Text Retrieval Evaluation Conference) and MUC (Message Understanding Conference). Actually, the first two MUC conferences (named MUCK-1 and MUCK2) and their success were the "triggers" for the TIPSTER enterprise, according to (Prange, 1996).

As it stands, ARPA has been enjoying the success in its traditional speech evaluation programs and so the NLP people were driven (both by themselves and by others) to adopt the notions of evaluation and competition now also to the fields of IR and IE.

This was and is by no means self-evident, as it has become clear by what I said earlier. Although one cannot really claim that evaluation has not been an issue in NLP before ARPA came along, it was certainly not a major concern for most researchers. Competition before that meant more the competition of ideas rather than of numbers.[2] To illustrate the way people were thinking, I quote

[2]This is of course also due to the fact that numbers are hard to obtain from an NLP task:

the following passage from (Herzog and Rollinger, 1991, 13)'s report about the huge LILOG NLP project in Germany (my emphasis):

> LILOG was a project that produced more than could have been expected at the outset. Not only were a *large number of scientific papers, implemented systems, diplomas, dissertations and professors produced* but also a large part of a whole generation of scientists working on language understanding was brought together.

How much the concentration on evaluation, on numbers and percentages etc. makes sense is first of all not really the issue here and secondly probably hard to tell anyway. There is the argument by some people that while engaging in these evaluations, valuable research potential is lost because of spending much time in scientifically uninteresting issues such as fine-tuning, data formats, hardware and software configurations etc. On the other hand, it is not clear a priori where to draw the line between an intrinsically scientific question and a (allegedly?) trivial one.

Whatever is the stand in the debate, the *progress* and *impact* that the TIPSTER, TREC, and MUC programs are making, is undoubtedly there, and so IR and maybe even more so IE systems are on the verge of becoming one of the first applied, engineered, and economically viable “NLP products” in the history of this science.

how does one precisely measure the accuracy of a parser, a semantic interpreter, or a MT system?

3 Some Key Concepts of Information Retrieval and Its Relatives

3.1 Main Tasks of an IR System

Textual information is stored in computer systems in at least two distinguishable forms:

1. in database management systems (DBMS): as data elements in tabular form, or

2. in information retrieval (IR) systems: as natural language texts in large collections or corpora.

Whereas for the former type, the user typically has to query the system in such a way that it can provide him with the exact information he needs (*fixed ways of access*), things are different for the latter: With IR systems, what the user is looking for, is typically not a *specific* entry or item (or an array/table of these) but rather a list of items (in particular: documents, or parts of them, in certain circumstances) which "deal with" a subject matter he is interested in. Typical queries in IR systems consist of boolean AND–OR–NOT–combinations of keywords which the system then has to match to its database to retrieve a set of items as the "best approximation" to this query.

Hence, from a functional point of view (see (Salton and McGill, 1983, 10ff.)), an IR system consists of

- a set of information items (the data itself),
- a set of user queries or requests, and
- some mechanism which has to determine which of the information items match the user's request.

So there are — at least — three main tasks which an IR system has to accomplish:

1. indexing the information items, i.e. assigning a number of discriminative keywords (index words) to all items and (possibly) also weighting them[3]

2. mapping the user's request onto an internal representation language[4]

[3]This can be done and has often been done by humans but is increasingly done automatically, using statistical methods; in fact, it has been shown that automatic indexing is not only as good as manual indexing but even superior for IR purposes (Lewis and Sparck-Jones, 1996).

[4]In general, this will have some form of the above mentioned boolean keyword-combination. However, the user does not always have to put his queries in this format directly; some IR systems allow for "higher level" input, in a format which is more like natural language.

Items Retrieved	Items Being Relevant	Recall	Precision
6	4	0.27 (4/15)	0.67 (4/6)
10	6	0.40 (6/15)	0.60 (6/10)
20	9	0.60 (9/15)	0.45 (9/20)

Table 1: Recall and precision of an example IR system, assuming 15 relevant items for a given user query.

3. determining the proper subset of those information items the indices of which have the highest similarity with the (converted) user query and return these items to the user

Usually, the user's initial query will be either too general or too specific or not "close to the point" (where the user wants to get but does not know how because he does not have the system's "map" of keywords). Therefore, the system's response will not satisfy him at first. Typically, in an interactive online system, an iterative process of successive refinement develops which will get shorter as the user gets accommodated to the system. Some more sophisticated IR systems are also able to learn from these adaptive processes and also might be able to help the user speeding up his search by giving him hints and suggestions about possible (related) keywords, combinations or request strategies in general.

3.2 Evaluation Metrics: Precision, Recall, and F-score

In terms of assessment of the quality of IR systems, the following points are generally considered to be important (see e.g. (van Rijsbergen, 1979, 145)):

1. the *user–interface*: how easy are access to and usage of the system, how fast is the learning curve?

2. the system's *speed* (retrieval response times of more than a few seconds tend to frustrate the users soon)

3. the system's *recall*: the proportion of relevant items that actually were retrieved

4. the system's *precision*: the proportion of retrieved items which are actually relevant

We shall explain the two terms *recall* and *precision* by giving a small example (see Table 1). Assume, a collection holds n documents, $d_1...d_n$, which are indexed for a number of keywords contained therein. A user, being interested in information about "holidays in Spain", queries the system by using a boolean combination of index keywords, e.g., by formulating the query ("holiday" AND "Spain"). We now assume that 15 documents actually deal with “holidays in Spain”, so these would be exactly the items the user is interested in. The system now performs a similarity computation and presents the 10 top ranked documents to the user. It turns out that 6 of them belong to the 15 “relevant” documents about “holidays in Spain” but 4 do not (e.g., they may contain sections about holidays and about Spain but none about holidays IN Spain, therefore getting a high similarity measure while not being relevant to the user.) The *recall* value is 0.4 (there were 6 out of 15 relevant items in the retrieved items), whereas the *precision* value is 0.6 (6 out of 10 retrieved items were relevant).

As the number of items grows which the IR system retrieves, recall improves but precision decreases: assume, we ask the system to present 20 items instead of 10. Say, there are now 9 relevant items in total (i.e., just 3 of the second 10 items are amongst the relevant ones), which yields 0.6 for recall and 0.45 for precision. The converse also holds, so if the system just presented 6 items, 4 of them being relevant, we get 0.27 for recall and 0.67 for precision. Both recall and precision are always in the interval [0.0,1.0], their optimum being at 1.0. But they are, as our example shows, inversely related to each other.[5] So, in a realistic IR system one has to find a compromise between these two parameters which can also be dependent on the user's interest: sometimes, when exhaustiveness is important, he will want to maximize recall, whereas if he exclusively wants to deal with *relevant* information, a high precision will be required.

A problem with the precision/recall-measure is that, in general, it is not a priori clear how actually to determine the subset of *relevant items* which is necessary for computing both of these indicators. One would have to ask the user to look over the entire document collection and to check each article in order to determine whether it would be relevant for his respective query or not. To avoid this laborious and realistically unmanageable task, there exist test–collections where the “optimal matching documents” for a given set of queries are known and the precision/recall values can therefore be computed. An alternative to that, the so-called *pooling method* is used, e.g., for the TREC evaluations (Harmann, 1993): here, *all* the documents retrieved by the participating systems are being put into a *pool* and judged by human assessors for their relevance. Against this set, precision and recall of each individual system can now be measured.[6] Obviously, this method is suboptimal but it avoids the time-consuming and costly effort of human labeling which, in case of a collection of several Gigabyte, is unrealistic to perform exhaustively in the first place.

Since it is not so straightforward to compare two parameters (precision and recall) at the same time, various combination methods have been proposed, the

[5]Though this is an empirical observation and not a strict mathematical relation.

[6]For a fair comparison, typically all systems are requested to submit only the top N retrieved documents.

most prominent of which is the F-score which is defined as follows:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{1}$$

where β is a weighting parameter for weighting either precision (P) more than recall (R) or the other way round.[7]

3.3 Automatic Indexing: Determining Term Weights

Virtually all IR systems rely on some form of indices or keywords which are associated to all documents of a collection. A standard way of generating text indices automatically would be as follows (compare, e.g., (Salton and Buckley, 1990, 8)):

1. Identify the individual text words
2. Use a dictionary of common high-frequency function words (i.e., a stop list) to ignore these words from the text
3. Use a suffix stripping routine (a morphological analyzer) to reduce the words to their stems
4. Remove all terms whose count is below a pre-specified threshold
5. Compute a term weight for each word stem
6. Represent a document as a vector of the weights of all index terms in a collection (i.e., all word stems found by this procedure)

A standard method for term weight computation in IR is the "term-frequency * inverse-document-frequency" method (*tf*idf*, see e.g. (Salton and Buckley, 1990, 8)): The weight for a term k (w_k) is a product of the number of occurrences of a term in a single document (tf_k) times the logarithm of the number of documents in a collection (N), divided by the number of documents where this term occurs (n_k):

$$w_k = tf_k * \log \frac{N}{n_k} \tag{2}$$

This formula yields a high number for words which are frequent in one document but appear in very few documents only; hence, they can be considered as "indicative" for the respective document.

When a user's request is given to the system, it computes the similarity of the query-vector (Q) with all the document vectors (D) and retrieves those

[7]Most of the times, $\beta = 1$ — s.t. recall and precision are weighted equally —, so $F_1 = \frac{2PR}{P+R}$.

documents with the highest similarity measure (the w weights are the *tf*idf*-weights of the document):

$$sim(Q_i, D_j) = \sum_{k=1}^{n} tf_{ik} w_{jk} \tag{3}$$

Since long documents contain many distinct words, many of them with higher frequencies, a similarity metrics just based on these *tf*idf*-weights will exhibit a bias towards longer documents. Therefore, it is often useful, to normalize these weights for document length, e.g., for term T_k in document D_i (Salton and Buckley, 1990, 9):

$$w_{ik} = \frac{tf_{ik} idf_k}{\sqrt{\sum_k (tf_{ik} idf_k)^2}} \tag{4}$$

4 Information Extraction

Information Extraction (IE) is a comparatively new field in NLP. Its prime objectives are twofold: (i) to divide input text in “relevant” and “irrelevant” sections (“filtering”), and (ii) to fill pre-specified *templates* with information extracted from the text; these filled templates can be used, e.g., to be entered in standard relational databases, to serve as automatic triggers for informing human data analysts of events of interest, or to provide a basis for generating text abstracts or condensed versions of source texts.

The major push in this field came from the MUC evaluations, now being grandfathered from the TIPSTER program (see section 5). Several of the example systems described here, were successful participants of the recent MUC evaluation.

While simple tasks can be done to a high degree of accuracy already (e.g., extracting all proper names and companies mentioned in texts), the performance with more complex tasks (e.g., getting co-reference right, determining a sequence of events) is still significantly below that of a human expert; on the other hand, it has to be taken into account that these systems are orders of magnitudes faster and they can be useful in selecting the *right* data which are then further dealt with by human analysts, if necessary.

Typically, IE systems are constructed as modular pipelines from components that are either fully or at least semi-automatically trainable from corpora and annotated texts; unlike their predecessors that focused on “understanding” (see the next subsection), they use *shallow* knowledge instead of *deep* knowledge, the advantage being the speed of implementation, adaptation to new domains, and increased robustness, the disadvantage being a loss in accuracy.

4.1 Information Extraction from Written Text

4.1.1 Text Understanding Systems

A precursor of IE was and is the field of text understanding. AI researchers have been building various systems where the goal is to get an accurate representation of the contents of an entire text. Unlike modern IE systems which have a relevance filter, they usually deal with every sentence in the text, and try to come up with a full scale syntactic, semantic, and pragmatic representation. Obviously, a large amount of hand-crafted engineering is needed here, particularly for building the knowledge bases. For this reason, these systems typically operate in very small domains only, are usually not very flexible in the sense of being easily portable to a new domain, and not very robust when faced with “real world” data.

Two examples for frequently discussed systems in this realm are the PUNDIT system (Hirschman et al., 1989) and Hahn’s TOPIC system (Hahn, 1989).

A system which claims to use “full natural language understanding” for information extraction, is PAKTUS (Thompson, 1994). It performs full syntactic and semantic processing, enhanced with pattern matching and filtering components. PAKTUS participated in the earlier MUCs (up to MUC-4); for the recent MUC-6, no results could be found.

4.1.2 "True" IE Systems

4.1.2.1 SCISOR SCISOR ("System for Conceptual Information Summarization, Organization, and Retrieval") is a system which performs text analysis and question answering in the domain of financial news, selecting and analyzing stories about corporate mergers and acquisitions (Jacobs and Rau, 1990). It performs the following tasks:

1. Lexical analysis of the input stream (names, dates, numbers, etc.)
2. Separation of the raw news into story structures
3. Topic classification ("acquisition/merger/other")
4. Natural language analysis
5. Interaction with a knowledge base (storing/retrieving information)

SCISOR combines AI methods with more shallow methods, the goal being to get the best benefits from both of these divergent approaches; this amounts to a combination of "bottom-up" and "top-down" technologies.

SCISOR participated in the MUCK-2 conference, but no performance data are available for that. (Jacobs and Rau, 1990) report accuracies of over 90 percent in topic classification and F-scores in the 0.8-0.9 range for filling the domain-specific templates.

4.1.2.2 FASTUS At SRI, the NL research group around Jerry Hobbs has been developing the FASTUS system for about five years, which is also aimed at extracting information from natural-language text (Hobbs et al., 1992; Hobbs et al., 1996). FASTUS (in its current form (Appelt et al., 1995)) consists of a multiple stage cascaded non-deterministic finite-state transducer. The stages are organized as a pipeline as follows:

1. tokenization
2. multi-word analyzation (e.g. "because_of")
3. string-mapping preprocessor (e.g. "twenty two" $\rightarrow$ "NUM(22)")
4. recognition of fixed expressions (e.g., names)
5. parsing basic noun groups, verb groups

6. building complex noun groups and verb groups, coordination, appositions
7. building of domain specific event-structures
8. merging of event structures and database entry

FASTUS was probably the first large system built for a NL task which heavily relies on finite state technology. Its success and its inspiration for other approaches can hardly be overestimated. The main advantages of FASTUS are (i) its conceptual simplicity, (ii) its effectiveness, (iii) its fast run time behavior, and (iv) its fast development/adaption time when moving to a new domain.[8]

The MUC-6 results[9] of the FASTUS system were F=.94 on the Named Entity task, F=.75 on the Template Entity task, F=.65 on the Coreference task, and F=.51 on the Scenario Template task.

4.1.2.3 The University of Massachusetts System The UMass system (Fisher et al., 1996) used for the MUC-6 evaluation consists of the following main components, most of which use decision trees and possibly other machine learning techniques that allow for automatic training:

- String Specialists: pattern matching routines for the recognition of proper names, dates and similarly fixed noun phrase descriptions
- BADGER: a part-of-speech (POS) based sentence analyzer which also does apposition-attachment and instantiates semantic case frames. Both POS dictionary and the semantic feature hierarchy are manually customized for a specific domain.
- CRYSTAL: a fully automatic inductive dictionary construction system
- WRAP-UP: decides about the proper role of a noun-phrase, using various sources of evidence
- RESOLVE: handles the merging decisions and decides, when two noun phrases refer to the same entity

The results in MUC-6 were as follows: F=.85 for Named Entity, F=.61 for Template Entity, F=.46 for Coreference, and F=.40 for Scenario Template. Although this system was partly based on previous approaches, it should be mentioned (to be fair wrt. to the longer evolved FASTUS) that most of the components were developed only in a few months' time before the evaluation.

[8]NYU has recently also moved from a traditional approach that used global coverage parsing to an approach which is very similar to the FASTUS system (Grishman, 1995).

[9]More information about MUC and its tasks is provided in section 5.3.

4.2 Information Extraction from Spoken Language

Whereas for written language, a variety of systems have been built over the past decade (primarily driven by the MUC evaluation series), the situation wrt. spoken language is really different: so far, most evaluations in the spoken language field have been concerned with comparing word error rates (WER), but have not yet gone beyond that, maybe except for the German VERBMOBIL project (Wahlster, 1993); but here, the focus has not been so much on IE, but rather on MT and related issues. On the other hand, in any limited domain setting of MT, where some sorts of intermediate representations are available, these can be interpreted as IE templates, at least to a certain extent. Seen this way, the two parsers employed by the JANUS speech-to-speech translation system (Waibel et al., 1996), Phoenix (Ward, 1994) and GLR* (Lavie, 1996) — in combination with a discourse processor (Qu et al., 1996) —, perform (also) an IE task in the scheduling domain.

The only *explicit* IE system for speech is the MIMI system from SRI, in a way an offspring from the successful FASTUS system.

4.2.0.4 MIMI MIMI operates in the Conference Scheduling domain and tries to collect as much information as possible from the utterances of the two involved dialogue partners ((Kameyama and Arima, 1994; Kameyama et al., 1996) report results using 150 transcribed Japanese dialogues). While the dialogue proceeds, a "summary" is built and updated incrementally. The processing steps are basically the same as for FASTUS, but three major extensions are made to accommodate to the spontaneous speech dialogue situation (Kameyama et al., 1996):

1. flexible recognition of sentence/utterance units (to handle the lack of clear sentence boundaries in spontaneous speech)

2. utterance lookahead (to handle disfluencies, discontinuities, and interruptions)

3. information override (to accommodate to the negotiation type dialogues)

Since the speech recognizer used in MIMI was trained on read speech, the word error rate (WER) was high (over 50%), even after two thirds of the test set was discarded due to the occurrence of out-of-vocabulary words. On transcripts, an F-score of 0.8 was achieved (MIMI-filled templates scored against human filled templates). Due to the high WER rate, the F-score on recognizer output dropped significantly to 0.53. However, an indication for the robustness of the system wrt. speech recognizer accuracy is its linear drop in performance with increasing word error rate. (Since (Kameyama et al., 1996) do not report how many slots and possible fillers there are for this task, these F-scores are hard to interpret properly.)

5 The Driving Forces: TIPSTER, MUC, and TREC

This section will give some general background about the history and layout of the DARPA sponsored TIPSTER program and its two major administered conferences, the MUCs and TRECs.

5.1 The TIPSTER Text Program

The TIPSTER program was founded in 1989 by DARPA's Speech and Technology Program. At that time, this technology program was heavily dominated by its speech component and NLP people within DARPA were wondering how they could possibly establish a similarly successful program in the domain of text and document databases, where the demand for high quality products had been rapidly increasing over the past decade, due to reasons mentioned earlier.

The major conclusion these people drew were that they would have to adopt the "Evaluation Driven Research" paradigm from the speech branch in order to achieve a continously evolving progress and to become as respected and as important as the speech program was at that time (Prange, 1996).

For that to be accomplishable, they would need

- a final objective and specific tasks which can lead to achieve that goal eventually
- a clear metric for evaluation
- participants from industry and universities, willing to participate in regular evaluations and to discuss their approaches and their progress
- sufficient data for training and testing
- sufficient funding sources

An important goal of the TIPSTER enterprise has also been to encourage the transfer of technology from research into industry, from the lab to the end-user. Obviously, this goal has strongly influenced the whole design of the program in that the assumptions made had to be as realistic and non-idealizing as possible to reflect "what is out there in the real world."

The state-of-the-art around 1989 in document detection (DD) and information extraction (IE) systems was investigated and the main conclusions were that

- most systems required experts as users
- their performance was pretty poor
- most work has been done only for English documents

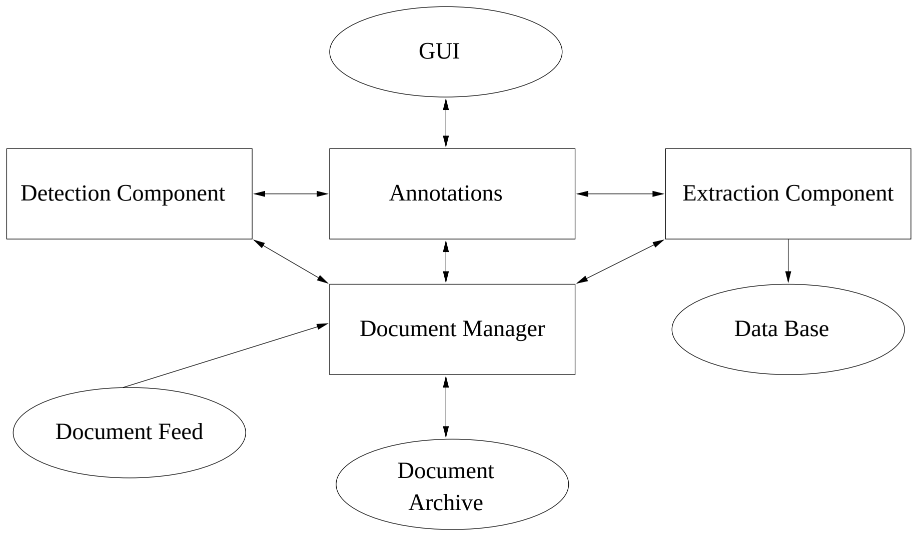

Figure 1: TIPSTER Architecture

- the systems were hardly portable and had a low error tolerance

As a consequence of that, a two stage process was designed: on Phase I (started in 1991), the focus was to push the state-of-the-art in DD and IE; Phase II (started in 1994) should concentrate on bringing these two technologies together synergeticly — the so-called "TIPSTER-Architecture" was developed which intends to "facilitate the deployment unto the workplace of advanced DD and IE software" (see Figure 1). The four major components are Detection (document retrieval and routing), Extraction (entity-relationship identification), Annotation (using information from the Extraction component), and the Document Manager who handles the documents' storaging and archiving.

Since Fall 1996, the Phase III of TIPSTER has been underway, where, additionally to the goals of the earlier phases, the challenge of text summarization is added on top.

TIPSTER has been hosting two main strands of conferences, the MUCs and the TRECs which will be described next.

5.2 The Text REtrieval Conferences (TRECs)

From 1992 to 1995, there have been four TRECs; participants were mostly academic and private research institutions in the U.S., but also some sites from

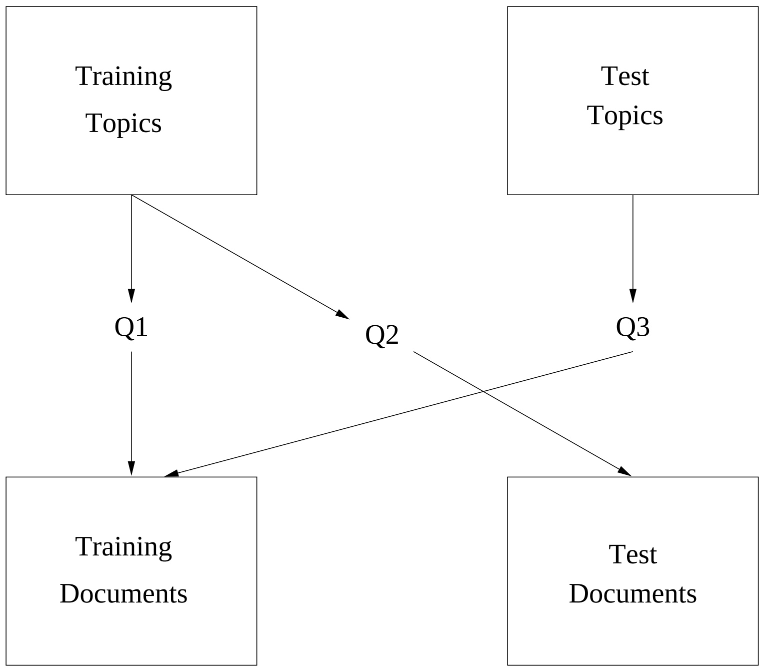

Figure 2: TREC tasks

other countries (e.g., UK, Australia, Switzerland); the total number was 36 participants for TREC-4.

The two major rationales of TREC are:

1. to provide uniform data and evaluation methods for all participants
2. to provide a realistically sized text collection

The TREC data comprises two CDROMs with data from Wall Street Journal, AP News Wire and some other minor sources, totalling about 2 Gigabyte. These are split in training and test corpus of equal size and also, there were 50 topics put in the training and 50 topics in the test set.

The two main TREC tasks are (see Figure 2):

- document retrieval using automatic and/or manual query construction (using topic descriptions for that purpose) ("adhoc task", Q3 in Figure 2)
- document routing ("routing task", Q2 in Figure 2)

The purpose of the routing tasks is to assign *known* topics to a set of *unknown* documents. In the adhoc-task, a sample of queries related to *unknown* topics is used for the *known* corpus of documents, to retrieve those that best match these topics.

Evaluation follows the traditional notions of precision and recall. Particularly, precision was measured for recall rates at every 0.1 interval, from 0.0 to 1.0. However, to limit the amount of data for these evaluations, an arbitrary cutoff of 200 returned documents per query was determined. Therefore, the precision values diverge from the "true" ones as soon as recall increases beyond a certain level.

In order to get the "keys" (i.e. the set of relevant documents), the so-called "pooling method" was used (see section 3.2).

At recent TRECs, systems typically performed at approx. 0.4 precision when recall is 0.5 for the adhoc task with automatic query formulation, slightly worse or about equal for manual query formulation, and about the same also for the routing task.

The interesting result from the evaluations was, that automatically constructing queries was as good if not better than doing it manually.

5.3 The Message Understanding Conferences (MUCs)

Actually, the first two MUCs[10] were *predecessors* of the TIPSTER program. In fact, the success of MUCK-2 (held in June 1989) was one of the major driving forces for the establishment of TIPSTER. Later, the MUCs were incorporated into the TIPSTER paradigm and have been serving as its "Information *Extraction*" part, as opposed to the TRECs which focus on Document *Retrieval.*

The focus has always been on research on the automatic analysis of texts. In the course of time, different tasks and subtasks emerged out of this rather general goal.

5.3.1 A brief history

The MUC-1 in 1987 was merely exploratory, there was no formal evaluation made. In MUC-2, held in 1989, the task was to fill templates; these templates were related to event-description of events dealt with by the respective texts (e.g. type, agent, place, time, etc.). Evaluation was done using recall and precision in terms of filled templates. (One template had 10 slots).

Whereas for the first two MUCs, the subject matter were military messages about naval sightings and engagements, in the next two MUCs (1991, 1992), the topic was "Latin American Terrorism". Now, the number of slots increased to 18 resp. 24 per template.

[10]At that time, their names were actually MUCK-1 and MUCK-2.

Task	Precision	Recall
Named Entity	.96	.97
Coreference	.72	.58
Template Element	.86	.75
Scenario Template	.70	.47

Table 2: Best Results for the 4 Tasks of the MUC-6 conference.

Then, from MUC-5 (1993) on, the conference was conducted as part of the TIPSTER program (see above). Topics shifted again, the task complexity further increased, and a second language (Japanese) was added. Also, a nested object hierarchy (as opposed to a flat one) was used for MUC-6 for the first time.

5.3.2 The tasks of MUC-6

MUC-6 (1995) grew to a total of four distinct tasks (Sundheim, 1995):

- Named Entity (NE): SGML tagging of all persons, organizations, places, dates, or times mentioned in the text
- Coreference (CO): SGML tag-links to all coreferring noun phrases in the text
- Template Element (TE): filling templates about persons and organizations
- Scenario Template (ST): filling templates about events, relating these to any participants (persons, organizations, etc.)

5.3.3 Performance of MUC-6 systems

The test set for the two latter tasks consisted of 100 WSJ articles; 30 of them were used for the two annotation tasks. The best results for these four tasks are given in Table 2.

The surprise for the organizers was the high performance of many systems for most of the tasks. Specifically, the NE task was done very well by almost all systems, not (much) worse than humans would do that. Also, for the TE task, quite high scores were achieved. Coreference and ST proved to be harder (as expected), but still manageable at a good level.

6 Summarizing Texts

6.1 The Issue of Text Abstracting

In many situations, users would and do prefer to look at text *abstracts* rather than at the whole text, before they decide whether they are going to read through the entire text or not. The two main types of abstracts are

1. (man-made) summaries
2. (machine-made) extracts

Man-made summaries, generally produced by the author himself, are very common in scientific articles and their purpose is to give the reader a *short* and *coherent* impression of the main idea of this article. Although, in general, the authors do no just collect the important phrases from their articles, (Kupiec et al., 1995) showed that about 80% of the sentences in man-made abstracts were "close sentence matches", i.e., they were "either extracted verbatim from the original or with minor modifications." ((Kupiec et al., 1995, 70))[11]

In a machine system, the goal of coherence is doubtless difficult to achieve and would (in the optimal case) require not only the system's ability of text understanding but also of text generation. However, what a machine system can do quite readily, is to produce an *extract* of the text, i.e., to select a number of "most relevant sentences for this text" and present them to the user in text order. For many purposes, coherence is not a crucial point, but *relevance* certainly is. So: how does the system determine the relevance of passages in a given document? We will investigate this question in the following subsection, specifically by looking at a few existing systems in this domain, which range from "bottom-up" approaches, where a combination of statistical word distribution information and some either general or text specific heuristics are combined, to (limited domain) "top-down" approaches, where (at least *some*) understanding of the subject matter is crucial.

6.2 Written Text Summarization

6.2.1 Systems Using Term Statistics (and Heuristics)

Interest in producing simple indicative abstracts, i.e., "extracts as abstracts", arose as early as the fifties. An important paper of these days is the one by (Luhn, 1958) who suggested to weight the sentences of a document as a function of high frequency words, disregarding the very high frequency common words.

[11]These abstracts were *not* created by the authors themselves but by professional abstractors.

6.2.1.1 The Abstract Generation System by Edmundson (1969)

(Edmundson, 1969) implemented a system for automatically generating text abstracts which, additionally to the standard keyword method (i.e., frequency depending weights), also used the following three methods for determining the sentence weights:

1. Cue Method: This is based on the hypothesis that the relevance of a sentence is affected by the presence or absence of certain (pragmatic) cue words; there were both "bonus" and "stigma" words in the cue dictionary, i.e., words which would have a positive or a negative effect on the respective sentence weight. These cue word lists were computed statistically, using man-made abstracts as a reference.

2. Title Method: Here, the sentence weight is computed as a sum of all the content words appearing in the title and (sub-) headings of a text.

3. Location Method: This method is based on the assumption that sentences occurring in initial/final position of both text and individual paragraphs have a higher probability of being relevant.

The results showed, that the best correlation between the automatic and target (human-made) extracts was achieved using a combination of these three latter methods, i.e., omitting the key word method entirely.

6.2.1.2 The Trainable Document Summarizer by Kupiec *et al.* (1995)

This system also performs a sentence extracting task, based on a number of different weighting heuristics. Specifically, the following features for sentence scoring, some of them resembling those employed by (Edmundson, 1969), were used and evaluated:

1. Sentence Length Cut-Off Feature: sentences containing less than a prespecified number of words are not included in the abstract

2. Fixed-Phrase Feature: sentences containing certain cue words and phrases (or: following section headers containing these) are included

3. Paragraph Feature: this is basically equivalent to Edmundson's Location Method

4. Thematic Word Feature: the most frequent words (considering word length as parameter) are defined as thematic words; sentence scores are functions of the thematic words' frequencies

5. Uppercase Word Feature: upper-case words (with certain obvious exceptions) are treated as thematic words, as well

(Kupiec et al., 1995) use a corpus which contains 188 document/summary pairs from 21 publications in a scientific/technical domain. The summaries were produced by professional abstractors and the sentences occurring in the summaries were aligned to the original document texts, indicating also the degree of "similarity": as mentioned earlier, the vast majority (about 80%) of the summaries' sentences could be classified as "direct sentence matches", i.e., (almost) verbatim extractions or extractions with minor modifications only, having an equivalent content.

A statistical evaluation, which computes the performance for each of these features separately, shows that the summarizer produces a recall between 0.2 and 0.33[12], the latter achieved by the Paragraph Feature. The best feature combination was (Paragraph + Fixed-Phrase + Sentence-Length) and yielded a recall of 0.44. Since the number of sentences produced by the system always equals the number of sentences in the corresponding manual summary, precision and recall are identical.

When looking at these results, one has, however, as (Kupiec et al., 1995) mention, to take into account the inter- and intra-human variability, concerning the production of a document abstract. (Rath et al., 1961)[13] showed in a study that extracts selected by four different human judges had only 25% overlap, and for a given judge over time only 55%.

Interestingly, the frequency-keyword features (the thematic word feature and the uppercase feature) also gave the poorest performance, similarly to the results of (Edmundson, 1969).

6.2.1.3 The ANES text extraction system by Brandow *et al.* (1995)

ANES is a system that performs automatic, domain-independent condensation of news data. The process of summary generation has four major constituents:

1. corpus analysis: this is mainly a calculation of the *tf*idf*-weights for all terms
2. statistical selection of signature words: terms with a high tf*idf-weight plus headline-words
3. sentence weighting: summing over all signature word weights, modifying the weights by some other factors, such as relative location
4. sentence selection: high scoring sentences are selected; additionally, some adjacency constraints apply to enhance readability

[12] A relevant sentence is defined as either being a "direct match" or part of a "direct join", where a manually produced sentence was constructed by putting together a number of sentences from the original. In the latter case, *all* other members of this join must also have been produced by the system.

[13] Quoted from (Kupiec et al., 1995).

An evaluation showed that human readers tend to prefer summaries generated by a simple *lead*-method (i.e. just the first N words) instead of the "intelligently" produced summaries by the ANES system (Brandow et al., 1995). As it was shown by (Zechner, 1996), summaries generated by using term weight statistics outperform those generated by a *lead* method in terms of recall and precision (as compared to a human made gold standard). Thus there is an apparent trade-off between relevance and readability in the area of automatic abstracting.

6.2.1.4 Other approaches to summarization Other methods and ideas which could be employed for automatically generated text abstracts in a bottom-up approach were suggested by various authors, e.g.:

- Text-Tiling: The principal idea behind text tiling, as described e.g. in (Hearst and Plaunt, 1993), is to identify coherent passages in a given text, to find "topic boundaries", which is done by computing similarity of adjacent blocks of texts.
- Hidden Markov Models (HMMs): As (Mittendorf and Schäuble, 1994) describe in their paper, HMMs prove to be a mathematically sound framework for document retrieval and also can yield effective retrieval results. If one approaches the task of text abstracting from a such a probabilistic modeling perspective, it might well be possible that HMMs could be employed for this purpose, as well.
- Clustering: Building links and/or clusters between index terms, phrases and/or other subparts of the documents has been employed by standard IR for a long time (see e.g. (Bookstein et al., 1995)); although this is not an issue in any of the above mentioned abstracting systems, it seems to be worth of consideration when building such systems.

6.2.2 A System Using Text Understanding

The methods and systems we have looked at so far were purely statistical and heuristic in nature, they did not involve any understanding of the texts at all. However, as (Mauldin, 1989) points out, there appears to be sort of an "upper boundary" or limit for any system which solely uses keyword statistics (and, maybe, general heuristics). The main reasons of this "keyword barrier" are

- synonymy
- polysemy
- anaphora
- metaphors, metonymy

- context sensitivity

This means, that various forms of ambiguity, together with the problem of anaphora resolution must be dealt with, if one aims to achieve a system which performs significantly better than a purely statistically based keyword and heuristics approach would do. On the other hand, this would require the existence of a "global and common sense world knowledge" component to sort out all possibly occurring ambiguities in the analysis process. The complexities and intricacies of this task are the major reasons why all high-level systems built so far had to focus on a fairly restricted domain which at once exclude them for dealing with general domain texts, as it is necessary, e.g., for newspaper articles.

In the following, we will briefly look at the BREVIDOC system which attempts to do a thorough text analysis before generating a summary.

6.2.2.1 The BREVIDOC full-text retrieval system by Miike *et al.* (1994)

(Miike et al., 1994) present a system named BREVIDOC which is a full-text retrieval system that enables the user to specify an area within the text for abstraction and also to control the volume of the abstract interactively. BREVIDOC builds heavily on Rhetorical Structure Theory (RST), developed by (Mann et al., 1989). The heart of the system is the Document Structure Analyzer which performs the following processes:

1. Document Organization Analysis: This process determines the "outer" structure of a document, i.e., the division into sections, their headings, and the paragraph structure of the sections.

2. Sentence Analysis: Here, the sentences are parsed by a morphological and a syntactic analyzer.

3. Text Structure Analysis: This process first extracts the rhetorical relations between sentences, by matching them against a set of connective expressions and building a sequence of sentence identifiers and relations. Then, a number of *if-then*-rules, which look at certain cue words and patterns, are applied for segmenting these sequences. Finally, binary trees which represent the rhetorical structure of the text are constructed and heuristic rules are used to select the most plausible tree candidate. This whole analysis is done for separate paragraphs first and then is performed again for inter-paragraph structures, using the rhetorical relation of the first sentence of each paragraph.

4. Semantic Role Extraction and Word Indexing: For each sentence in every document, a semantic role is extracted, by matching sentence-trees

to *if-then*-rule patterns. Eventually, for every semantic role, a word index is generated that is based on document IDs and words obtained by morphological analysis of sentences with that role.

When an abstract is to be produced, the abstract generation function looks at the analyzed text structure for each section, and then generates an abstract based on the relative importance of the rhetorical relations which were found therein. The system uses cumulative penalty scores for different types of relations, depending on whether the respective node is belonging to a right-, left-, or both-nucleus-relation and on which branch of the subtree it occurs.

6.2.3 An Abstracting Evaluation Study

(Black and Johnson, 1988) performed an interesting study comparing machine made extracts using two different methods (keyword resp. indicator phrase method) with abstracts created by humans. They tested 35 subjects and used two texts to produce extracts/abstracts from. Each subject had to read one abstract and one extract but was not told which one was which.

The results were that while the readers' *subjective* judgments indicated a clear superiority of the human generated abstracts, there was no marked difference in terms of conveying the essential information between extracts and abstracts (the latter was tested by a form-filling task after reading the abstract/extract). Interestingly, there was also no significant difference in reading time observed between abstracts and extracts. It thus appears that while extracts are less "fun" to read, they still fulfill the same purpose as abstracts do and do not even produce a noticeable time delay in digesting the information.

6.3 Spoken Language Summarization

While summarization of written language has been extensively studied in a variety of approaches and applications, research on summarizing speech is virtually inexistent so far. (Kameyama et al., 1996) claim to have developed an "Automatic Dialogue Summarizer", but in fact their system should be properly called an IE system, since it fills templates and does not generate a readable text output.

It is very likely that the need for both IE and summarization of spoken language will increase significantly in the near future, given the combination of the increasing availability of spoken text databases (e.g., broadcast news) and the continuous improvement of speech recognition accuracy, which is particularly crucial for spontaneous human-to-human dialogues in unrestricted domains.

The major challenges are at least twofold: First, one has to deal with both the imperfections of the speech recognizer and the extragrammaticalities and disfluencies showing up in spontaneous speech; secondly, moving from limited domain to open domain applications is probably even more difficult than it is for written language.

7 Conclusions and Future Perspectives

In line with the "paradigm shift" within the field of NLP, there has been a substantially increased interest of automatic, fast, and shallow processing of natural language documents in the past decade. Whereas in previous systems the main focus was on completeness and correctness (even in rare, exceptional cases) and on narrow (or even toy) domains, the needs are now for NL processing techniques of texts in unlimited domains, where most system components can be automatically trained and easily specialized and tuned to particular applications with a minimum of human effort.

The driving force of the TIPSTER program with its two main evaluation style conference series, TREC and MUC, have significantly shaped the scientific community, in terms of priorities, research paradigms, and approaches.

Another aspect is, that, maybe for the first time in history, a range of NLP technology is now on a standard where it can be deployed on the market; examples are the successful Web browsers, speech interfaces to software packages, automatic news welders, etc.

Although predicting future developments is always hard, it is very likely that the trends observed now in increasing amounts of on-line data of multi-modal form will also further push the needs for sophisticated NLP products that can "handle" these loads of information in a way that maximally satisfies the user's needs and takes away much of the burdens which he currently has to bear, such as dealing with information overload, finding the right kinds of information, extracting the passages and contents from documents he is interested in, or summarizing relevant information in a readable and customizable way.

8 References

Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Myers, and Mabry Tyson. 1995. SRI international FASTUS system: MUC-6 test results and analysis. In *Sixth Message Understanding Conference (MUC-6)*, pages 237–246.

W.J. Black and F.C. Johnson. 1988. A practical evaluation of two rule-based automatic abstracting techniques. *Expert Systems for Information Management*, 1(3):159–177.

A. Bookstein, S. T. Klein, and T. Raita. 1995. Detecting content-bearing words by serial clustering — extended abstract. In *Proceedings of the 18th ACM-SIGIR Conference*, pages 319–327.

Ronald Brandow, Karl Mitze, and Lisa F. Rau. 1995. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 31(5):675–685.

Eric Brill. 1994. Some advances in transformation-based part of speech tagging. In *Proceeedings of AAAI-94*.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

H. P. Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM*, 16(2):264–285, April.

David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert. 1996. Description of the UMass system as used for MUC-6. WWW document, CCIR, Univ. of Mass.

Ralph Grishman. 1995. The NYU system for MUC-6 or: Where's the syntax? In *Sixth Message Understanding Conference (MUC-6)*, pages 167–175.

U. Hahn. 1989. Making understanders out of parsers: semantically driven parsing as a key concept for realistic text understanding. *International Journal of Intelligent Systems*, 4(3):345–393.

Donna Harmann. 1993. Overview of the first TREC conference. In *Proceedings of the 16th ACM-SIGIR Conference*, pages 36–47.

M.A. Hearst and C. Plaunt. 1993. Subtopic structuring for full length document access. In *Proceedings of the 16th ACM-SIGIR Conference*, pages 59–68.

O. Herzog and C.-R. Rollinger, editors. 1991. *Text understanding in LILOG. Integrating computational linguistics and artificial intelligence. Final report on the IBM Germany LILOG–project.* Springer Verlag, Berlin, Germany.

Lynette Hirschman, Martha Palmer, John Dowding, Deborah Dahl, Marcia Linebarger, Rebecca Passonneau, François-Michel Lang, Catherine Ball, and Carl Weir. 1989. The PUNDIT natural-language processing system. In *Proceedings of the Annual AI Systems in Government Coference, Washington, DC*, pages 234–243.

Jerry R. Hobbs, Douglas E. Appelt, John S. Bear, David J. Israel, and W. Mabry Tyson. 1992. FASTUS: A system for extracting information from natural language text. Technical report, SRI International, Menly Park, CA, November.

Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1996. FASTUS: A cascaded finite-state transducer for extracting information from natural language text. WWW document, SRI International.

P.S. Jacobs and L.F. Rau. 1990. Scisor: Extracting information from on-line news. In *Communications of the ACM, 33 (11)*, pages 88–97.

M. Kameyama and I. Arima. 1994. Coping with aboutness complexity in information extraction from spoken dialogues. In *Proceedings of the ICSLP 94, Yokohama, Japan*, pages 87–90.

Megumi Kameyama, Goh Kawai, and Isao Arima. 1996. A real-time system for summarizing human-human spontaneous spoken dialogues. In *Proceedings of the ICSLP-96*, pages 681–684.

Thomas S. Kuhn. 1970. *The structure of scientific revolutions.* University of Chicago Press, Chicago, 2nd edition.

J. Kupiec, J. Pedersen, and F. Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th ACM-SIGIR Conference*, pages 68–73.

Julian Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225–242.

Alon Lavie. 1996. *GLR*: A Robust Grammar-Focused Parser for Spontaneously Spoken Language.* Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

David B. Lewis and Karen Sparck-Jones. 1996. Natural language processing for information retrieval. *Communications of the ACM*, 39(1):92–101, January.

H. P. Luhn. 1958. The automatic creation of literature abstracts. In *IBM Journal of Research and Development*, pages 159–165.

William C. Mann, Christian M.I.M. Matthiessen, and Sandra A. Thompson. 1989. Rhetorical structure theory and text analysis. Technical report, ISI Research Report ISI/RR-89-242, November.

Michael L. Mauldin. 1989. Information retrieval by text skimming. Technical report, Carnegie Mellon University, Pittsubrgh, PA, CMU-CS-89-193.

Seiji Miike, Etuso Itoh, Kenji Onon, and Kazuo Sumita. 1994. A full-text retrieval system with a dynamic abstract generation function. In *Proceedings of the 17th ACM-SIGIR Conference*, pages 318–327.

E. Mittendorf and P. Schäuble. 1994. Document and passage retrieval based on hidden markov models. In *Proceedings of the 17th ACM-SIGIR Conference*, pages 318–327.

John D. Prange. 1996. Evaluation driven research: The foundation of the TIPSTER text program. In *Proceedings of the TIPSTER text program phase II workshop*, pages 13–21.

Yan Qu, , Carolyn P. Rosé, and Barbara Di Eugenio. 1996. Using discourse predictions for ambiguity resolution. In *Proceedings of COLING-96*, pages 358–363.

G. J. Rath, A. Resnick, and T. R. Savage. 1961. The formation of abstracts by the selection of sentences. *American Documentation*, 12(2):139–143.

Gerard Salton and Chris Buckley. 1990. Flexible text matching for information retrieval. Technical report, Cornell University, Department of Computer Science, TR 90-1158, September.

G. Salton and M.J. McGill. 1983. *Introduction to Modern Information Retrieval.* McGraw Hill, Tokyo etc.

Beth M. Sundheim. 1995. Overview of the results of the MUC-6 evaluation. In *Sixth Message Understanding Conference (MUC-6)*, pages 13–31.

P. Thompson. 1994. Concept based information extraction. *Integrated Computer-Aided Engineering*, 1(6):537–546.

C. J. van Rijsbergen. 1979. *Information Retrieval.* Butterworths, London, 2nd edition.

Wolfgang Wahlster. 1993. Verbmobil — translation of face-to-face dialogs. In *Proceedings of MT Summit IV, Kobe, Japan.*

Alex Waibel, Michael Finke, Donna Gates, Marsal Gavalda, Thomas Kemp, Alon Lavie, Lori Levin, Martin Maier, Laura Mayfield, Arthur McNair, Ivica Rogina, Kaori Shima, Tilo Sloboda, Monika Woszczyna, Torsten Zeppenfeld, and Puming Zhan. 1996. JANUS-II - advances in speech recognition. In *Proceedings of the ICASSP-96*.

Wayne Ward. 1994. Extracting information in spontaneous speech. In *Proceedings of the ICSLP 94, Yokohama, Japan*, pages 83–86.

Klaus Zechner. 1996. Fast generation of abstracts from general domain text corpora by extracting relevant sentences. In *Proceedings of COLING-96*, pages 986–989.