Extracting Semistructured Data - Lessons Learnt

Udo Kruschwitz, Anne De Roeck, Paul Scott, Sam Steel, Ray Turner, and Nick Webb

Department of Computer Science, University of Essex,
Wivenhoe Park, Colchester,
CO4 3SQ, United Kingdom
{udo, deroe, scotp, sam, turnr, webbnw}@essex.ac.uk**

Abstract. The Yellow Pages Assistant (YPA) is a natural language dialogue system which guides a user through a dialogue in order to retrieve addresses from the *Yellow Pages*[1]. Part of the work in this project is concerned with the construction of a *Backend*, i.e. the database extracted from the raw input text that is needed for the online access of the addresses. Here we discuss some aspects involved in this task as well as report on experiences which might be interesting for other projects as well.

1 Introduction

The YPA is a directory enquiry system which allows a user to access addresses and other information of advertisers in a classified directory [4]. A substantial part of the work on this project has been the extraction of data from the source files to create a *Backend* database. This source data is *semistructured*, in other words "data that is neither raw data, nor very strictly typed as in conventional database systems" [1]. A second notable fact is that we do not have access to vast amounts of data, our sources are in the range of a few megabytes. The solution is to construct a *Backend* such that the *online* query processing is midway between theorem-proving and *Information Retrieval (IR)*. The theorem-proving is about metafacts (e.g. relations between different indices) rather than entities like addresses or single index entries. In this paper we will describe aspects of the *Backend* construction and what has been learnt in this process.

The structure of this paper will be as follows. First we will give a general overview of the architecture of the YPA system (section 2). In section 3 we will point to related work. Section 4 focusses on the *Backend* component of the YPA. Finally we describe recent experiences and evaluation results (section 5) and conclude with an outline on future work (section 6).

** Nick Webb now at University of Sheffield (n.webb@dcs.shef.ac.uk)

[1] Yellow Pages® and Talking Pages® are registered trade marks of British Telecommunications plc in the United Kingdom

2 System Overview

The YPA is an interactive system for querying a database of addresses in a classified directory. A query can consist of simple keywords, phrases or can look like this:

I need the addresses of hotels with en suite bathrooms in Wivenhoe please!

Figure 1 is an overview of the system architecture (depicting the data flow).

A conversation cycle with the YPA can be roughly described as follows. A user utterance (recognised by the *Speech Recognition* or typed in via the *Graphical User Interface*) is sent to the *Dialogue Manager*. The *Dialogue Manager* keeps track of the current stage in the dialogue and controls the use of several submodules. Before handing back control (together with the relevant data) to the *Toplevel*, the input is first sent to the *Natural Language Frontend* which returns a so-called *slot-and-filler query*. The *Dialogue Manager* then consults the *Query Construction Component*, passing it the result of the parsing process (possibly modified depending on the *Dialogue History* etc). The purpose of the *Query Construction Component* is to transform the input into a *database query* (making use of the *Backend* and possibly the *World Model*), to query the *Backend* and to return the retrieved addresses (and some database information) to the *Dialogue Manager*. Finally the *Dialogue Manager* hands back control to the *Toplevel* which for example displays the retrieved addresses. It could also put questions to the user which were passed to it by the *Dialogue Manager*, if the database access was not successful (i.e. did not result in a set of addresses). At this stage the cycle starts again.

3 Related Work

If we look at the YPA as a system, then it is comparable with natural language dialogue systems. A lot of online dialogue systems have been described mainly for dealing with times or train schedules , for example [3, 18, 14, 10, 7]. This is of little interest for the YPA. However, dialogue systems have also been built for retrieving addresses from *Yellow Pages*, for example *Voyager* [24, 22, 6] and *Galaxy*, but the implementations are restricted to sample domains or small-scale address databases (e.g. 150 objects in *Voyager* [6] or 2400 tourist related listings in *Galaxy* [23]). This is of course not unrelated to the fact that these are *spoken* language dialogue systems.

However, the task of extracting and representing *semistructured* data within such a system can be tackled in different ways. Different communities approach the problem of dealing with this sort of data in different ways. From the perspective of database management systems (*DBMS*) there has been a research interest in how to represent this data as well as how to extract parts of it. The *Lore* database management system [11] is designed specifically for the management of semistructured data. The *Information Manifold* system [8] allows the

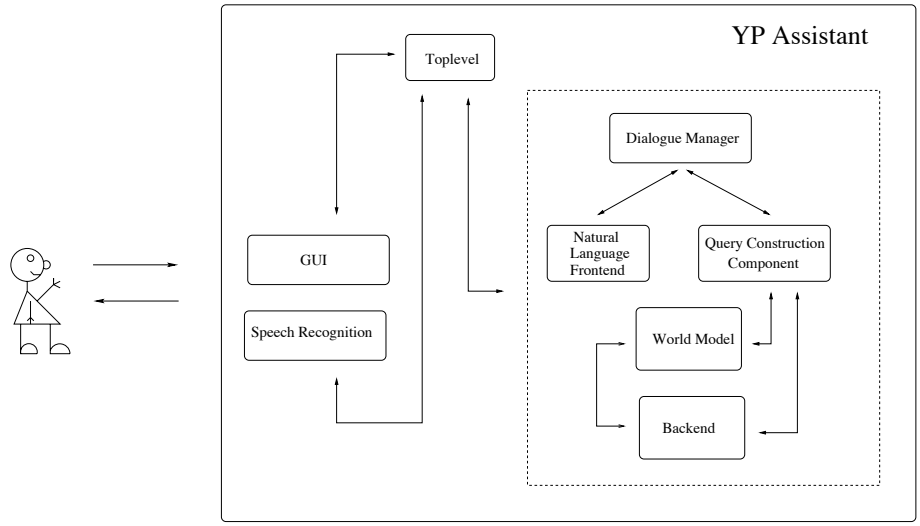

Fig. 1. Architecture of the YPA

access to data in heterogeneous sources. By declaratively describing the contents and the query capabilities it allows a uniform access to various resources.

On the other hand, a typical *IR* approach tackles the problem in quite a different way. The traditional task is to select a set of documents from a large collection so that the selected documents satisfy a user's query. It is not necessary to think about actually accessing semistructured data in an *online* system, but to preprocess this data *offline*. It is therefore reasonable to extract data from the corpus of documents that can then be represented by a traditional relational database system. The remaining problem is then *how* to represent the content of the source documents. The *NLIR* system [16] which is a *Natural Language Information Retrieval* system uses parallel indices which represent the same document using different indexing methods. Natural language processing (*NLP*) has been incorporated into *IR* systems (lexical processing, stemming, POS tagging etc.) [9, 15] but "NLP techniques used must be very efficient and robust, since the amount of text in the databases accessed is typically measured in gigabytes" [21]. *IR* systems usually rely very much on statistics. As we pointed out earlier, the amount of data in the YPA system is in the region of just megabytes. But we could abstract the YPA to an *IR* task involving *short queries* for the access of *short documents* which is quite different to classical *IR* tasks like those considered at TREC [17].

From this point of view there are at least two approaches that should be mentioned here. *Publishers Depot* [5] is a commercial product that allows the retrieval of pictures based on the corresponding caption.[2] That means the doc-

[2] `http://www.picturequest.com`

uments are very short compared to standard *IR* tasks. Some points are worth mentioning here as we are concerned with similar problems. First of all, *WordNet* [12] is the base for semantic expansion, i.e. synonyms, hypernyms etc. are used to broaden the search. For this purpose *WordNet* was tailored. Among other things, that included the deletion of "bad" words, the removal of odd links and addition of new terms which do not exist in the *WordNet* lexicon. Overall *WordNet* is seen as well suited for *IR* tasks like this.

On the other hand there has been work on *conceptual indexing* [20]. A knowledge base is built that is based on the observation that there are few true synonyms in the language and usually there is a generality relationship like: *car* –> *automobile* –> *motor vehicle* which can be expressed as a *subsumption* relationship. [20] employs an example from the *Yellow Pages*:

- The *Yellow Pages* do not contain a heading "automobile steam cleaning".
- There are basic facts (axioms): (1) a *car* is a *automobile* and (2) *washing* is some kind of *cleaning*.
- The algorithm has to infer that *car washing* is some sort of *automobile cleaning*.

Such a knowledge base has been applied to precision content retrieval knowing that standard *IR* approaches are not very effective for *short* queries [2].

Our approach can finally be placed somewhere between the one used in [5] and [20].

4 The Backend

4.1 Input Data

The *Backend* construction process takes the raw data (the so called *YP printing tape*) and creates a database that retains as much information and relations as possible for the online enquiry system. This was described in detail in [4]. The input data is *semistructured* in a sense that a record structure for the addresses and headings does exist but the internal structure of these entries is not formally defined. Usually this consists of partial English sentences, address information, telephone patterns etc. Several types of adverts exist: *free*, *line* and *semidisplay* entries with *semidisplay* adverts carrying more information than just the name and the address of the advertiser. Here is a typical *semidisplay* entry as it appears in the printed directory (in this case listed under *Hotels & Inns*):

UDO, THE

TOWN CENTRE BED & BREAKFAST
EN SUITE ROOMS-AMPLE PARKING
High St,Wivenhoe,01206 827735

The actual source code in the *printing tape* does not give us much more structured information than what can be seen in the printed advert:

```
195SS15SBZS9810289N955030800 0 015OUDO,THE^
195SS15SBZS9810289B9350208000C0     TOWN CENTRE BED & BREAKFAST^
195SS15SBZS9810289B9350208000C0     EN SUITE ROOMS-AMPLE PARKING^
195SS15SBZS9810289B9350208000C0    CHigh St,Wivenhoe,01206\$_827735^
```

When comparing this data with a typical user request it is often hard to find adverts which do actually satisfy the complete user query. In our initial example the user asked for *hotels with en suite bathrooms*, something which cannot be found in the list of adverts. But the above example is still a very good match.

We see the task of the dialogue system as narrowing the gap between a user request and what the index database of the system can supply. Figure 2 presents a different example for this task.

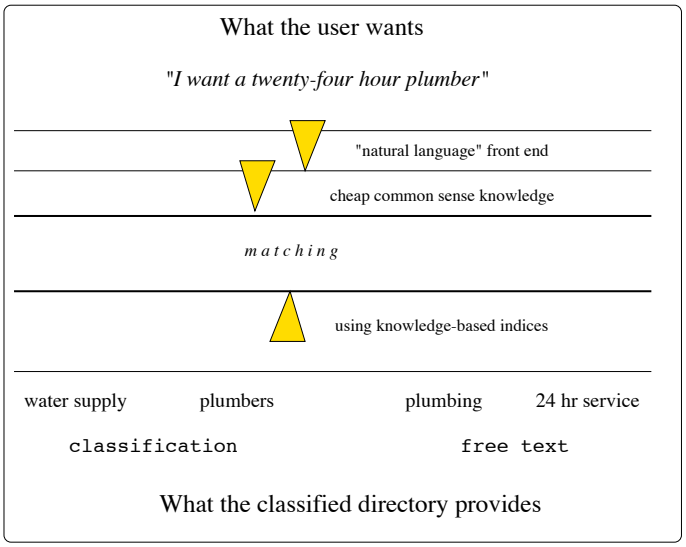

Fig. 2. Narrowing the Gap

Thus, the input data must be transformed into an appropriate representation which does allow to match user queries to adverts as good as possible combining various sorts of information like the name of the heading, the keywords in the free text of an entry (*TOWN CENTRE BED & BREAKFAST ...*) etc.

In order to obtain such a *Backend* database of indices we can distinguish two extreme positions for the construction process: *Knowledge Representation*,

i.e. express the data in logic and do theorem proving in the online YPA system or *Information Retrieval*, i.e. use morphology and string matching. The intermediate position that we have chosen extracts relatively flat relations from the incoming data and uses text-retrieval-like methods for this *offline* process, while we can still apply theorem-proving-like methods in the *online* system.

4.2 Data Representation

The *Backend* database which is compiled *offline* consists of a set of different tables. The indices in these tables are represented as normalised representations of (stemmed) base forms of simple keyword or phrases, but the relations between index tables can be used in the online system which goes beyond string matching. Threrefore we speak about *knowledge rich* indices. The performance of the retrieval depends on how well the *Query Construction Component* (YPA-QCC) and the *Dialogue Manager* (YPA-DM) exploit this knowledge.

For the above example the simplified information stored in the various tables for this advert includes the following: *location indices* ("wivenhoe"), *heading indices* ("hotels", "inns"), *business name indices* ("udo"), *free text indices* ("bed_breakfast", "en_suite_rooms" ...), *heading reference indices* (for "guest houses" see also the heading "hotels & inns"), *street indices* ("high_street").

An extract of this information as encoded in our Prolog database is shown here:

```
keyword_heading('hotel_inn', 33264).
keyword_heading('hotel', 33264).
keyword_heading('inn', 33264).

entry(33264, 33302, 'udo,the').

keyword_entry('bed_breakfast', 33302).
keyword_entry('bed_breakfast_suit', 33302).
keyword_entry('room_suit', 33302).
...

see_also(31136, 33264).
```

Some of the information is easy to find, for example the heading name and the references, but on the other hand information like the street name must be spotted within the free text, this is not marked.

With a database created like this it is now the responsibility of the YPA-QCC to decide how to combine the different index tables in the retrieval process. Several cases are possible given the following user request:

Are there any guest houses in Ipswich?

First of all, the list of adverts listed under *Guest Houses* could be displayed. But with the given reference to *Hotels & Inns* the list of addresses might be

extended. However, if this list becomes too big, should the user be given a choice of different headings to select from or should only be the *best* matching adverts displayed? What if all adverts have the same ranking?

Some strategies for the YPA-QCC can be triggered by different setup options available to the user. For example it might be that the user is interested in high *recall* rather than *precision*, in which case the strategy employed by YPA-QCC and YPA-DM will look different to the default strategy (in the example that could mean that adverts under the heading *Hotels & Inns* will always be retrieved when asked for *guest houses*).

4.3 Interaction with Other Components

We referred to the *Backend* as a database of *knowledge rich* indices. It remains the task of the other components in the system to actually apply this in a sophisticated way. In parallel to the work on the database creation we revised the *Query Construction Component* and the complete *Frontend* to be able to exploit the *Backend* knowledge. But how much does the *Backend* influence these components?

First of all, it turned out that we can still use a very flat and robust parser which then calls the slot filling process to group the information in a domain dependent second processing step (some slots are: *location*, *transaction* and *products & services*). That means only very general syntactic structure and almost no semantic information is preserved in the process of transforming the input query into some internal representation except the implicit semantics of the slots. This was explained in more detail in a separate paper about the *Frontend* [19]. For the *Query Construction Component* this flat structure seems sufficient, because to solve the task of retrieving information from the *Backend* the *Query Construction Component* mostly relaxes or constrains a database query and this is triggered by simple assumptions:

- Simple syntactic information (like the detection of prepositional phrases or modifiers) is exploited to relax queries. This works very efficiently.
- The semantics of the slots can be used to relax or constrain queries. This knowledge can be encoded declaratively, for example:
  - A slot *streetname* has some relation to a slot *location* in a way that relaxing a query can be achieved by emptying the *streetname* slot and leaving *location* untouched.
  - A slot *products & services* is more important than the *methods of payment* slot, this is of course part of the domain dependent customisation.
- Finally, other metafacts (knowledge about the *Backend* database) which are not encoded as semantics of slots can be used.

After all, the *Query Construction Component* is responsible to combine these flat but heterogenous knowledege sources in the retrieval process.

4.4 Applying the Backend Database

We want to report some experiences gained while working on the YPA which involve the usage of the *Backend*.

Initially we went for a high *recall* because a lot of queries could not be answered just because the query terms could not be mapped to the indices in the *Backend*. When testing the system with real users we soon noticed that a high *precision* is much more desirable than a high *recall*, something that has also been reported by [5]. Nothing seems more irritating for a user than addresses whose retrieval cannot be explained. As an example a user asked for *car magazines*. The results that the system offered as the best it could find came from under the headings of *Scrap Metal Merchants*. That resulted from expanding both the query terms *car* and *magazine* using synonyms, hypernyms and cross references in the directory because nothing matched the initial query:

- The word *product* is a hypernym of *magazine*.
- The heading *Reclaimers - Waste Products* cross-references to *Scrap Metal Merchants*.
- The heading *Car & Commercial Vehicle Dismantlers* cross-references to *Scrap Metal Merchants*.

As a result, we had to restrict the query expansion procedure. Addresses that are now retrieved are filed under *Newsagents*, though these do not match the complete query.

A different example is the request for *kitchen cupboard specialists in Ipswich*. In an old version of the YPA the result was not very user friendly. What happened was that there was no match for the complete query but trying partial matches resulted in *too many* adverts, because the only partial match that was considered was looking for *specialists* which of course gives *too many* addresses. And that was exactly the information given to the user with the request to relax the query. In the latest version of the system the user now still does not get any addresses with this query but rather than just asking the user to relax the request there is now a dialogue step that offers relaxations in several ways. In this example you would get:

1. *Do you want to see matches in a wider area?*
2. *Should I search for kitchen specialists?*
3. *Should I search for cupboard specialists?*

For each of these options a corresponding checkbox is displayed. However, the default input field allows the user to continue the dialogue in a different way in case none of the displayed options make the user very happy (in this case the user might input *Restart please!*).

Finally, a last example: In a version with a brutal but simple stemmer we were able to match a query for *press photography* to a heading *Photographers – Press* (which would not work with the Porter stemmer, another simple stemming algorithm [13]), however at the same time a user querying for *hospitality*

retrieved addresses that contained a list of *hospitals*, not the best match. We are still experimenting with two different stemmers, but are convinced that too little stemming is better than too much stemming.

5 Recent Experiences and Evaluation

5.1 Other Input Data

In a recent development we created a similar system using much richer data which however still fits into the same *Backend* structure only that new relations are added. Look at this extract which represents only a part of the information stored for one advert:

```
PRODUCTS AND SERVICES
* MPA Regional Wedding Photographer of the year, 1997
* Commercial, PR and legal photography at competitive prices
* Wedding and portraits a speciality
...

HOURS OF BUSINESS
8.00-21.00 7 days a week including Bank Holidays by arrangement
24 hour answerphone

FACILITIES
Small studio with studio lighting, photographic framing
...
```

It proved to be very simple to convert the new data files once the set of scripts had been developed. Minimal customisation of the scripts were needed. Essentially we extended the extraction process by additional steps.

The main advantage of accessing the new data is that the *free text* in the advert is much longer. In this version there is more information than printed in the directory. Addresses are still very short if we see them as documents in an *Information Retrieval* task. But for the *free text* we discovered:

- The information they contain is still very dense, i.e. every phrase is meaningful.
- The information is often grouped under different labels which are just words or phrases but highlighted in certain ways (e.g. capital writing followed by a line break or underlined). The text can then be split into subunits containing information of type *Products*, *Opening Hours*, *Brands* etc. Now different sorts of *free text* indices exist. This task is based on pattern matching and performs well for the data we have so far.
- The different types of information extracted in the former step can be treated differently, for example a word like *Monday* cannot be ignored when creating indices for the type *Opening Hours*, but it can in all other cases.

– The distribution of keywords for *all* adverts under a given heading shows that the most frequent keywords describe that heading generally very well, i.e. these keywords can be associated with that group of adverts even if they do not occur explicitely in the name of the heading (e.g. the keyword *ticket* is very frequent under the heading *Travel Agents*).

We quickly built a system based on this new data and the first results are very promising though they must be validated by more data.

5.2 Evaluation

For the evaluation we used the YPA version 0.7 for the Colchester area with about 26.000 addresses.

We performed two sorts of evaluation, one technology focussed one precision-based. While the technology based evaluation is of particular interest to find problems in the different components of the YPA, the precision based evaluation is appropriate to see how the system performs overall comparing the results with the user query. Note however, that this is not a trivial task. We cannot just adapt a standard *IR* method of measuring *precision* at certain *recall* points because the YPA is a dialogue system and will not display addresses as long as it cannot find an appropriate set of addresses. We therefore applied simple metrics for the evaluation described here.

We used a query corpus of real user queries, in this case our *Bristol corpus* of 75 user queries. We measured how many of the resulting addresses are appropriate. This was done as a *black box* evaluation where the whole system was hidden. For the results presented here we only queried the database once and did not continue the dialogue if there was one initiated by the system. We then evaluated the system under different assumptions simplifying the evaluation:

1. If the initial user query is successful then calculate the average precision of the addresses by counting appropriate results as *100%* and everything else as *0%*. The average precision should be *0%* if no addresses are retrieved after the initial query. This gives us a baseline.
2. The same, but define the average precision as *100%* for those cases where the query does not give a result. This might seem unrealistic, but in most of these cases the request cannot be satisfied at all because there are no matching addresses as for example in the query *Skydiving in Colchester*.
3. Ignore those queries in the corpus which cannot be matched to any addresses in the database and then calculate the average precision as explained before.

The average precision that we calculated for each of the three assumptions is 61% for assumption 1, 79% for assumption 2 and finally 74% for assumption 3. These are the results of a very simplistic evaluation. One aspect for example is not reflected at all, that is the order and ranking values of the adverts in a successful user query.

We continue with evaluating the YPA. At the moment we are most concerned about a more detailed evaluation and measuring the *recall* as well.

6 Outline

We continue work on the YPA system. One of the interesting parts is to extract more structural information from the new (richer) data files. At the same time this will influence the other components in particular the *Query Construction Component* and the *Dialogue Manager*, because a more varied *Backend* database allows a refined query construction and a more user friendly dialogue.

We also noticed that the function of the *Dialogue Manager* changed quite a lot in the process of the overall development of the YPA. More and more tasks which were initially located in the *Dialogue Manager* have been implemented as customisations of the *Query Construction Component*. It could well be that in future we will merge those two components to just one.

We also plan to make the developed data extraction tools more generic so that they can be applied to completely different domains.

As a major task for the future remains a deep evaluation of the system which involves another user trial once we have developed a framework for this.

7 Acknowledgement

We want to thank all test users at BT Laboratories, Martlesham Heath, for bravely querying the system at various stages without giving up.

References

1. ABITEBOUL, S. Querying Semi-Structured Data (invited talk). In *Proceedings of the 6^{th} International Conference on Database Theory (ICDT)* (Delphi, Greece, 1997), pp. 1–18.
2. AMBROZIAK, J., AND WOODS, W. A. Natural Language Technology in Precision Content Retrieval. In *Proceedings of the 2^{nd} Conference on Natural Language Processing and Industrial Applications (NLP-IA)* (Moncton, Canada, 1998), pp. 117–124.
3. AUST, H., OERDER, M., SEIDE, F., AND STEINBISS, V. The Philips automatic train timetable information system. *Speech Communication 17* (1995), 249–262.
4. DE ROECK, A., KRUSCHWITZ, U., NEAL, P., SCOTT, P., STEEL, S., TURNER, R., AND WEBB, N. YPA - an intelligent directory enquiry assistant. *BT Technology Journal 16*, 3 (1998), 145–155.
5. FLANK, S. A layered approach to NLP-based Information Retrieval. In *Proceedings of the 36^{th} ACL and the 17^{th} COLING Conferences* (Montreal, 1998), pp. 397–403.
6. GLASS, J., FLAMMIA, G., GOODINE, D., PHILLIPS, M., POLIFRONI, J., SAKAI, S., SENEFF, S., AND ZUE, V. Multilingual Spoken-Language Understanding in the MIT VOYAGER System. *Speech Communication 17* (1995), 1–18.
7. HEISTERKAMP, P., MCGLASHAN, S., AND YOUD, N. Dialogue Semantics for an Oral Dialogue System. In *Proceedings of the International Conference of Spoken Language Processing* (Banff, Canada, 1992).

8. Levy, A. Y., Rajaraman, A., and Ordille, J. J. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proceedings of the 22nd VLDB Conference* (Mumbai (Bombay), India, 1996).
9. Lewis, D. D., and Sparck Jones, K. Natural language processing for information retrieval. *Communications of the ACM 39*, 1 (1996), 92–101.
10. McGlashan, S., Fraser, N., Gilbert, N., Bilange, E., Heisterkamp, P., and Youd, N. Dialogue Management for Telephone Information Systems. In *Proceedings of the International Conference on Applied Language Processing* (Trento, Italy, 1992).
11. McHugh, J., Abiteboul, S., Goldman, R., Quass, D., and Widom, J. Lore: A Database Management System for Semistructured Data. *SIGMOD Record 26(3)* (1997), 50–66.
12. Miller, G. Wordnet: An on-line lexical database. *International Journal of Lexicography 3*, 4 (1990). (Special Issue).
13. Porter, M. F. An Algorithm for Suffix Stripping. *Program 14*, 3 (1980), 130–137.
14. Sikorski, T., and Allen, J. F. A task-based evaluation of the TRAINS-95 dialogue system. In *Proceedings of the Workshop on Dialog Processing in Spoken Language Systems, ECAI-96* (Budapest, 1996).
15. Smeaton, A. F. Using NLP or NLP Resources for Information Retrieval Tasks. In *Natural Language Information Retrieval*, T. Strzalkowski, Ed. Kluwer Academic Publishers, 1997.
16. Strzalkowski, T., Guthrie, L., Karlgren, J., Leistensnider, J., Lin, F., Perez-Carballo, J., Straszheim, T., Wang, J., and Wilding, J. Natural Language Information Retrieval: TREC-5 Report. In *Proceedings of the Fifth Text Retrieval Conference (TREC-5)* (NIST Special Publication 500-238, 1997).
17. Voorhees, E. M., and Harman, D., Eds. *Proceedings of the Sixth Text Retrieval Conference (TREC-6)* (1998), NIST special publication 500-240. TREC web site: http://trec.nist.gov.
18. Wahlster, W. Verbmobil: Translation of Face-to-Face Dialogues. In *Proceedings of the 3rd European Conference on Speech Communication and Technology* (Berlin, Germany, 1993), pp. 29–38.
19. Webb, N., De Roeck, A., Kruschwitz, U., Scott, P., Steel, S., and Turner, R. Natural Language Engineering: Slot-Filling in the YPA. In *Proceedings of the Workshop on Natural Language Interfaces, Dialogue and Partner Modelling (at the Fachtagung für Künstliche Intelligenz KI'99)* (Bonn, Germany, 1999). http://www.ikp.uni-bonn.de/NDS99/Finals/3_1.ps.
20. Woods, W. A. Conceptual Indexing: A Better Way to Organize Knowledge. Technical Report SMLI TR-97-61, Sun Microsystems Laboratories, Mountain View, CA, 1997.
21. Zhai, C. Fast Statistical Parsing of Noun Phrases for Document Indexing. In *Proceedings of the 5th Conference on Applied Natural Language Processing* (Washington DC, 1997).
22. Zue, V. Toward Systems that Understand Spoken Language. *IEEE Expert Magazine February* (1994), 51–59.
23. Zue, V. Navigating the Information Superhighway Using Spoken Language Interfaces. *IEEE Expert Magazine October* (1995), 39–43.
24. Zue, V., Glass, J., Goodine, D., Leung, H., Phillips, M., Polifroni, J., and Seneff, S. The VOYAGER Speech Understanding System: Preliminary Development and Evaluation. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing* (1990).