Natural Language Access to Yellow Pages

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

Department of Computer Science
University of Essex
Colchester, Essex, CO4 3SQ, UK
{udo,webbnw}@essex.ac.uk

Keywords: Natural Language Processing, Semi-Structured Data, Dialogue Management

Abstract

The YPA is a directory enquiry system[1] that allows a user to access addresses and other advertiser information in a classified directory. BT's *Yellow Pages*[2] provide an example of a classified directory with which this work would be useful.

1. Introduction

The YPA allows access to a database containing advertiser information that has been extracted from some raw input source. This input data contains address information as well as some *free-text*, often describing goods or services offered by an advertiser. In general, we are interested in *semi-structured* input data, in other words "data that is neither raw data, nor very strictly typed as in conventional database systems" [1]. Access to this data is provided through a natural language dialogue system, which takes queries as input, returns addresses, consults knowledge sources or engages the user in a dialogue if necessary. This paper gives an overview of the system and is structured as follows. First we will give a general description of the architecture of the YPA system (section 2). In section 3 we will point to related work. Section 4 explains the modules of the online system, although for further detail, we refer the reader to [2]. Finally we describe some evaluation results (section 5) and conclude with an outline of future work (section 6).

[1] This work has been funded by a contract from the ISR group at BT Laboratories, Martlesham Heath, UK.

[2] *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 be a fully structured query, such as:

> *'I want a plumber for my boiler who takes visa in Ipswich'*

A conversation cycle with the YPA can be roughly described as follows. A user utterance (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 (QCC)*, passing it the result of the parsing process. 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 that 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 consider the YPA as a natural language dialogue system, we find it comparable to numerous previous systems. In the main these sophisticated systems, such as OVIS [3], Sundial [4] and TRAINS [5], are concerned with simple, heavily restricted domains in terms of either size or complexity of data. In contrast, a number of more basic online directory enquiry systems are actually in use. Some of the existing systems that offer information one would expect from the *Yellow Pages* are: *Electronic Yellow Pages*[3] (U.K.); *Scoot* (U.K.)[4]; *NYNEX Yellow Pages* (U.S.A)[5]; *BigBook* (U.S.A)[6] and *Switchboard* (U.S.A)[7]. This type of commercial system functions quite differently from traditional *Natural Language Processing (NLP)* or *Information Retrieval (IR)* systems. While they access large address databases of the same sort that we deal with they all share a number of limitations: a very flat frontend which at most offers pattern matching; a very simple lookup database which does not permit the access of free text hidden in the addresses and the inability to cope with words not found in the database (i.e. in the categories or names).

4. The Online System

4.1 The Natural Language Frontend

The function of the Natural Language Frontend is to parse the input from the user then take this parse tree and submit it to a slot-filling procedure. Parsing methods for handling real NLP applications are being developed all the time. Notably, there are methods such as the LR parsing algorithm [6], phrasal parsing as seen in the SPARKLE project [7] and chunking [8].

It is a primary aim of these systems that they return some linguistically correct parse tree for the input, and have the ability to choose correctly between syntactically (and semantically) ambiguous analyses. However, we need to make no claim about the resolution of ambiguity on the basis of syntactic structure, but rather let the domain database resolve such ambiguities as it sees fit. What we wanted to do was rapidly assign a skeleton to any input based on the closed class words. This would provide us with a frame, and any other words in the input would be given a limited freedom to try out different classes until one fitting a parse could be generated. As soon as one parse tree exists, it is accepted.

Bearing these factors in mind a simple DCG-like grammar, without any feature requirements was adopted, easy to both understand and modify. A basic bottom-up chart parser was implemented [9]. The resultant parse tree produced by this parser was passed to the slot-filling process.

A number of Natural Language Dialogue Systems have used a similar representation, notably Voyager [10], ATIS [11] and RailTel [12]. The key difference between the use of the slot-filler mechanism in these systems and our approach is our preservation of some syntactic structure. This allows for a greater degree of flexibility during the query construction phase, and goes beyond simple keyword matching.

For this domain we initially identified three main relevant slots, *Goods*, *Transaction* and *Location*. Once we received richer data sources, we were also able to incorporate three more slots, *Street Name*, *Payment Method* and *Opening Hours*. Once a parse tree has been produced it is passed to the *domain independent structure analysis* and is split, where possible, into <*subject*>, <*verb*>, <*object*> and <*modifiers*>.

When the input is broken in this way, it is passed to the *domain dependent pragmatic analysis.* At this point, it becomes necessary to know about the behaviour and requirements of the domain.

Here the slots are identified and parts of the input mapped into them. Some mappings are straightforward. For example, in this domain, it is usually not necessary to know the subject information so this is discarded. The verb information maps directly into the *Transaction* slot, and generally speaking the object maps into the *Goods* slot.

4.2 Dialogue Manager

The *Dialogue Manager* plays an important role by controlling the access to the various modules. The user input is passed to the *Dialogue Manager*, which calls the *Natural Language Frontend*. Depending on the result, it decides whether for instance the database should be accessed (by calling the *Query Construction Component*), whether the dialogue should be restarted or if clarification is required from the user of some unknown query terms. This has been described in [2].

[3] http://www.eyp.co.uk
[4] http://www.scoot.co.uk
[5] http://www.bigyellow.com
[6] http://www.bigbook.com
[7] http://www.switchboard.com

4.3 The World Model

The function of the *World Model* is to allow query expansion and query modification. This is done either to retrieve addresses or to allow the user to choose various ways of modifying the original input if the query cannot be fired successfully. The *World Model* is not a homogeneous component. It currently has these main parts: a large lexicon containing various simple hierarchies (*WordNet*); the heading structure from the *Yellow Pages*, and knowledge acquired by the user (updated via the YPA *AdminTool*).

4.4 The Backend Database

The *Backend* contains three quite different parts.

Firstly, the (relational) database that contains tables extracted from the *Yellow Pages printing tape*, using information found in the *free-text* of an advertisement in addition to other relations (mapping locations to dialling codes for example). Secondly a database containing *meta data* for these indices (such as ranking values for the appropriate indices, relevant to the *Query Construction Component*). Finally, there is the *Language Module* (e.g. for reduction of words to base forms). The *Backend* is explained in far greater detail in [2], including a description of the processes used to create the database itself from the original source data.

4.5 The Query Construction Component

The structure passed to the *Query Construction Component* is a *slot-and-filler* query. The task is to match this query to a set of addresses by consulting different sources of knowledge. It is therefore the task of the *Query Construction Component* to evaluate the various information sources (e.g. indices *versus* ranking values) and retrieve a set of addresses from the *Backend* if possible. The constructed query is sent to the address database. If this results in a set of addresses (up to a maximum number defined by the administrator in the set-up), then the *Query Construction* is finished. Otherwise there is a general strategy of successive relaxation of the query. A query that resulted in no matching addresses would for example be relaxed by ignoring slots which are not as important as others (e.g. *Opening Hours* as opposed to *Goods*) or by exploiting syntactic information (prepositional phrases could be ignored). This all depends on how the QCC is set up.

If too many addresses are retrieved, then the query will be further constrained if possible. In case that after query modification there is still no set of addresses then an appropriate flag is passed back to the *Dialogue Manager*.

5. 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 and 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. For the technology evaluation we used the method of coverage, where we calculate the percentage of sentences assigned an analysis from an unannotated corpus.

90% were successful from the parser point of view, although of those 22% failed at some later point in the system (incorrect query construction or spelling errors - note that these did not cause a parse failure, but failed to match information in the database). 10% were recorded a *Frontend* failure - where either queries failed the parse completely, or information was misplaced, and put in incorrect slots.

For the precision-based evaluation we used a query 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.

We developed a response sheet for our queries, where we recorded if addresses were returned, how many dialogue steps were necessary, the total number of addresses recalled and the number of those deemed relevant to the original query. 62 of our 75 queries caused the system to return addresses, and 74% of those addresses were relevant to the original query. This leaves the question of how to treat the remaining 13 queries, which did not produce answers. For example, if the user asks:

'I want the number of Barclaycard gold'

and the system replies:

```
'There is nothing in the data-
base about Barclaycard gold'
```

then how should this be scored?

In order to have some baseline score, we can assign all of these 13 queries to a score of 0%. This gives us a worst case scenario score of 61%. Instead, we could have measured these 13 queries as 100%. Although this at first seems unrealistic, in the case of the example above it

would be a valid result. In fact, we believe that the nature of reporting failed queries to the user lies central to the issue of user satisfaction with the system.
These are the result 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. Future Work

We continue work on the YPA system. One of the interesting parts is to extract more structural information from 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 that 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 would merge those two components to just one. We 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.

References

[1] Abiteboul, S. "Querying Semi-Structured Data", invited talk, Proceedings of the 6th International Conference on Database Theory (ICDT), 1997, pp. 1-18.

[2] 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, 1998, 16(3), 145–155.

[3] Nederhof, M-J., Bouma, G., Koeling, R. and van Noord, G. "Grammatical analysis in the OVIS spoken-dialogue system", Proceedings of the ACL/EACL Workshop on Spoken Dialog Systems, Madrid, 1997.

[4] Peckham, J. "A new generation of spoken dialogue systems: results and lessons from the SUNDIAL project", Proceedings of the 3rd European Conference on Speech Communication and Technology, Berlin, 1993.

[5] Allen, J., Schubert, L., Ferguson, G., Heeman, P., Hwang, C., Kato, T., Light, M., Martin, N., Miller, B., Poesio, M. and Traum D. "The TRAINS Project: a case study in building a conversational planning agent", Journal of Experimental and Theoretical Artificial Intelligence, 1995, 7:7-48.

[6] Tomita, M. "Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems", Kluwer Academic, 1985.

[7] Briscoe, T., Carroll, J., Carroll, G., Federici, S., Grefenstette, G., Montemagni, S., Pirrelli, V., Prodanof, I., Rooth, M. and Vannocchi, M. "Phrasal Parser Software – Deliverable 3.1", 1997, http://www.ilc.pi.cnr.it/sparkle.html.

[8] Abney, S. "Parsing by chunks" in Abney, S., Berwick R. and Tenny, C. (eds.) Principle-Based Parsing, Kluwer Academic Publishers, 1991.

[9] Gazdar, G. and Mellish, C. "Natural Language Processing in PROLOG: An Introduction to Computational Linguistics", Addison Wesley, 1989.

[10] Zue, V. "Towards Systems that Understand Spoken Language", IEEE Expert Magazine, February 1994, pp 51-59.

[11] Bannacef, S. K., Bonneau-Maynard, H., Gauvain J. L., Lamel, L. and Minker W. "A Spoken Language System for Information Retrieval", Proceedings of the International Conference on Speech and Language Processing, Yokohama, Japan, 1994.

[12] Bannacef, S. K., Devillers, L. Rosset S. and Lamel, L. "Dialog in the RAILTEL Telephone-Based System", Proceedings of the International Conference on Speech and Language Processing, Philadelphia, 1996, pp. 550-553.