# My Instant Expert

*George Ferizis*
CSIRO ICT Centre
Locked Bag 17
North Ryde NSW 2113 AUSTRALIA
*george.ferizis@csiro.au*

*Peter Bailey*
CSIRO ICT Centre
GPO Box 664
Canberra ACT 2601 AUSTRALIA
*peter.bailey@csiro.au*

**Abstract** *This paper gives an overview of a mobile device question answering application that has recently been developed in the CSIRO ICT Centre. The application makes use of data in an open-domain encyclopaedia to answer general knowledge questions. The paper presents the techniques used, results and error analysis on the project.*

**Keywords** Information retrieval, Question answering

## 1 Introduction

*My Instant Expert*™is a mobile device question answering system. The aim of the application is to create a general knowledge question answering system that can be accessed from a mobile device. It uses Wikipedia [2] data to answer questions.

Many question answering systems already exist [6, 8]. *My Instant Expert*™is different to most other question answering systems as it answers open-domain questions from a closed corpus [1], it has tight time constraints on pre-processing and query processing, and it delivers to mobile devices.

Answering open-domain questions from a closed corpus presents many challenges, as it is difficult to construct open-domain ontologies or taxonomies that would aid question answering. The small amount of text redundancy in the corpus makes it challenging to retrieve the correct text segments while searching for answers. Applying computationally expensive language processing techniques, such as coreference resolution, which could potentiall improve the accuracy of the system, cannot be applied due to the tight time constraints on pre-processing and query processing stages.

This paper begins by discussing the development methodology behind the project before giving an overview of the components of the system. The results of system testing using questions from previous TREC question answering tracks [16] is then presented, along with discussion on these results and some error analysis.

The paper continues with a discussion of some of the challenges related to developing mobile device applications, and presents an overview of the mobile device application that we developed. Particular focus is given to how the application differs from an equivalent application deployed via a web browser. A description of a user evaluation which was conducted using the mobile phone application is then given, and the results of the evaluation are discussed. The evaluation was conducted to understand user preferences for question answering on mobile devices. The final section discusses some future research directions that result from the error analysis and the user evaluations.

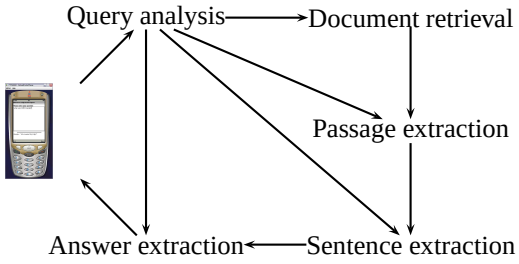



Figure 1: System diagram

## 2 Description of work

*My Instant Expert*™does what a number of question answering systems do. It sequentially applies more computationally expensive search techniques to reduce the search space from the entire corpus to a small number of candidate answers. Figure 1 contains a diagram of the system. In order of application these techniques are: document retrieval, sub-document passage retrieval, sentence extraction, and answer extraction. An input to each of these stages is the output of the previous stage and the results of some query analysis. Communication is done through the transfer of XML files between modules. This allows modules to be substituted with others easily.

The original system design envisioned an application that would run on a mobile platform. Unfortunately as the size of the Wikipedia corpus is currently doubling annually and contains 6GB of data as of September

[1] The MIT START system is open domain, but it is restricted to a set of queries; see *When did John Howard become prime minister?*

**Proceedings of the 10th Australasian Document Computing Symposium, Sydney, Australia, December 12, 2005.** 

2006, device constraints make this task very difficult. As a result, we developed a client server architecture where the mobile device application queries a server which then processes the natural language questions.

## 2.1 Query analysis

The query analysis module identifies what "topics" or phrases the query is asks about. This is done by chunking the text using the OpenNLP toolkit before removing chunks that contained words to do with question structure (e.g., "*what is*" and "*when did*").

The type of answer required (*expected answer class*) is also determined. The set of answer classes is restricted to: *location*, *numeric*, *person*, *date* and *description*. Everything else is placed into an *unidentified* class. The questions are classified using a Bayesian classifier over features such as named entities in the query, term frequency of words from specific dictionaries, part of speech bigrams and chunking bigrams.

Query expansion is also done during this stage. During query expansion, terms that are semantically related to terms in the query are discovered using WordNet [3]. The kinds of relationships found are synonym, hyponym, derivational and attribution relationships.

## 2.2 Document retrieval

Typically question answering systems use a document retrieval engine to retrieve documents from the corpus that may contain a possible answer. Experimentation of various ranking techniques using the wikipedia corpus found that retrieval on title metadata was superior to full-text retrieval for question answering. This lead us to test a naive method of document retrieval that ranks according to the presence of the document title in the question, with the score being the percentage of terms from the query occurring in the title. An example of this is shown in Figure 2 for the query "*When did John Doe eat cake?*" [2].

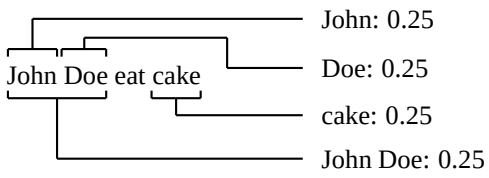



Figure 2: Ranking of documents against the query "*When did John Doe eat cake?*"

A comparison of this technique with several other techniques is presented in Table 1. Google queries were of the form "`site:en.wikipedia.org (intitle:<CHUNK>)+`", when querying on title metadata and "`site:en.wikipedia.org (<CHUNK>)+`", when querying on full text. The queries to the other search engine were of similar form. The table shows that the method of "*title matching*" outperforms both full text search, and probabilistic search over title metadata using two different search engines.

The reasons for this performance difference can be attributed to the task and the corpus. Wikipedia document titles are always unique and are more often than not named entities, or useful noun phrases that describe the contents of the document. Wikipedia documents also tend to have multiple titles, with over a third containing more than one title and over 3% containing over 5 titles.

## 2.3 Sub-document passage retrieval

Retrieved documents are broken into sub-document elements such as: image descriptions, lists, paragraphs, tables and templates (a form of MediaWiki marked up data). The elements are combined with contextual information about the position of the element in the document such as the section headings and document title.

Including the title and section headings allows us to approximate using a pronoun resolution tool. Wikipedia documents, like most documents, typically do not reference nouns repeatedly but instead use pronouns. Intuitively the noun that an encyclopaedia document will most frequently refer to using pronouns is the entity the document corresponds to. As such including the title and section headers when ranking a candidate sentence or passage (although weighted differently to terms occurring in the actual sentence) allows the contextual information of surrounding tags in the document to be included in the ranking of the sub-document elements.

The probabilistic Okapi BM25 [13] is used to rank passages from the retrieved document set. Stemming using the Porter stemming [12] algorithm was found to help the performance of the system.

## 2.4 Sentence extraction

The sentence extraction algorithm ranks sentences by comparing terms in the query and the candidate sentences. If the terms from the original question cannot be found in candidate sentences, the system searches for terms obtained from query expansion. Matches for terms obtained from query expansion are weighted lower during ranking than terms from the original sentence.

Tabular elements are parsed to disambiguate the contents of entities in the table. This is done using column and row headers if they are present in the table.

## 2.5 Answer extraction

If the expected answer class can be identified, then the answer extraction module attempts to find the matching named entity in the candidate sentences. This both reranks the sentences if the named entity is detected and extracts the desired named entity from the sentence. The module does this through named entity recognition,

[2] Due to space constraints a real query and real Wikipedia documents are not used in the example.

Table 1: Comparison of title matching with various other methods

| Method | P@1 | P@5 | MRR |
|---|---|---|---|
| Title matching | 0.77 | 0.85 | 0.81 |
| Full-text search | 0.44 | 0.62 | 0.53 |
| Full-text search - Nouns only | 0.42 | 0.62 | 0.51 |
| Full-text search using title metadata | 0.31 | 0.48 | 0.38 |
| Full-text search using title metadata - Nouns only | 0.33 | 0.58 | 0.43 |
| Google | 0.25 | 0.35 | 0.29 |
| Google - Nouns only | 0.48 | 0.56 | 0.52 |
| Google using title metadata | 0.42 | 0.56 | 0.49 |
| Google using title metadata - Nouns only | 0.42 | 0.58 | 0.50 |

using a tool base on the GATE toolkit [7] and some shallow sentence parsing to identify similar syntactic relationships between terms in the query and terms in the sentence.

The results returned contain a short answer, which corresponds to the named entity regarded to be the answer, and a longer answer to justify the shorter answer (typically the sentence in which the shorter answer was found). These two separate answers are used for display purposes on smaller mobile device screens. If the named entity is not found in the sentence, or if the named entity required by the question is unidentified, the answer only contains the short answer which corresponds to the candidate sentence. Figure 3 shows sample output in response to two questions.

```
<original>
    When was John Howard born?
</original>
<short-answer>26 July 1939</short-answer>
<detailed-answer>
    John Winston Howard (born 26 July
    1939) is an Australian politician and
    is currently the Prime Minister of
    Australia.
</detailed-answer>

<original>What color is grass?</original>
<short-answer>
    Grass generally describes a
    monocotyledonous green plant in the
    family Poaceae, botanically regarded
    as true grasses.
</short-answer>
<detailed-answer />
```

Figure 3: Sample output

## 3 Special circumstances

There are two circumstances where *My Instant Expert*™ does not follow the steps outlined above to answer a question. The first involves answering definitional questions such as "*Who is X?*" and "*What is Y?*". In the absence of any contextual information that could be used to infer points of interest about *X* or *Y*, the answer returned is a summary of the Wikipedia document describing the entity. As Wikipedia documents typically start with a brief summary of the document, the returned answer is usually the first paragraph of the document.

The second involves disambiguation of entities. Again, in the absence of any contextual information, it is difficult to answer question such as "*Who is George Bush?*", or "*What is a plane?*", as the names used in either question can refer to multiple entities. To solve this, a list is presented to the user containing entities that match their query. The user then obtains an answer by selecting the entity they are interested in.

## 4 Results

Automated result quality testing using the TREC QA track test sets [16] was conducted during development. As the TREC QA test set contains many questions that are temporally constrained (e.g., *What is the population of Mississippi?*), a snapshot of the Wikipedia corpus was taken, and all answers were verified using the corpus.

During testing, over 800 questions were used. We took measurements of precision at 1 (*P*@1) - how often was the first answer a right answer, and the mean reciprocal rank (MRR) - on average what was the rank of the highest ranked correct answer. A response from the system was considered correct if it contained one of the answers that was manually identified as correct. As no attempt was made to construct sentences for answer responses, an answer returned by the system was considered correct if it contained all of the terms in the correct answer. The overall system performance found that the correct answer was ranked first 36% of the time, and that the MRR was 0.50.

The graph in Figure 4 displays the *P*@1 and *MRR* results for the document retrieval, passage extraction, sentence extraction and answer extraction phases of the system. As is expected, the accuracy of the system decreases as the amount of data being returned by the system decreases, with retrieval of large quantities of

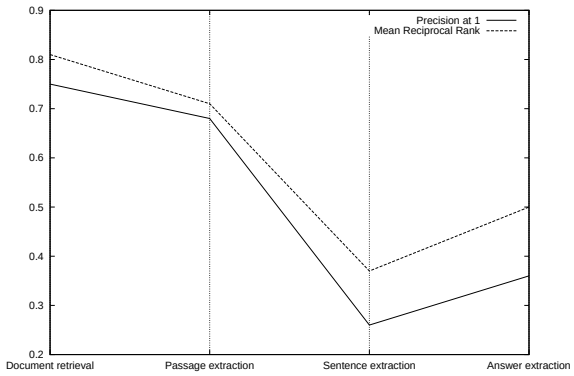



Figure 4: Results of testing by phase

text such as documents and passages having a high accuracy. The reranking of answers based on the presence of the requisite answer class in the final stage improves the accuracy of the system significantly.

The accuracy gains of the answer extraction phase supports the assertion that named entity recognition impacts on the performance on the system. Table 2 shows that, with the exception of the *mountain* entity (which can be attributed to a statistical anomaly due to infrequent occurence), the answer tended to be more accurate when the entity required by the question could be easily discovered by the named entity recogniser.

Table 2: Results of testing, grouped by answer class

| **Answer class** | **P@1** | **MRR** |
|---|---|---|
| Group | 0.67 | 0.70 |
| Mountain | 0.50 | 0.55 |
| Description | 0.49 | 0.67 |
| Date | 0.47 | 0.63 |
| Individual | 0.44 | 0.55 |
| City | 0.38 | 0.49 |
| Country | 0.33 | 0.39 |
| Unidentified | 0.32 | 0.45 |
| Numeric | 0.12 | 0.29 |
| Title | 0.07 | 0.17 |
| State | 0.00 | 0.23 |
| *Overall* | 0.36 | 0.50 |

Our results compare favourably with recent results from the TREC QA track, ranking in the top 10 performers on *initial factoid* questions, although it is unlikely that our assessment was as strict as that used in the TREC track. There were differences to how our system and the systems for the TREC track were evaluated:

- TREC participants did not need to implement document retrieval - they were given a list of documents from the corpus that could possibly answer the question.
- The TREC corpus is fixed, however precedent to make use of external corpora such as Wikipedia has been established [4].
- It is not clear how precise an answer must be before it is considered correct for the TREC track, e.g., is *Mississippi* a suitable substitute for the answer *Mississippi river*?

## 5 Discussion

The results showed that the system may be improved if the accuracy of the named entity recogniser was higher. Further error analysis showed how vital the accuracy of the question classifier is.

Table 3: Results of testing, grouped by correct identification of answer entity

| **Identification** | Number | **P@1** | **MRR** |
|---|---|---|---|
| Correct | 307 | 0.45 | 0.59 |
| Incorrect | 504 | 0.31 | 0.46 |
| *Overall* | 811 | 0.36 | 0.50 |

The results in table 3 demonstrate that when the required answer entity was correctly identified the performance of the system increased dramatically. The results also highlight how inaccurate our question classifier was.

Improvements to the answer classifier must address two abstract questions: "*How many types of entities should be detected?*" and "*How fine grained should these entities be?*".

### 5.1 Question classification vs. named entity recognition

Named entity recognisers and classifiers for closed domain applications, such as the medical or military domains, focus on classifying entities into a large set of very specific manually developed categories which are derived from a small and closed set of broad categories. As *My Instant Expert*$^{TM}$is an open domain application, questions could be about anything. Thus it is not feasible to develop manually an exhaustive set of non-overlapping entities that make sense. Even if this set was developed, the task of creating a classifier would be more difficult as having more categories and fewer distinctions between the categories makes classification much less accurate.

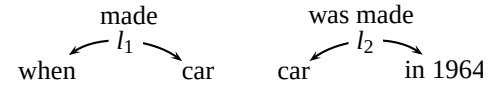
Research into classification of questions [9] from the TREC QA track [16] shows that results of up to 84% can be obtained for the finest classification, over a variety of entities ranging from vehicles to currency. Inspection of their methods is possible as they have generously provided the training data and pre-processing information on-line [1]. It shows that the feature selection is very specific to their categories, e.g., they have specific rules for colours. This would suggest that if more category specific features were included more accuracy gains could be made. However, with a large set of categories, how would these features be obtained?

Being able to classify the question is only halfway to solving the problem. Even if a question classifier

could always determine what entity a question is asking for, the named entity recogniser and classifier still needs to find the entity in the candidate answers. The task of identifying the entities becomes even more difficult. When there are a larger number of entities. The errors of the two stages compound each other, when using current state of the art techniques and a small set of entities, the error rate is such that 1 in 5 questions will have an error as a result of either of these two processes. Without improvements to either the question classification or named entity recognition this error rate will remain quite high.

## 5.2 Semantic relationships

Techniques that attempt to discover syntactic and semantic relationships between words [14] may reduce the dependency of the system on the problem of entity classification. Consider the example in Figure 5. If the relationships in Figure 5(a) and Figure 5(b) could be discovered and if similarity between the labels for the relationships $l_1$ and $l_2$ could be inferred, there would be no need for the system to recognise that the question is asking for a date.




(a) *When was the car made?* (b) *The car was made in 1964.*

Figure 5: Example relationships

The example given is a basic one, and the graphs in the relationships unrealistic. However it is illustrative. It is trivial to infer that the term `made` is identical to the term `made`. However, how easy is it to infer that the term `manufactured` may have similar meaning? How much harder still is it to infer that the phrase `rolled off of the assembly line` may also be equivalent in some circumstances? The problem of discovering semantic relationships such as these contributes significantly to the error rate of the system. While it is hard to put a figure on this contribution, manual experimentation suggests that it could be quite high.

It is hard to quantify the effects of missing semantic relationships. Our results show that the right answer was ranked at 1, 68% of the time during the ranking of paragraphs, while this result drops to 26% when the unit of text is sentences. A substantial part of this decrease in performance is likely due to the text redundancy present in the larger passages, hiding the inability of the system to discover semantic relationships between terms in the query and terms in the passage, such as anaphora. The importance of resolving anaphora has already been measured in previous work [15].

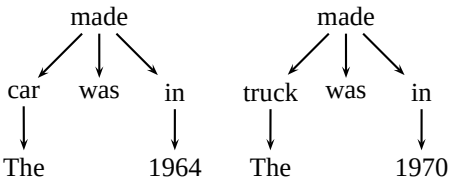
Previous measurement into the negative contribution of undetected synonymy on question answering performance [6] suggests that it contributes to only 2% of the incorrectly answered questions. However, these measurements used the web as the corpus. Using the web as a corpus makes the case for detecting synonymy far less compelling as, due to the size of the web, it is likely that the same relationship has been expressed using many different phrases. Manual experimentation suggests that the inability of the system to detect other semantic relationships such as synonymic, metaphoric and metonymic[3] relationships between query terms and phrases in potential answers results in a large number of poorly answered questions.

WordNet is used in *My Instant Expert*™ to detect some of these relationships, and it works very well for detecting term-to-term synonymy and even well for some term-to-phrase synonyms such as `kicked the bucket` and `die`. However, it is far from complete. If, for example, the question *"When was the first Ford Mustang manufactured?"* was submitted, we are incorrectly informed that it was manufactured in 1972. Examining the Wikipedia corpus, we could find an answer to this question only in the phrase *"The first production Mustang, .., rolled off the assembly line in Dearborn, Michigan on March 9, 1964"*. If the metonymic relationship between `manufactured` and `rolled off the assembly line` was detected, this would have been a highly ranked answer. But unfortunately, it was not.

In 2001, Lin and Pantel [10] described a method that measures syntactic relationships between two phrases to determine whether two phrases are similar. This method could be used to populate databases like WordNet automatically. The method works by identifying relationships between parse trees such as those shown in Figure 6. Using a starting phrase of "`The car was made in 1964`", the algorithm will determine that the paths between {`car, 1964`} and {`truck, 1970`} are similar as they follow a similar route through their parse trees (`made in`).

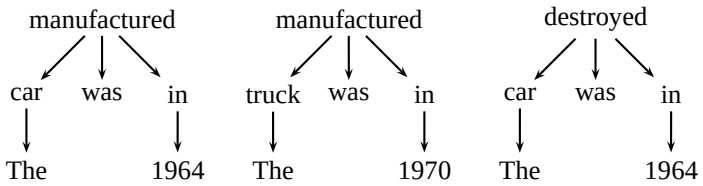
The next step of the algorithm involves it trying to discover all other statistically significant paths that connect the terms {`car, 1964`}, and {`truck, 1970`}. Figure 7 shows some phrases that will likely be matched by this phase of the algorithm. The three trees in Figures 7(a), 7(b) and 7(c) all would be returned. While on first appearance this is incorrect, it is in fact correct as `destroyed` is in some way related to the term `made`. Without a significantly higher rate of occurrence of the other two trees, it is very likely that `destroyed` will be related to `manufactured`. While this is correct it is not the desired outcome. The initial work attempts to leverage textual redundancy which does not exist in Wikipedia. Preliminary results of using this technique over the Wikipedia corpus has found that there is not enough text redundancy to filter out antonym-like semantic relationships. The results

[3] metonymy: A figure of speech in which one word or phrase is substituted for another with which it is closely associated, as in the use of `the crown` for the English monarchy, or `Washington` for the United States government.




(a) Parse tree for *The car was made in 1964.* (b) Parse tree for *The truck was made in 1970.*

Figure 6: Similar parse trees




(a) Parse tree for *The car was manufactured in 1964.* (b) Parse tree for *The truck was manufactured in 1970.* (c) Parse tree for *The car was destroyed in 1964.*

Figure 7: Possibly similar parse trees

however has only processed a 10% of Wikipedia due to time and space constraints.

## 6 Mobile application

Developing applications for mobile devices presents many problems that are not encountered when developing browser based applications. Different manufacturers of mobile devices create vastly different APIs for their devices, and devices from the same manufacturer often have significantly different APIs. While this problem can be solved by using portable APIs such as Sun's J2ME, to remain portable the APIs do not take full advantage of the device's I/O capabilities. Even basic capabilities such as horizontal screen scrolling cannot be taken for granted or joystick functionality. As a consequence, developing an application that remains both portable and usable is especially difficult.

Our application was developed using J2ME and is compatible with any mobile devices that are MIDP 2.0 and CLDC 1.0 compliant. We attempted to get around constraints on the use of phone I/O capabilities by making novel use of rudimentary API functions.

Our application allows the user to enter a new question or rephrase an existing question using the default SMS-style input interface of the mobile device as shown in Figure 8(a). This allows the input aspect of the application to remain as portable and as familiar to the user as possible.

While it is possible to present a large list of possible answers that are very verbose in a web browser, due to the space constraints on the screen of a mobile device, this is not possible. This forced us to reconsider two common design aspects of question answering interfaces and search interfaces: how the justification of an answer is presented, and how multiple answers or results are presented to the user.

As shown in Figure 8(b) the answer to the question "*When was CSIRO founded*" is split into two portions, one containing a short answer (1926), and the other containing the text this answer is obtained from. This text is included to justify the answer to the user. Splitting the text like this gives the user the ability to read an answer without having to scroll through multiple screen folds if they do not desire justification.

The screen shots in Figures 8(b) and 8(c) show different answers to the same question being displayed on the mobile device screen. These answers are selected through the use of a tab control on the top of the screen. This differs from the usual result list presented by most search engines. This allows users to quickly access different answers without a need to scroll through many screen folds.

## 7 Usability and system evaluation

A central concern when designing any application is usability. We developed a series of user stories and personas to help identify the kinds of environments in which the application would be used. These provided input into the initial user interface design prototype for the mobile device.


(a) Question input screen. (b) First answer screen. (c) Fifth answer screen.

Figure 8: Screenshots of the mobile device client

Various usability evaluations were carried out through the life of the project. These included early think-aloud evaluations of the prototype mobile device UI, which identified various problems with the initial design. Once the system was operational, we discovered technical problems with answer quality and mobile device UI software development kits that required additional changes. We followed these later with more comprehensive whole-system evaluations with a new set of individuals. A survey questionnaire was developed for the latter to help identify familiarity of the evaluator with mobile device technology in general. Again, a number of issues were picked up, leading to improvements in the design of the interaction processes and layout.

System evaluation was conducted on the users after the usability evaluation. The users were asked a series of questions revolving around their "feelings" about the system. The questions included:

- Did they feel the system met their information need?
- Was the information returned in a timely manner?
- How highly did they value justification of the answer?
- Did they enjoy reading related information?
- Overall satisfaction with the system.

Several results of the evaluation were surprising:

- While users preferred the short answer being displayed on the screen first, 90% of users appreciated having a longer answer present on the mobile device even if it meant having to scroll repeatedly.
- Often enough only 1 out of 5 of the answers was correct, the corollary of this being that the remaining 4 were incorrect. Surprisingly so long as the answers were on topic the users liked it. The reasoning for this was that the system had provided them with more and still relevant information which made them feel satisfied that they had obtained useful information from the system.
- Users did not mind waiting up to 2 minutes for an answer to be returned, even though the communication was not asynchronous.

The first two results suggest that presenting serendipitous information, or giving easy ways to to access such information is judged to be favourable by users. The third results suggests that applying more computationally expensive techniques to answering questions may be feasible without raising the ire of users. Overall the system was liked by users.

## 8 Scope and plans for future research

The *My Instant Expert*$^{TM}$architecture also supports ongoing enhancements to components and functionality extensions, which serve the goal of building a platform for research experiments. Again due to the short development timeframe, some of the research experiments involving contextual delivery of information could not be carried out. For example, do alternative methods of presenting information make a qualitative difference to users and. can prior conversational history be used to answer follow-up questions better? These can be investigated subsequently using the basic system.

The dilemma faced by researchers interested in QA systems is that nearly all existing QA test collections are not reusable. (An exception is the factoid question test collection developed by Lin and Katz [11].) The reason for this is that judged documents (and ensuing relevance measures) from a corpus are rarely sufficient to support stability of the measures when new systems discover additional documents (not previously judged).

As mentioned in [5], we plan to provide logs (suitably anonymised) of real user queries from the system to the broader research community. These can be used to understand what real users ask of an open domain QA system. From a research perspective, access to real user query logs is an invaluable resource, and one that is often available only to the operators of search services. These query logs will provide one component required for the development of additional reusable QA test collections.

The query logs can also be used to work towards a solution for one of the problems identified during error analysis. Inspection of query logs may allow us to develop a taxonomy of entity classes specific to open-domain question classification based on the kinds of queries users frequently make.

As raised during the system evaluation, users valued extra on-topic information. Research into how to determine what would be appropriate "follow-up" questions to present the user with is currently being conducted. The research is looking at what questions would be valuable to users based on the question asked and the answer received.

## 9 Acknowledgements

We would like to acknowledge various members of the CSIRO ICT Centre for their contribution to this project: Alexander Krumpholz, Ronnie Ma, Denis Mikhalkin, Shannon O'Brien, Cécile Paris and Tom Rowlands. We also appreciate the input on other matters relating to the project from Jim Lilley, Ross Wilkinson, David Hawking, Colin Murphy, David Lau, Anthea Roberts, Andrew Lampert, Pascale de Souza Dromund, Tom McGinness, and Gautam Tendulkar. Special thanks to Alex Zelinsky for sponsoring the project in the first place.

## References

[1] Learning Question Classifiers - Experimental Data for Question Classification. *http://l2r.cs.uiuc.edu/ cogcomp/Data/QA/QC/*.

[2] Wikipedia. *http://www.wikipedia.org*.

[3] *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May 1998.

[4] David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Muller, Maarten de Rijke and Stefan Schloback. Using Wikipedia at the TREC QA track. In *TREC*, 2004.

[5] Peter Bailey and George Ferizis. Possible approaches to evaluating adaptive question answering systems for mobile environments. In *First International Workshop on Adaptive Information Retrieval (AIR)*, Glasgow, UK, October 2006.

[6] Eric Brill, Susan Dumais and Michele Banko. An analysis of the askmsr question-answering system. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 257–264, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[7] H. Cunningham, D. Maynard, K. Bontcheva and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[8] Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy J. Lin, Gregory Marton, Alton Jerome McFarland and Baris Temelkuran. Omnibase: Uniform access to heterogeneous data for question answering. In *NLDB '02: Proceedings of the 6th International Conference on Applications of Natural Language to Information Systems-Revised Papers*, pages 230–234, London, UK, 2002. Springer-Verlag.

[9] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[10] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, Volume 7, Number 4, pages 343–360, 2001.

[11] Jimmy Lin and Boris Katz. Building a reusable test collection for question answering. *Journal of the American Society for Information Science and Technology*, Volume 57, Number 7, pages 851–861, 2006.

[12] Martin Porter. An algorithm for suffix stripping. *Program*, Volume 14, pages 130–137, 1980.

[13] S.E. Robertson, S. Walker and M. Beaulieu. Okapi at TREC-7: Automatic ad hoc, filtering, VLC and interactive track.

[14] Dan Shen, Geert-Jan M. Kruij and Dietrich Klakow. Exploring syntactic relation patterns for question answering, 2005.

[15] José L. Vicedo and Antonio Ferrández. Importance of pronominal anaphora resolution in question answering systems. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 555–562, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

[16] Ellen M. Voorhees. Question answering in TREC. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 535–537, New York, NY, USA, 2001. ACM Press.