Designing a System for Swedish Spoken Document Retrieval

Botond Pakucs[1,2] **Björn Gambäck**[1,3]

botte@speech.kth.se gamback@sics.se

[1] Information and Language Engineering
Swedish Institute of Computer Science
Box 1263
S-164 29 Kista, Sweden

[2] Centre for Speech Technology
Royal Institute of Technology
Drottning Kristinas väg 31
S-100 44 Stockholm, Sweden

[3] Computational Linguistics
University of Helsinki
P.O. Box 4
SF-00014 Helsinki, Finland

Abstract

It is only during the last few years that attention has started to shift from pure text-based retrieval towards other media. Information retrieval from spoken documents is analogous to text-based retrieval; however, accessing audio documents causes some extra problems, in particular with respect to document segmentation, choice of indexing features, and robustness. In addition, retrieval of documents in Swedish, like most non-English languages, adds the extra dimension of morphology; also, when analysing spoken Swedish data, prosodic patterns have to be taken into account. In this paper we introduce SIREN, the Swedish Information Retrieval Engine, a very flexible, modular IR system which has been designed with a specific eye towards these issues.

1 Introduction

The field of information retrieval (IR) has been moving steadily forward for several decades. During the 90's we have seen several major break-throughs. Until recently, most of the work has been focused on texts; not only has most of the material processed been in text format, but even when other media such as audio and video have been considered in a system, text has been the primary concern. Consequently, most multimedial retrieval systems are modifications of existing text-based IR systems, disregarding the particular problems caused by the new media types. However, the amount of material in other formats increase all the time, increasing the need for tools that handle this information both from a system-oriented and a user-oriented perspective. An important issue for information management is how to represent these objects and collections of objects to best support retrieval. Another issue is to let users search in several modalities. An example would be when a person wants to search a news database: the database contains both news in text format and audio sequences of spoken material. These are stored and indexed in different ways, but the user wants to search both archives at the same time.

The term 'Spoken Document Retrieval' (SDR) has, in itself, rendered some confusion. We will use it exclusively for the particular case of information retrieval, when the information is to be retrieved from large volumes of spoken documents. Thus, what media the query itself is formulated in is of no importance to us here; only the format from which the sought information is to be accessed. In Section 2 we review the difficulties caused by trying to access multimedial documents, in particular audio documents, and some previous attempts to overcome them. We are building a flexible toolkit specifically designed to allow for different approaches to addressing these difficulties and for handling data in different types of media. The toolkit is functional but still open to improvement; however, Section 3 discusses the underlying design philosophy.

2 Problematic Issues

An obvious difference between information retrieval from written and spoken documents is that the written ones actually are physical documents, while the boundaries of the spoken ones, in contrast, have to be decided on inside larger audio files possibly containing quite diverse information. We will start out by looking at this segmentation issue and also at some other common problems in non-text-based retrieval, in particular with respect to choice of indexing features and robustness.

2.1 Document Segmentation

Many research projects, such as the ones in the TREC SDR Track (Garofolo *et al.*, 1997), have left document segmentation aside and instead manually tagged and segmented the documents. A straight-forward, automatic approach is to break down the audio documents into equally long time slices (windows) and treat the windows as individual documents (Schäuble and Wechsler, 1995; Smeaton *et al.*, 1998).

However, the linguistically most correct solution would be to automatically achieve meaningful segmentation of the audio documents through analysis of the prosodic information inherent in the audio documents, as done by Falk (1997). Unfortunately, not much work has been done on prosody in speech recognition. So far, the major success story was in the German spoken dialogue machine translation system *Verbmobil* (Bub *et al.*, 1997), where prosodic information was used in the semantic processing and transfer steps in order to produce correct translations (Lieske *et al.*, 1997). The input to a spoken dialogue system is structured in terms of turn-taking in the dialogue. Syntactic and semantic analyses can be assigned to meaningful linguistic entities at the clausal or phrasal level, but first the turn has to be segmented into a sequence of linguistically credible segments. The prosodic indications of clausal boundaries are crucial to this process. The prosody module of the *Verbmobil* system annotates the word lattices which have been output from the recogniser with three different kinds of prosodic information: sentence modality, phrase boundaries, and stress (Hess *et al.*, 1996).

Stolcke *et al.* (1999) introduces a combined probabilistic and prosody-based approach for recognising topic and sentence boundaries in speech. They employ a hidden Markov model (HMM) to find the most likely states given the recognised words and prosody (duration of pauses, etc., as well as pitch level, that is, $F0$ patterns; see below). In a first step the speech signal is "chopped" into small "sentence" segments which are assumed to belong to one topic each. In a second step the sentences are combined into continuous stretches belonging to one topic so that the sentence boundaries are classified according to whether they represent a topic shift or not (Hakkani-Tür *et al.*, 1999).

A somewhat similar but purely probabilistic segmentation technique is used by Delacourt *et al.* (1999) for recognising which persons are engaged in a conversation. They try to segment the speech data at every point in which a speaker change occurs. This is done in a two-step process where the first step applies a measure function reflecting how similar two adjacent segments are (in this case with respect to melcepstral coefficients). Dissimilarity indicates speaker change. In the second step a Bayesian likelihood criterion is applied to each changing point candidate to verify the results of the first step.

The prosodic issues are also very important when analysing spoken data from a tonal-type language like Swedish; the stress patterns, in particular, simply have to be taken into account. Within an utterance, the frequency may take on four relative values depending on the accent level of a phoneme. This may be defined either in absolute terms (due to, for example, the sentence accent, or the stress level within a word), or in terms relative to the frequency of the preceding phoneme.

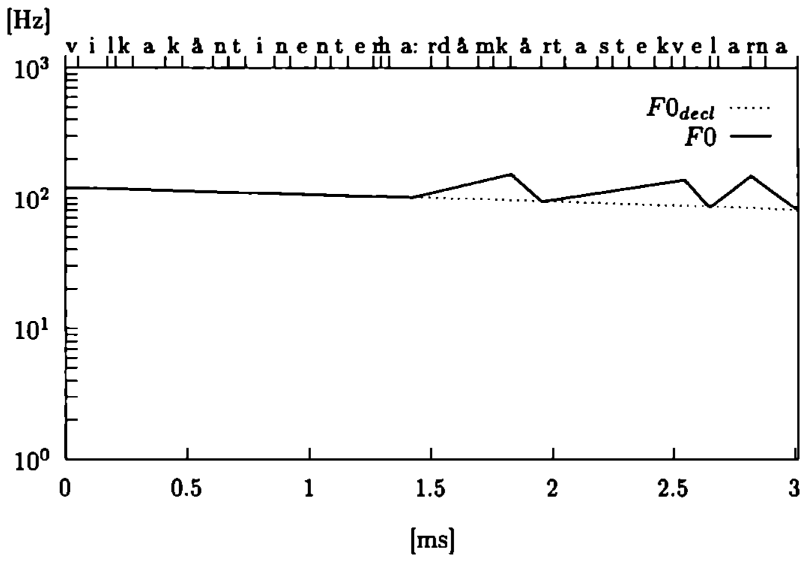

Figure 1: $F0$ and $F0$-declination functions

As described in more detail in, for example, Gambäck *et al.* (1995), the fundamental frequency contour ($F0$) is produced by superimposing the $F0$-variation on the $F0$-declination. The time points for the vocal chord excitations are then calculated from the resulting frequency function. The relation between $F0$, the variation and the declination is defined as

$$F0(t) = F0_{decl}(t) * F0_{var}(t)$$

As a concrete example, take the values that would be produced for the sentence *Vilka kontinenter har de kortaste kvällarna?*. Suppose that $F0_{decl}(t)$ starts at 120 Hz and ends at 80 Hz. $F0(t)$ is 1.8 times the declination at the time point $t = 2817$ ms and 1.6 times the declination at $t = 1827$ ms and $t = 2544$ ms. $F0(t)$ returns to the declination curve at $t = 1953$ ms, $t = 2649$ ms and $t = 3013$ ms. The function thus gets the form shown in Figure 1.

2.2 Indexing Features

In text-based information retrieval methods, the most obvious and frequently used indexing features are the words. This approach is also the dominating one in spoken document retrieval systems. However, in spite of rapidly improving speech recognition techniques, automatically produced transcriptions contain a multitude of errors and operate with limited dictionaries.

2.2.1 Word-based methods

For small domains, it is possible to select *a priori* fixed sets of keywords for recognition and indexing. This is, however, not feasible if large and diverse domains are covered. In such situations, a possible solution is to use a large-vocabulary word-based speech recogniser to convert the audio data to text and then filter the transcriptions through a language understanding system to reduce the recognition errors, as in the CMU *Informedia* project (Hauptmann, 1995). Still, when dealing with large and diverse domains of spoken documents, there are just too many words for current speech recognition systems to handle efficiently. In addition, out-of-vocabulary (OOV) words tend to be proper names or technical terms which generally have low frequencies in document collections. Thus, it is difficult to train speech recognisers for them, even though these word types are very common in user queries. This has lead to the investigation of subword unit approaches.

2.2.2 Subword unit approaches

The drawback with the use of subword indexing features is that elaborated information retrieval and linguistic techniques, such as stop word elimination and morphological analysis, are not suitable. This is specially cumbersome when the spoken information is in a language with more serious morphological properties than English. However, Ng and Zue (1997) showed that with appropriate subword units, it is possible to achieve performance comparable to that of word-based retrieval methods, if the underlying phonetic units are recognised correctly. Both syllable- and phoneme-based indexing features have been tried.

Syllabic units as indexing features was proposed by Glavitsch and Schäuble (1992). These units are composed of non-overlapping, variable-length letter sequences, CVC-features (where V and C stand for the maximum sequence of vowels and consonants, respectively, within a specific word). The syllabic units can be generated with a rule-based system (Glavitsch *et al.*, 1994). A problem with this is that the CVC-features are not based on acoustic data and do not take the characteristics of the recognition system into consideration.

n-grams, that is, overlapping, *fixed-length* phonetic sequences are the most straightforward phoneme-based subword units. With large enough length-units n, cross-word features can be captured. As indicated by, for example, Ng and Zue (1997), trigrams are normally long enough to capture information sufficient for indexing, while not being so long that they corrupt retrieval performance.

m-grams, non-overlapping, *variable-length* phonetic sequences with some maximum length m, can be discovered automatically by applying iterative unsupervised learning algorithms. Wechsler (1995) used an efficient algorithm to find variable length phone sequences for indexing. After frequency analysis, low frequency sequences were discarded, while high frequency sequences were extended. In this way an indexing vocabulary covering most parts of the phonetic transcription was obtained.

bclass, broad phonetic class sequences, are derived via unsupervised hierarchical clustering of the original phones. Acoustically similar phones are grouped into the same class, so that some of the typical speech recognition errors can be avoided. Ng and Zue (1997) claimed that even after collapsing the number of phones from 41 down to 20 bclasses, enough information is preserved to perform reasonable retrieval: Given perfect recognition, they got a precision of 0.82.

2.3 Robustness

The robustness issues facing spoken document retrieval systems are similar to when text retrieval systems encounter optical character recognition errors in scanned text documents. Thus, even given an ideal speech recognition component, low quality audio data will still effect the speech recognition accuracy and thus the retrieval performance. Accordingly, tolerance of the retrieval models to different disturbing factors such as noisy speech, conversational speech, telephone bandwidth speech, and multiple speakers is important. Quite a few solutions have been proposed to cope with speech recognition errors caused by low quality audio data; however, in the area of robust methods for spoken document retrieval, there is still much left to be done.

In word-based systems, OOV-words can cause significantly weaker retrieval results. THISL (Abberley *et al.*, 1998) applies query-time word-spotting based on posterior probability estimates derived from the recurrent network acoustic model, in order to detect index terms not in the recogniser's vocabulary. Jones *et al.* (1996) used a continuous-speech large vocabulary recognition system in combination with the phone-lattice-based word-spotting method of James (1995). They showed that the two methods are complementary and work best in combination.

In phoneme-based solutions, one possibility is to expand the query with errorful variants of the original terms, in order to improve the chance of matching wrongly recognised terms. A similar method is to expand the spoken document representation by including high scoring recognition alternatives to increase the retrieval precision. Ng (1998) tried to combine the various types of information captured by the different subword unit representations. He obtained a marginal improvement in retrieval performance by using n-best recognition hypotheses and linear combination of the individual retrieval scores obtained with different subword unit methods.

Crestani and Sanderson (1997) aimed at taking advantage of some particular features of the automatic transcriptions. They used the *Abbot* speaker independent continuous speech recognition system which associates a measure of uncertainty to each word it recognises (Robinson *et al.*, 1996). These measures were used in a probabilistic term frequency weighting scheme for improving retrieval efficiency. The results were encouraging, even though the performance was very bad on some particular queries. Later, they attempted to merge the transcriptions produced by multiple recognisers and slight improvements in retrieval performance were achieved (Sanderson and Crestani, 1998).

2.4 Retrieval result presentation

Presenting multimedial retrieval results to the users introduces some new problems into the IR field. Determining whether a retrieved text document contains an answer to a query is quickly done by simple browsing. However, browsing information objects such as TV or radio news broadcasts can be rather cumbersome due to the sequential nature of the video/audio files.

Having decided on what piece of information to retrieve, we are still far from having decided on *how* to present the information. Most interfaces to text-based retrieval systems simply show the user a list of documents, sorted according to some kind of relevance measure, which often is incomprehensible to the user. A notable exception is the SICS–Telia prototype *Easify* where search results are presented as matrixes, rather than lists (Bretan *et al.*, 1998). *Easify* applies a machine learning method to create collection specific stylistics for genre prediction together with rapid topical clustering in order to represent documents as members of topically and stylistically homogeneous clusters.

Present multimedial retrieval systems have no, or fairly simple, graphical user interfaces. *VoiceGraph* is a graphical interface developed for audio and multimedia retrieval (Slaughter *et al.*, 1998). The interface was designed to facilitate rapid browsing of spoken documents. Another, web-based, graphical interface was developed for the *Taiscéalaí* radio broadcast retrieval system (Smeaton *et al.*, 1998). In *Taiscéalaí*, the retrieved documents are presented as time series of window scores. The users are allowed to select parts of a broadcast. The selections are delivered from the archive and transformed to a RealAudio file which is played to the user.

3 System Design

Most multimedial retrieval systems are modifications of existing text-based IR systems, disregarding the particular problems caused by the new media types. However, in order to manage the problems presented in Section 2 we need to test new solutions and experiment with several different algorithms. So instead of using existing text-based platforms we decided to design and develop a new flexible IR system. The requirements for a retrieval toolkit can be formulated as:

- A *modular* design is desirable: it should be easy and fast to make small changes to specific parts of the toolkit without reimplementing the whole system.
- The system must be *extensible* and *open*, since we want to be able to experiment with new algorithms and new media types. Thus, adding new modules has to be fairly straight-forward.
- The toolkit has to run *dynamically* and *flexibly*, allowing for quick adaptations to new requirements at any time by connecting/disconnecting modules.

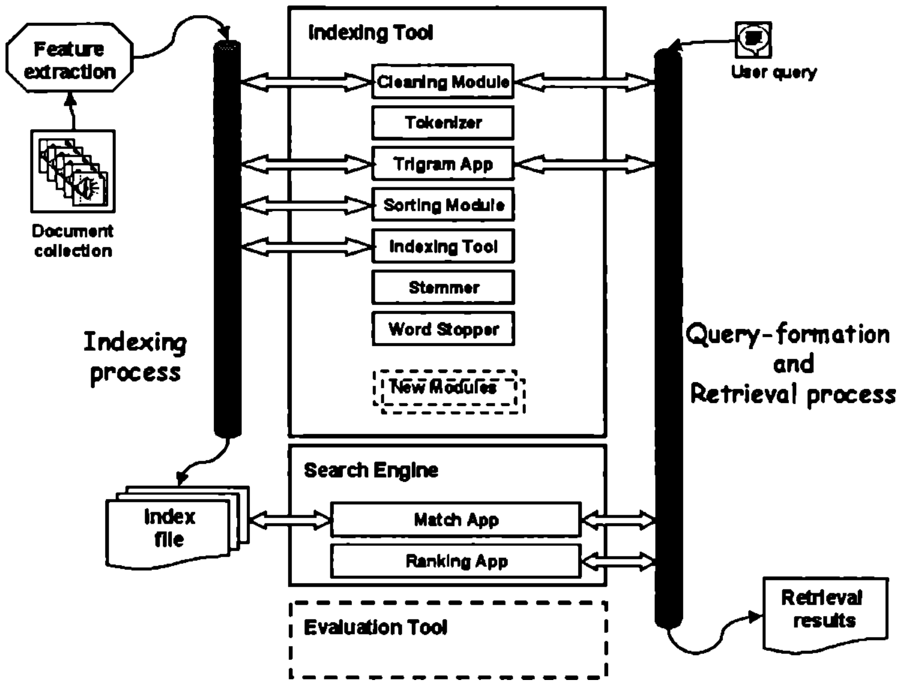

Figure 2: The design of SIREN

- Another desirable quality is *transparency*: we might want to examine the data flow under certain conditions.
- Due to the distributed nature of digital libraries, *interoperability* within a library architecture is necessary. To achieve a high level of access and sharing, we need a general framework for handling information across various domains such as different communities and different types of information objects.
- *Portability* is not necessarily a major requirement, but in itself a desirable quality of software systems.

We chose the object-oriented Java environment in order to address the portability issue.[1] When implementing the search engine, we used the built-in `Thread` class features, allowing us the option of adapting the system to the use of parallel search threads for searching multiple index files. This is essential when several media types are searched simultaneously.

The two major components of our IR toolkit are the indexing tool and the search engine as shown in Figure 2. (A third component, an evaluation tool, is planned.) The IR solutions and algorithms are based on well-founded results: We use a common inverted-file structure for storing indexing and posting information. The index-term weighting scheme is the *tf*idf* formula. The search engine implements the vector-space model and the cosine metric matching function. Thus, the novelty of the approach lies not in the

[1]Certainly, the choice of programming language is crucial for system performance, and Java is not well-known for its speed qualities. However, it is still under development, promising better speed properties. Also, modular system design coupled with the Java environment makes reimplementation of vital components in more time-effective languages tractable.

algorithms used, but in the way these algorithms are implemented in the system, and in the system design which allows for fast reimplementation and integration of new modules.

3.1 The modules

The main tools are built up of several small, independent modules, each of them taking care of one well-defined atomic task. The main advantage of this modular design is the possibility of reusing the modules in both the indexing and the query-formation process.

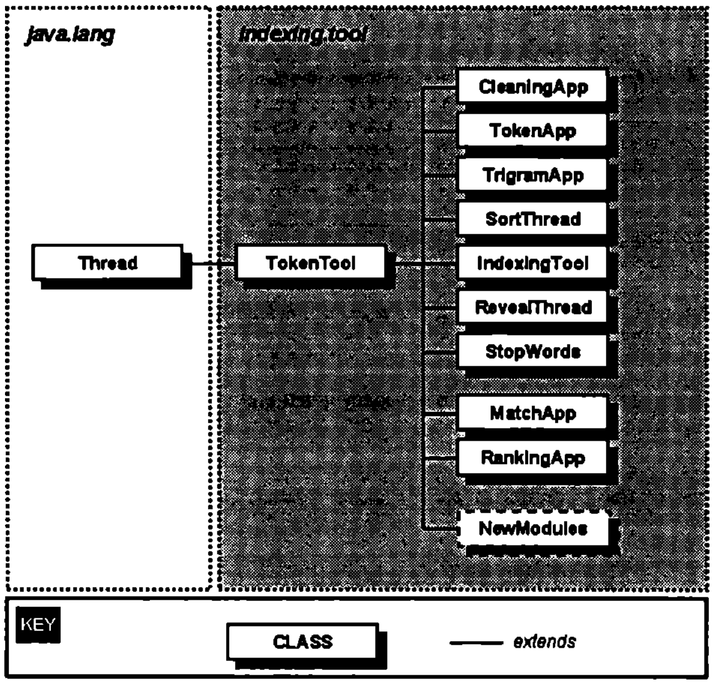

Figure 3: The SIREN class structure

The modules are implemented as independent Java classes as shown in Figure 3. Some examples of the tasks performed by the indexing tool: parse a document into a word-based token stream; parse a document into a trigram-based token stream; remove stop-words from the stream; perform morphological analysis; sort the stream; index the token stream.

As indicated in Figure 2, the modules are connected together in a pipelined architecture, using the Java built-ins `PipedReader` and `PipedWriter` for input and output, respectively. In this way, pipe syncronisation is taken care off automatically. The connections between the modules are dynamic, so that modules can be invoked "on the fly". So could, for example, the indexing tool be called with the following command sequence

```
java Siren clean token sort index words.txt
```

which would lead to the file `words.txt` being passed through the cleaning, tokenisation, sorting, and indexing modules. Modules can even behave in a mutually interchangeable way, if they perform similar tasks. Thus, it is possible to change the indexing feature from words to trigrams by just changing the token command to trigram when invoking the tool as above.

A specific parser was implemented for reading the command lines. This tool scans the input stream and activates the corresponding modules. The order in which the commands are written on the line is critical. The system runs the modules in the same order as the command line indicates. Running modules in a faulty order does not result in system failure, but probably makes no sense in the overall indexing or retrieval process. In the worst case, the index file could be updated with wrong index terms. The parser accepts two different types of commands. The simple, one-word command calls one specific module which is operating on the data stream. The second type of command needs an additional argument which is assumed to be a file name. That file may be used for either input or output, depending on the specific module's design and purpose.

An advantage of the piped architecture is that the need of using and saving intermediate files is eliminated. Implementing and integrating new modules is fairly straight-forward. A new module is implemented as a new Java class and can take advantage of the built-in token language. To integrate a new component, only new method calls have to be declared, as long as the module supports PipedReader and PipedWriter.

The main requirement on the modules is to extend the TokenTool class described in the next section. The modules must also implement a run method. In addition, each module needs to have a corresponding calling method. These methods have to be declared just once. The system keeps track of the available methods through the Java built-in Reflect class.

The system is *open*, thus, the pipes are open at both ends and adding new modules with specific preprocessing tasks, such as prosodic analysis is possible, as is also connecting the pipe directly to a speech recogniser. Appending new modules for relevance feedback or retrieval evaluation is also feasible.

3.2 The token language

The modules are communicating through a common token stream, a solution slightly inspired by University of Glasgow's SIRE system (Sanderson and Crestani, 1998). The token stream consists of simple ASCII text. The modules are operating on the stream by modifying the tokens, by adding new tokens, or by removing some of them. Each token holds one term and a number of additional attributes associated to it. The sorting module assumes that the first component in every token is the indexing term. The modules are not sensible to the order of the other attributes. A small example of a token stream is given in Table 1.

The first column contains the indexing terms. In this example, a list of sorted trigrams. The second attribute is an internal tag and carries information about the type of the term. Here the tg tag indicates that the terms are in trigram-form. The typing is not a strong typing. Any character combination is allowed except for tab and newline. The number following indicates the term's position in the original file (byte offset). The last component is the term's *tf* score in the document. Adding new attributes can be easily done by just appending the tab separator and the desired piece of information. In the above example, we might include some information about the index-terms' weight scores.

Table 1: An example token stream

term	**tag**	**offset**	***tf***
tha	tg	1034	2
the	tg	1010	19
thi	tg	950	1
tic	tg	1066	3
tig	tg	927	1
tin	tg	323	2
tio	tg	1006	6
tis	tg	274	1
tit	tg	136	1
tl'	tg	572	2
tly	tg	369	2
tme	tg	82	1
toc	tg	293	1

The tokens are processed as they are by most of the modules. Converting the stream to an internal representation is needed in just a few cases, for example, in the sorting module. This is a nice quality since it enhances execution performance. By using pipes as the linking method, it is possible to inspect the data flow between two different modules by calling a specially designed `RevealThread` module which saves or prints a copy of the data flow.

Due to the ASCII nature of the token stream, the resulting data can be manipulated as plain text. However, the underlying data representation is hidden from the modules in a object-oriented manner. Specially built `TokenTool` class methods provide a better abstraction level in manipulating the attributes connected to the tokens. Thus, the modules are separated from the data representation and rather use general IR concepts, such as index-term, weight, frequencies, etc. This solution allows a more general implementation of the IR modules and operations, so that data manipulation is independent of the addressed media type. In addition, the system implementation also allows for an eventual redesign and reimplementation of the data representation without major impacts on the system design and on the system components.

4 Conclusions / Future Work

We have described SIREN, a flexible, modular information retrieval system designed to allow for different approaches to addressing the particular problems which are encountered when attempting to access non-text documents. The toolkit is already functional and useful for text and spoken document retrieval. New modules with specific tasks will soon be added, such as stylistic analysis and name recognition.

References

Abberley, D., Renals, S., and Cook, G. 1998. Retrieval of Broadcast News Documents with the THISL System. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, Seattle, Washington. IEEE.

Bretan, I., Dewe, J., Hallberg, A., Wolkert, N., and Karlgren, J. 1998. Web-Specific Genre Visualization. In *Proc. 3rd World Conference on the WWW and Internet*, Orlando, Florida. AACE.

Bub, T., Wahlster, W., and Waibel, A. 1997. Verbmobil: The Combination of Deep and Shallow Processing for Spontaneous Speech Translation. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pp. 71–74, München, Germany. IEEE.

Crestani, F. and Sanderson, M. 1997. Retrieval of Spoken Documents: First Experiences. Technical report, Dept. of Computing Science, University of Glasgow, Scotland.

Delacourt, P., Kryze, D., and Wellekens, C. J. 1999. Speaker-based Segmentation for Audio Data Indexing. In Robinson, T. and Renals, S., eds., *Proceedings of the Workshop on Accessing Information in Spoken Audio*, pp. 78–83, Cambridge, England. ESCA.

Falk, J. 1997. Pauses in Synthesized Speech: Automatic Prediction of Silent Intervals in Swedish. Master of Art Thesis, Dept. of Linguistics, Göteborg University, Sweden.

Gambäck, B., Eineborg, M., Eriksson, M., Ekholm, B., Lyberg, B., and Svensson, T. 1995. A Language Interface to a Polyphone-Based Speech Synthesizer. In *Proc. 4th European Conference on Speech Communication and Technology*, volume 2, pp. 1219–1222, Madrid, Spain. ESCA.

Garofolo, J. S., Voorhees, E. M., Stanford, V., and Jones, K. S. 1997. TREC–6 1997 Spoken Document Retrieval Track Overview and Results. In Voorhees, E. M. and Harman, D. K., eds., *Proceedings of the 6th Text Retrieval Conference*, pp. 83–91, Gaithersburg, Maryland. National Institute of Standards and Technology.

Glavitsch, U. and Schäuble, P. 1992. A System for Retrieving Speech Documents. In Belkin, N., Ingwersen, P., and Mark Pejtersen, A.-L., eds., *Proc. 15th International Conference on Research and Development in Information Retrieval*, pp. 168–176, København, Denmark. ACM SIGIR.

Glavitsch, U., Schäuble, P., and Wechsler, M. 1994. Metadata for Integrating Speech Documents in a Text Retrieval System. *SIGMOD Record*, **23**(4):57–63.

Hakkani-Tür, D., Tür, G., Stolcke, A., and Shriberg, E. 1999. Combining Words and Prosody for Information Extraction from Speech. In Prószéky, G., Németh, G., and Mándli, J., eds., *Proc. 6th European Conference on Speech Communication and Technology*, volume 5, pp. 1991–1994, Budapest, Hungary. ESCA.

Hauptmann, A. G. 1995. Speech Recognition in the Informedia^{TM} Digital Library: Uses and Limitations. In *Proc. 7th International Conference on Tools with AI*, Washington, DC. IEEE.

Hess, W., Batliner, A., Kiessling, A., Kompe, R., Nöth, E., Petzold, A., Reyelt, M., and Strom, V. 1996. Prosodic Modules for Speech Recognition and Understanding in Verbmobil. In Sagisaka, Y. *et al.*, eds., *Computing Prosody: Approaches to a Computational Analysis and Modelling of Prosody of Spontaneous Speech*, pp. 363–384. Springer, New York, New York.

James, D. A. 1995. *The Application of Classical Information Retrieval Techniques to Spoken Documents.* Doctor of Philosophy Thesis, Downing College, Engineering Dept., University of Cambridge, England.

Jones, G. J. F., Foote, J. T., Sparck Jones, K., and Young, S. J. 1996. Retrieving Spoken Documents by Combining Multiple Index Sources. In Frei, H. *et al.*, eds., *Proc. 19th International Conference on Research and Development in Information Retrieval*, pp. 30–38, Zürich, Switzerland. ACM SIGIR.

Kokkinakis, G., Fakotakis, N., and Dermatas, E., eds.. 1997. *Proc. 5th European Conference on Speech Communication and Technology*, Rhodes, Greece. ESCA.

Lieske, C., Bos, J., Gambäck, B., Emele, M., and Rupp, C. 1997. Giving Prosody a Meaning. In Kokkinakis *et al.*, eds. 1997, pp. 1431–1434.

Ng, K. 1998. Towards Robust Methods for Spoken Document Retrieval. In *Proc. 5th International Conference on Spoken Language Processing*, Sydney, Australia.

Ng, K. and Zue, V. 1997. Subword Unit Representations for Spoken Document Retrieval. In Kokkinakis *et al.*, eds. 1997, pp. 1607–1610.

Nikolaou, C. and Stephanidis, C., eds. 1998. *Proc. 2nd European Conference on Research and Advanced Technology for Digital Libraries*, Heraklion, Greece.

Robinson, T., Hochberg, M., and Renals, S. 1996. The Use of Recurrent Networks in Continuous Speech Recognition. In Lee, C.-H. and Soong, F. K., eds., *Advanced Topics in Automatic Speech and Speaker Recognition*, chapter 7. Kluwer, Dordrecht, Holland.

Sanderson, M. and Crestani, F. 1998. Mixing and Merging for Spoken Document Retrieval. In Nikolaou and Stephanidis, eds. 1998, pp. 397–407.

Schäuble, P. and Wechsler, M. 1995. First Experiences with a System for Content Based Retrieval of Information from Speech Recordings. In *Proc. IJCAI Workshop on Intelligent Multimedia Information Retrieval.*

Slaughter, L., Oard, D., Warnick, V., Harding, J., and Wilkerson, G. 1998. A Graphical Interface for Speech-Based Retrieval. In *Proc. 3rd Digital Library Conference*, Philadelphia, Pennsylvania. ACM.

Smeaton, A., Morony, M., Quinn, G., and Scaife, R. 1998. Taiscéalaí: Information Retrieval from an Archive of Spoken Radio News. In Nikolaou and Stephanidis, eds. 1998, pp. 429–442.

Stolcke, A., Shriberg, E., Hakkani-Tür, D., Tür, G., Rivlin, Z., and Sönmez, K. 1999. Combining Words and Speech Prosody for Automatic Topic Segmentation. In *Proc. Broadcast News Workshop*, Herndon, Virginia. DARPA.

Wechsler, M. 1995. Eine neue Indexierungsmethode für Information Retrieval auf Audiodokumente. In *Proc. Hypertext-Information Retrieval-Multimedia*, pp. 117–128.