IQABOT: A Chatbot-Based Interactive Question-Answering System

James Smith
Computer Science
Supervisor: Jonathan Ginzburg

27th April 2010
Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary.

James Smith
Abstract

Interactive question-answering programmes provide an environment wherein a user is encouraged to interact with the question-answering system in a manner which should facilitate the expedient retrieval of information and sometimes even related information. This is done by allowing users to query the system in a manner similar to how they would a person by allowing the use of anaphora.

Anaphora queries are utterances where a noun (known as the anaphor) has been replaced by a word or phrase (known as the referent) which refers directly back to a previous utterance of the missing noun.

An example:

How is John?
John is fine.
Where is he?

In this exchange the anaphor is “John” and the referent is “he”.

This report details the design and implementation of IQABOT; an interactive extension to pre-existing question-answering services. IQABOT will focus on pronominal anaphora resolution in follow-up factoid queries (although the question-answering service used can provide answers to non-factoid queries). It will allow a natural discourse to take place by emulating the behaviour of chatbots.
Acknowledgements

I would like to thank:
Dr. Jonathan Ginzburg for his guidance during this final year. He provided me with valuable resources and greatly stimulated my interest in the field of Natural Language Processing.

Bryan Taylor of KCLSU for being a friend and mentor to me in my final year. His assistance has been invaluable.

Matthew Moroz for his assistance when Python sometimes drove me to madness.
1 Introduction
  1.1 Overview ................................................. 7
  1.1.1 Question-Answering ........................ 7
  1.1.2 Interactive Question-Answering ........ 7
  1.1.3 Chatbots ...................................... 8
  1.2 Motivation ....................................... 8
  1.3 Project Aims .................................... 10

2 Background
  2.1 Question-Answering ................................. 11
  2.1.1 The Early Days .................................. 11
  2.1.2 The World Wide Web ............................. 11
  2.2 The Text REtrieval Conference (TREC) .......... 12
  2.2.1 Beginnings .................................... 12
  2.2.2 Tracks ........................................ 12
  2.3 Interactive Question-Answering .................. 13
  2.3.1 Defining Interactive Question-Answering ...... 13
  2.3.2 Approaches to Interaction ..................... 13

3 Review of Existing Systems .......................... 14
  3.1 Graphical Interaction Systems .................... 14
  3.1.1 High Quality Interactive Question Answering (HITIQA aka QUADRA) .......... 14
  3.2 Conversational Interaction Systems ............. 14
  3.2.1 YourQA ...................................... 14

4 Specification .................................. 18
  4.1 Requirements ................................... 18
  4.1.1 Functional Requirements ..................... 18
  4.1.2 Non-Functional Requirements ................. 18

5 Design .................................... 20
  5.1 Overview .................................... 20
  5.1.1 Multi-Tier Architecture .................... 20
  5.1.2 Programme Flow ............................. 20
5.2 User Interface .................................................. 20
5.3 Pronominal Anaphora Resolution ....................... 20
5.4 First Name Gender Classification ....................... 23

6 Implementation .................................................. 24
6.1 Language and Resource ...................................... 24
  6.1.1 Java and OpenNLP ........................................... 24
  6.1.2 Python and The Natural Language Toolkit ............ 24
6.2 Knowledge Base ............................................... 25
  6.2.1 START (Syntactic Analysis using Reversible Transformations) ........................................... 25
  6.2.2 True Knowledge ............................................. 25
6.3 User Interfaces ............................................... 26
  6.3.1 Graphical Interaction Interface ...................... 26
  6.3.2 Conversational Interface ................................. 26
6.4 Dialogue Management ...................................... 26
  6.4.1 Dialogue Moves ............................................ 28
  6.4.2 Dialogue Manager .......................................... 29
6.5 First Name Gender Classification ....................... 30
  6.5.1 Weighted Lists ............................................. 30
  6.5.2 Non-Weighted Lists ...................................... 31
6.6 Anaphora Resolution ....................................... 31
  6.6.1 Bindings .................................................... 31
  6.6.2 Query Rewriting .......................................... 31

7 Testing ............................................................ 33
  7.1 Regression Testing .......................................... 33
  7.2 Black-Box Testing ........................................... 33
  7.3 White-Box Testing .......................................... 34

8 Evaluation ......................................................... 35
  8.1 Anaphora Resolution ........................................ 35
    8.1.1 Evaluation and Results ................................ 35
  8.2 User Interaction ............................................. 36
    8.2.1 The Questionnaire ..................................... 36
    8.2.2 Evaluation and Results ................................ 36
  8.3 Name Genders Classifier .................................. 37
    8.3.1 Features .................................................. 37
    8.3.2 Evaluating The Classifier .............................. 37

9 Conclusion & Future Work .................................. 39
  9.1 Conclusion .................................................... 39
  9.2 Future Work .................................................. 39
    9.2.1 Interaction ............................................... 39
    9.2.2 Named-Entity Recognition .............................. 39
    9.2.3 Follow-up Query Rewriting .............................. 40
    9.2.4 Anaphora Resolution .................................... 40
    9.2.5 Question-Answering Service ............................ 40

Bibliography ......................................................... 43
A User Guide

A.1 Prerequisites
   A.1.1 Python
   A.1.2 PyYAML
   A.1.3 Natural Language Toolkit (NLTK)
A.2 Running The Program

B Anaphora Resolution

B.1 Evaluation Results
List of Figures

1.1 Google Search results regarding U.S. President Barack Obama’s age ........................................ 9

3.1 The QUADRA GUI .......................................................... 15
3.2 The YourQA user interface .............................................. 17

5.1 System Architecture ...................................................... 21
5.2 Flowchart .................................................................. 21
5.3 Use case diagram .......................................................... 22

6.1 IQABOT command-line interface .................................... 27
6.2 Design considered for graphical user interface .................. 27
List of Tables

6.1 User and system dialogue moves ................. 28
6.2 1990 US Census First Names List Fragment ....... 30
6.3 Bindings for anaphora resolution ................. 31

8.1 User questionnaire on use of IQABOT .............. 37
Chapter 1

Introduction

1.1 Overview

1.1.1 Question-Answering

Question-Answering (QA) is described by Harabagiu and Moldovan (2004) as a process aimed “at identifying the answer to a question in large collections of online documents.” When questions such as “What is the capital of Egypt?” and “How much does Jonathan Ross earn per year?” are posed to a question-answering system, the program should return the answers it finds in documents rather than just returning a link to a document that may contain the answer as search engines do.

There are a few different systems available to try online, and research is still incredibly active. MIT’s START\(^1\) (SynTactic Analysis using Reversible Transformations) is a fine example of ongoing research and learning. More commercial programs exist, such as the Wolfram Alpha Computation Knowledge Engine\(^2\) or the True Knowledge Answer Engine\(^3\).

1.1.2 Interactive Question-Answering

Interactive question-answering systems turn the focus to the interaction between the user and the program, not just the question-answering. These systems allow either the user to drive the dialogue or the system to play a greater role by suggesting related materials or even refinements to a users query.

Varges et al. (2009) describe the essential role of the interactive layer in a QA system as being something to further reduce the time it takes a user to find information on their subject of interest by means of natural language dialogue and/or interactive information interfaces.

An interactive layer in a QA system also allows for more complex queries to be issued by the user as mistakes made by the system can be rectified through user input; or the system can engage the user to check its understanding of a users query is correct.

---

\(^1\)http://start.csail.mit.edu/
\(^2\)http://www.wolframalpha.com
\(^3\)http://www.trueknowledge.com
1.1.3 Chatbots

Kerly et al. (2007) describe chatbots as “conversational agents, providing natural language interfaces to their users.” In this way they are well-suited for use as the interactive layer in a question-answering system designed with dialogue in mind.

When humans converse we generally do so by taking turns. These pairs of turns are known as adjacency pairs and this is also how chatbots engage in dialogue with their human users. This natural conversational method applies particularly well to QA systems due to the fact that when they are queried by a user they should return some meaningful result to continue the conversation. This is not to say that the system must know the answer to a query but rather that it should not break the Gricean Maxims (specifically that of Relation) as described in Grice (1995) by returning an answer that bears little relevance to the users utterance.4

1.2 Motivation

The success of search engines has shown that we, as a society, are particularly interested in accessing specific information in accordance with our short or long term goals without having to process what might be considered extraneous data. Search engines such as Google5, Yahoo6 and Bing7 use ever-complex algorithms to return documents which may contain the answer to a users search query. As time has progressed search engines have taken steps such as highlighting keywords in the results and showing the containing – or part thereof – sentence in the hope that the answer is also shown. The results can be somewhat hit and miss depending on how a user has formulated his/her query as is shown in figure 1.1 on page 9. As you can see it is by sheer coincidence that the answer appears in figure 1.1(a). It is also worth pointing out that not only did the search query “How old is President Barack Obama?” in figure 1.1(b) not return his age, it also contained a link suggesting he is 47 years old. Further investigation reveals that this news article was first printed in 2008 and so the information is outdated and incorrect; QA systems must return correct answers.

Existing question-answering programs already address the problem of returning a correct answer (and the supporting document) but they typically do not have any dialogue or other interactive capabilities. Recent community activity in question-answering (Vorhees, 2007; Song et al., 2010; Sun and Chai, 2007; Moore et al., 2006) signifies that there is an interest in the interactive aspect of QA and research will continue as this seems a natural progression in enhancing the user experience with question-answering systems.

In approaching the problem of creating a program that can interact with the user to reduce their search time we have two prevalent options available to us. We can write a chatbot interface to enable a natural dialogue between system and user or we can create a graphical user-interface. We shall explore the former.

4Even not knowing the answer is acceptable under these conditions so long as the system states that it does not know the answer.
5http://www.google.com
6http://www.yahoo.com
7http://www.bing.com
Figure 1.1: Google Search results regarding U.S. President Barack Obamas age
1.3 Project Aims

The project’s aim is to create a chatbot-based interactive question-answering system capable of pronominal anaphora in a user-driven dialogue. The intention is that a user will be able to collect data on a given subject faster and more reliably than they would using a regular search engine and/or an existing QA service.

To this end the following are requirements:

- A clean interface showing the dialogue.
- Answer a query posed in the user’s natural language (The system is restricted to English).
- Recognise follow up queries.
- Rewrite follow up queries for the answering module.
- Engage the user without cluttering the dialogue.
- Do not include extraneous data in an answer.
Chapter 2

Background

2.1 Question-Answering

This section outlines the history of question-answering.

2.1.1 The Early Days

Question answering is not a particularly new technology. Bert F. Green et al. (1961) created the BASEBALL question answering system in 1961. The BASEBALL program used the same type of analytical tools today's system use – such as part-of-speech tagging, sentence chunking and named-entity recognition.

However advanced BASEBALL may have been it was limited because it operated on a closed-domain. It could only deal with questions relating to baseball. As research continued, more complex systems were developed and with the advent of the world wide web, open-domain QA systems such as Murax by Kupiec (1993) were developed.

2.1.2 The World Wide Web

The web can be seen as a large corpus of redundant, unstructured data (Banko et al., 2002). There are several approaches to using the web for question-answering. Some involve deep parsing of web pages (Buchholz, 2001; Kwok et al., 2001); a time-consuming method that works well on text with little redundancy as it is able to extract meaning from a sentence containing perhaps an odd formulation of the answer.

E.g. “Who wrote Romeo and Juliet?”

Weaving a tale of love and sadness, Shakespeare left a great literary legacy in Romeo and Juliet.

as opposed to

William Shakespeare wrote Romeo and Juliet.

Another approach, used by Brill et al. (2001), is to take full advantage of the sheer amount of redundancy on the web. This is done by rewriting queries and
then using these reformulations to match an answer to several possible answer-templates based on the original and rewritten queries.

In using the web as a corpus, or rather a set of dynamic corpora\textsuperscript{1}, it was discovered that one of the webs greatest strengths was simultaneously its greatest weakness. A quick search of Google’s\textsuperscript{2} index using the search term “When was Nelson Mandela born?” returns approximately 5,060,000 results. It is beyond the lifetime of even the most biblical of humans to read all of these results but thankfully we don’t have to. Search engines are usually very good at picking out relevant documents by use of keywords; the document containing the sought after information is usually within the first 10 results. Some open-domain QA systems such as those presented in (Banko et al., 2002; Eetzioni et al., 2005; Kwok et al., 2001) take full advantage of this and only use between the first ten or twenty returned documents to search for an answer.

The final approach discussed here is to use structured data from the web. This makes it much easier to find the answer to a query as one should be able to extract keywords and relationships and then query the structured data for the correct answer. There are drawbacks to using this method as the data must be structured manually or semi-automatically (Auer et al., 2008) and this can be a time-consuming process which effectively limits the amount of available knowledge until the structured data is updated.

2.2 The Text REtrieval Conference (TREC)

Perhaps one of the most significant motivational forces in information retrieval is the Text REtrieval Conference (TREC)\textsuperscript{3}. TREC is an annual meeting of researchers from academia and business with a shared goal of improving technology in the various fields of information retrieval.

2.2.1 Beginnings

TREC began in 1992 with the intention of encouraging research, communication and commercialization of IR products. It can be considered a great success as more and more academics and businesses come together to discuss novel ideas and approaches to modern problems in question answering. Vorhees (2007) in fact claims that “The state of the art in retrieval system effectiveness has doubled since TREC began, and most commercial retrieval systems, including many Web search engines, feature technology originally developed through TREC.”

2.2.2 Tracks

TREC works by organizing problems in to “tracks”. Each track deals with a specific problem which is to be solved by some program. Past tracks have included:

Genomics

The Genomics track last ran in 2007 as part of information retrieval re-

\textsuperscript{1}http://www.google.com

\textsuperscript{2}Search engines will place the most relevant links first and these can change over time so a page parsed for an answer one week might not be available in a following session.

\textsuperscript{3}http://trec.nist.gov
search on a specific domain. The track data included gene sequences, research papers, lab reports and other relevant documents.

**Question-Answering**

The Question-Answering track last ran in 2007 and was intended to investigate methods for information retrieval, rather than document retrieval, on an open-domain. It consisted of 2 tasks; the Question-Answering task and the Complex Interactive Question-Answering task. The interactive aspect was designed to provide a framework for participants to investigate interaction in the QA context (Dang et al., 2007).

The annual conference continues to this day with academics and business being able to submit suggestions for future tracks in TREC.

### 2.3 Interactive Question-Answering

#### 2.3.1 Defining Interactive Question-Answering

Interactive question-answering systems can range from something as complex as the HITIQA\(^4\) system developed at Albany University to a relatively simple chatbot-based program such as YourQA developed at York University by Manandhar and Quarteroni (2007). Whilst the interfaces for these two systems are very different their commonality is that they are both driven by user interaction.

#### 2.3.2 Approaches to Interaction

There are two main approaches to interactive question-answering. One is through natural dialogue and the other is through interactions in a graphical environment. The former focuses on emulating a human-to-human dialogue with the user in order to make the session seem as natural as possible. This kind of emulation is usually facilitated by allowing for follow-up queries and clarification utterances\(^5\) (Van schooten et al., 2009).

The latter, graphical, approach allows for much more user interaction at the cost of losing some of the natural feel of a dialogue based system. The graphical environment increases the user’s role by allowing them to disambiguate between a range of possible answers, respond to the system in order to filter out irrelevant text, raise the perceived importance of certain keywords, check the correctness of an answer and increase the search space in real-time (Small and Strzalkowski, 2009).

The graphical interactive model is more popular than the dialogue model possibly owing to the fact that it allows for more complex interactions between the system and user.

\(^4\)http://www.laancor.com/technology/quadra/

\(^5\)An utterance that requests the system or user clarify their previous utterance.
Chapter 3

Review of Existing Systems

Talk about reviewing other systems.

3.1 Graphical Interaction Systems

3.1.1 High Quality Interactive Question Answering (HITIQA aka QUADRA)

HITIQA\textsuperscript{1} is an advanced question-answering system developed by Albany University in association with Laancor\textsuperscript{2}. The programme is an incredibly complex interactive question-answering tool making active use of a graphical user interface to encourage interaction between the system and user (see figure 3.1 on page 15). The interaction however lacks dialogue and can seem very complicated as a user clicks on circles and diamonds until eventually resorting to the help files and an online video tutorial.

HITIQA can definitely be viewed as an expert system requiring training and patience. However, after a few hours using the system and reading of the help files it was found to be playing to its strengths very well. HITIQA is designed to be a research assistant tool and it performs this job solidly; working on gathering candidate answers and paragraphs from unstructured text quickly and accurately and always returning useful data.

3.2 Conversational Interaction Systems

3.2.1 YourQA

YourQA is an interactive question-answering system developed at York University in the United Kingdom. It takes a chatbot approach (Manandhar and Quarteroni, 2007) in its attempt to enable dialogue interaction between itself and the user.

The interface for YourQA (see figure ?? on page ??) is designed to be simple and absolutely free of clutter. Dialogue takes place in a panel on the left side of

\textsuperscript{1}http://www.laancor.com/technology/hitiqa/
\textsuperscript{2}Laancor have subsequently gone on to release the programme as QUADRA in a closed beta.
Figure 3.1: The QUADRA GUI
the display whilst answers are displayed on the right. The answer panel takes
the step of colouring text that may be of interest to the user; this includes
named-entities and terms matched in the query. Interestingly there is also an
HTML link to the supporting document. This can be a useful addition for those
using YourQA as a research tool.

YourQA is incredibly easy to use. A complete novice user can jump right
in and begin asking questions to the system. I much prefer using this system
to the QUADRA programme. It is far more naturally intuitive and requires no
learning whatsoever.
Figure 3.2: The YourQA user interface

Welcome!
Your: Hello.
Your: Who is this?
Your: I want to know when Shakespeare was born.
Your: And when he wrote Lear.
Ooops, one question at a time... Shall we start with when Shakespeare was born?
Your: yes, please.

The answers are on the right (you can click the links to know more). Can I help you further?

Results

Your query: When Shakespeare was born?

Text colors: ORANGE = NUMBER, TURQUOISE = TIME, RED = ORGANIZATION, GREEN = LOCATION, BLUE = PERSON, PURPLE = MATCHED TERMS, NAVY = QUERY TERMS

Answers

1. Title: Shakespeare's Birth. William Shakespeare. URL: http://www.shakespeare.com/shakespeare/birth.html. Google Rank: 3. William Shakespeare was born on April 26, 1564. There is no specific date of birth, because at the time the only date of importance was the date of baptism. Typical baptisms were held within three days after birth. Shakespeare's baptism date was April 26, 1564.


William Shakespeare was born on April 26, 1564. There is no specific date of birth, because at the time the only date of importance was the date of baptism. Typical baptisms were held within three days after birth. Shakespeare's baptism date was April 26, 1564.
Chapter 4

Specification

4.1 Requirements

4.1.1 Functional Requirements

In order for the programme to perform well in its role of a question-answering system with the ability to perform anaphora resolution and query rewriting the following specifications must be met:

Resolve anaphora follow-up queries
   Recognise an anaphoric query and correctly resolve anaphors and referents so that a grammatically valid query is sent to the answer service.

Correctly identify elliptic queries
   Recognise when the user has asked an elliptical question. This is necessary in order to ensure the query is not sent to the answering service. Elliptic queries are identified but they are not resolved in IQABOT.

Interface with existing question-answering service
   Use existing question-answering services in order to maintain the systems status as a chatbot-like interface. This will allow the system to be used with several different QA platforms.

Correctly guess the most likely gender of a name
   Gender agreement is important for being able to bind the referent with a correct anaphor. i.e. binding “he” with “William”.

4.1.2 Non-Functional Requirements

Simple user-interface
   Chatbots generally have very simple interfaces; showing only what is necessary to continue the flow of a conversation. With this in mind IQABOT will use a very simple command-line interface which will display previous queries and answers as adjacency pairs.

Timely response
   The system should return answers in a few seconds, rather than minutes.
User driven dialogue
In the system outlined in (Quarteroni and Manandhar, 2009) the program issues a follow-up proposal after it returns an answer. Whilst this, at first, seems to be a good idea, they found in their evaluation, detailed in Manandhar and Quarteroni (2007), that users tended not to reply to the chatbot offers and instead directly entered their own follow-up questions. For this reason IQABOT will only return answers and will not engage the user in such a way.

Be modular
IQABOT is designed to be as modular as possible so that other developers may plug their own modules in to IQABOT or so that they can use a module from IQABOT to extend their own program. This is important as there are no current "best
Chapter 5

Design

This chapter discusses design somehow!!!

5.1 Overview

A good initial design of the overall system will allow progresssession through the project without having to make too many sizeable changes which would have an overly negative affect on use of time. Obviously we expect changes will have to be made on the small scale but these can be dealt with as the causes of these changes are encountered.

5.1.1 Multi-Tier Architecture

Many systems use the 3 layer model of the multi-tier architecture because of its logical split between presentation tier, business logic tier and data tier. We will be no exception and a broad figure for our system is given in figure 5.1.

5.1.2 Programme Flow

As we deal with natural dialogue it is important to abstract away from the details to get an idea of how the programme should progress with each utterance.

As such we define a flow chart used to capture the mid-level actions involved in processing a users utterance in figure 5.2 and a use case diagram in figure 5.3.

5.2 User Interface

The user interface is an incredibly important aspect of this project. If should not be so complicated that it hinders users but should be complex enough to display the necessary information.

5.3 Pronominal Anaphora Resolution

Pronominal anaphora resolution is the task of attempting to resolve a referent to a suitable anaphor in a previous utterance. The system will focus on resol-
Figure 5.1: System Architecture

Figure 5.2: Flowchart
Figure 5.3: Use case diagram
ving pronominal anaphora in the form of 3rd person personal and possessive
pronouns. It will not deal with demonstrative pronouns.

5.4 First Name Gender Classification

An import aspect of pronominal anaphora resolution is gender agreement. When
a user references a person (or even some objects or locations such as ships and
countries) in a previous utterance it is crucial that we are able to correctly
identify the gender of the object being referred to. There are two main methods
we can use in an attempt to correctly identify the gender of a name: weighted
and non-weighted names lists. The favourite is clearly a weighted list as this
would allow us to design a classifier that takes in to account how popular a
gender-neutral name is with one particular gender.
Chapter 6

Implementation

This chapter details the reasoning behind the choices made with respect to some of the tools, platforms and functionality used by IQABOT.

6.1 Language and Resource

A list of open-source natural language toolkits was compiled. Of this list 2 languages showed dominance and 2 platforms stood out; The Java language with OpenNLP and Python with The Natural Language Toolkit. They are reviewed below.

6.1.1 Java and OpenNLP

Java is a statically typed, interpreted programming language whose programmes are compiled into a Java specific byte-code which can be run on any system for which a Java Virtual Machine has been written.

Having had previous experience writing programmes in Java I first decided to evaluate the OpenNLP\(^1\) project. The OpenNLP library and API were extensive and documented enough for someone who already had experience in natural language processing. A study by Prechelt (2000) showed that programmes written in Java ran faster than those written in Python and as some areas of linguistic processing are particularly time-consuming processes this was believed to be a substantial argument in favour of writing IQABOT in Java.

6.1.2 Python and The Natural Language Toolkit

Python, much like Java, is an interpreted language whose programmes can be run on any architecture for which there exists a Python interpreter. Unlike Java, however, Python is dynamically typed; this frees the programmer from having to concern himself with variable types\(^2\). Python is also described by various

---

\(^1\)http://www.opennlp.com

\(^2\)It is understood that dynamic typing does allow for runtime errors. Although static typing would catch any such errors at compile-time it is expected that almost all operations will be performed on strings or lists of strings.
professors engaged in natural language processing as “a simple yet powerful programming language with excellent functionality for processing linguistic data.” in Bird et al. (2009).

As Java has OpenNLP, Python has The Natural Language Toolkit (NLTK). NLTK, like OpenNLP, is also a large library and and comes complete with a well documented API. One can not overlook the fact that NLTK also has a book instructing in its use with several examples on using some of the modules included in the toolkit. Whilst Python has the benefit of being very easy to read and NLTK is written with the intention of being used as a teaching tool it should be stated that the code provided by the natural language toolkit is not optimized and will perform slower than code written in Java.

However, after some benchmarking tests it was decided that, although Python was slower than Java, the Natural Language Toolkit would be used over OpenNLP due to the fact that Python is easier to read, written code is generally shorter and NLTK is better for those who are new to natural language processing with supporting features in the form of a book, extensive documentation and demonstration code.

### 6.2 Knowledge Base

Herein we review two websites offering question-answering services. They were selected for review because they are able to return short answers suitable for a chat dialogue interface.

#### 6.2.1 START (SynTactic Analysis using Reversible Transformations)

START\(^3\) is a natural language question-answering system developed by Dr Boris Katz in the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. START is a very mature software system; existing in some form since the early 1980s (Katz, 1997). It was put online in 1993 and has continued to serve answers to natural language queries to this day.

Whilst START is very mature it lacks an API and the structured response page\(^4\)is in its infancy; sometimes not returning anything at all. For this reason START will not be used as the default knowledge base.

#### 6.2.2 True Knowledge

True Knowledge\(^5\) (TK) is a natural language question-answering service provided by the True Knowledge company based in Cambridge, UK. The service entered public beta\(^6\)on the 24th of February 2010 after a lengthy, 2 year private beta.

TK offers a simple-to-use API with several examples on using the service. There is also an active support forum for developers and regular news updates

\(^3\)http://start.csail.mit.edu/

\(^4\)A structured response is considered here as any response that is machine readable. XML appears to be the de facto standard with both True Knowledge and START offering query responses in these formats.

\(^5\)http://www.truenewknowledge.com
on changes to the service. Due to the good documentation, simple API and support True Knowledge was chosen as the default knowledge base for IQABOT. It is worth noting that the system can use any question-answering service that accepts natural language input and returns a clear, concise text answer. This feature is, however, not implemented at the time of writing due to a lack of publicly available QA services.

6.3 User Interfaces

The choice of user interface for this project is arguably the single most important design decision to be made. We must design an interface free of clutter; one which will allow the user to obtain their desired information with speed and accuracy that outdoes a search engine and a non-interactive question-answering system. To this end a comparison of dialogue and graphical interaction based interfaces were evaluated and a model was eventually chosen for IQABOT.

6.3.1 Graphical Interaction Interface

Graphical user interfaces or GUIs are by far the most prevalent form of interface in programmes today. They enable rapid and far more complex interactions than a command-line interface allows for. This level of complexity however comes at a cost as users must take time to familiarize themselves with the actions available to them in various contexts.

HITIQA, as reviewed in 3.1.1, is a fitting example of a very complex and complicated GUI with a steep learning curve.

6.3.2 Conversational Interface

Conversational interfaces can range from a simple command-line to a more complete interface displaying other information that may be of interest.

Van schooten et al. (2009) argue in favour of using a conversational interface as question-answering is already an instance of the conversational paradigm. This point is especially material as we are designing a program which allows for follow-up queries and so we should present the user with an interface which should not only feel natural in its use but also encourage a natural language interaction between the user and programme.

For these reasons a simple command-line was chosen as the standard interface for IQABOT. As can be seen in figure 6.1 it fits well with the projects aims of being uncluttered and only showing the necessary information. It should be noted however that a GUI was designed but development was abandoned in favour of the command-line interface (see 6.2).

6.4 Dialogue Management

Ensuring a successful dialogue is paramount in facilitating an expedient retrieval of information for the user. To this end we should define the user and system

6A “public (or open) beta” is a stage in the development cycle of a program where the system as a whole is opened up to testing by interested party’s outside the developers organisation.
Figure 6.1: IQABOT command-line interface

Figure 6.2: Design considered for graphical user interface
Table 6.1: User and system dialogue moves

<table>
<thead>
<tr>
<th>User move</th>
<th>Description</th>
<th>System move</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>greet</td>
<td>Begin dialogue</td>
<td>greet</td>
<td>Greet user</td>
</tr>
<tr>
<td>exit</td>
<td>Close dialogue</td>
<td>exit</td>
<td>Response to user exit</td>
</tr>
<tr>
<td>ask(q)</td>
<td>Ask question q</td>
<td>answer(a)</td>
<td>Answer a to question q</td>
</tr>
<tr>
<td>command(c)</td>
<td>Use built-in command c</td>
<td>response(r)</td>
<td>Response r triggered by command c</td>
</tr>
<tr>
<td></td>
<td></td>
<td>clarify(q)</td>
<td>Clarify question q</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ground(q)</td>
<td>Ground question q</td>
</tr>
</tbody>
</table>

 moves as well as describing a dialogue management schema.

6.4.1 Dialogue Moves

Larsson and Traum (2000) state that “Dialogue moves are meant to serve as an abstraction between the large number of different possible communications (especially in a natural language) and the types of update to be made on the basis of these communications.”. IQABOT does not expressly describe or store dialogue moves but they are inferred by the conversational nature of the programme. The moves, outlined in table 6.1 can be considered a subset of those described in Manandhar and Quarteroni (2007).

These moves and their properties are:

1. A greeting (greet move), command (command(c) move) or a query (ask(q)) to initiate the dialogue.

2. Utterance is analysed to see if it is a command or a query.

3. If utterance is a command c perform the matching action.

4. If utterance is a query q check if it is a follow-up.
   (a) If q is unrelated to a previous query submit it to the question-answering service.
   (b) If q is an elliptic follow-up ask user to reformulate their query (clarify(q) move).
   (c) If q is an anaphoric follow-up then attempt to rewrite to q' by replacing references with named entities discovered in previous queries and/or answers.
      (i) If multiple possible reformulations are discovered ask the user which query they intended to ask (ground(q) move) and submit query to QA service.
      (ii) If no reformulations are found ask the user to reformulate the query themselves (clarify(q) move)

5. Return answer a or state that no answer was found (answer(a) move).

6. If the user wishes to quit the programme (exit move) then exit gracefully.
6.4.2 Dialogue Manager

Having defined the possible moves available in the discourse we must now select a dialogue management model. There are 3 notable models discussed in Manandhar and Quarteroni (2007) but we have chosen a fourth option due to limitations and unexpected problems.

Finite-State (FS) Model

In a finite-state model we must model each stage of the conversation as a separate state and encode the dialogue moves (see table 6.1) as transitions between these states. This is a simple and very limited model which is useful for dialogue occurring in a closed domain. However, as IQABOT is designed for open domain dialogue we dismiss the FS as a viable model.

Information State (IS) Model

The information state model was developed and demonstrated by Larsson and Traum (2000) in the TRINDI\textsuperscript{7} project. IS theory is largely inspired by work done by Ginzburg in his papers on gameboard dialogue theory; specifically citing (Ginzburg, 1996a,b, 1998).

The IS model is a powerful tool for understanding the context and direction of a conversation. This power, however, is what harms its potential use as it is “too voluminous” (Manandhar and Quarteroni, 2007) for use in a question-answering context.

The Chatbot Approach

Quarteroni and Manandhar (2009) also discuss, and eventually adopt, what they call the “chatbot approach”. This approach uses the Artificial Intelligence Markup Language (AIML) in order to match a user’s utterance against a database of dialogue patterns and return a suitable response (known as a response template). The patterns and responses are part of what make up a “category” in AIML.

AIML seems very fit for purpose as it was originally designed for conversational chatbots. Numerous chatbots have been designed with the use of AIML as a dialogue manager including the original AIML chatbot ALICE\textsuperscript{8}.

AIML also enables a limited notion of context by allowing the topic attribute to be changed when a pattern is matched.

An example fragment of an AIML category:

\begin{verbatim}
<category>
<pattern>WHO IS *</pattern>
<think><set name="topic">person</set></think>
<template>
I don’t know who <star/> is.
</template>
</category>
\end{verbatim}

\textsuperscript{7}http://www.ling.gu.se/projekt/trindi/
\textsuperscript{8}http://www.alicebot.org
In this small example we can see that the topic has been changed to “person” as the user is obviously asking about a person. This will now allow the programme to make assumptions about the context of any latter queries. This is especially helpful with respect to follow-up queries.

**Regular Expression Pattern Matching**

Before AIML it was common to try to match an utterance against a list of regular expressions. The idea is similar to AIML but far less complex whilst sometimes being slightly more complicated. Unfortunately due to an inability to get either of the two available Python AIML interpreters to work we decided to go with this option instead. This means that there is no notion of context and dialogue management is split between a module checking for commands and one checking for queries.

### 6.5 First Name Gender Classification

An important function within IQABOT is the ability to distinguish between the likely genders of a name. This is something which has been explored by Bird et al. (2009). I have improved upon their work by increasing the size of the corpus and looking at different featuresets.

Selecting a corpus for this task was crucial as it was important that the training data would cover as large a set of names as possible. There were two main candidates to be used as training data. They are described below.

#### 6.5.1 Weighted Lists

Weighted lists present us with more data than a non-weighted list might. These types of list will contain some kind of information about the object in question. For us this means displaying the likelihood of a first name belonging to a certain gender within a sample.

The published data on the 1990 US Census uses the following formatting:

<table>
<thead>
<tr>
<th>Name</th>
<th>Frequency (%)</th>
<th>Cumulative Frequency (%)</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>3.318</td>
<td>3.318</td>
<td>1</td>
</tr>
<tr>
<td>John</td>
<td>3.271</td>
<td>6.589</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Donny</td>
<td>0.009</td>
<td>87.189</td>
<td>709</td>
</tr>
</tbody>
</table>

With this data we can begin to guess the likelihood of a known name belonging to a certain gender by checking how much more likely it is to occur in one gender than in the other. The drawback to this method lies in the fact that this type of data is relatively rare. Unfortunately the amount of data available to us was severely limited and we found it unreliable when guessing the genders of names not in the lists.
6.5.2 Non-Weighted Lists
Non-weighted lists contain only the objects of interest. In our case we are dealing with first names. With this type of information we can only extrapolate the patterns which give us the most information about which gender a name is likely to belong to - we do not know how much more likely a name is to belong to one gender according to its frequency. This type of data is readily available and can be found all over the Internet. As it is so much easier to expand upon a non-weighted list this type of corpus was chosen to be used as training data.

6.6 Anaphora Resolution
Anaphora resolution is an important aspect of question-answering for 2 reasons. (1) If we are able to resolve anaphora in text then we have more data available for answer selection. (2) We can build complex interactive question-answering systems emulating natural dialogue.

Anaphora resolution in IQABOT is implemented in a knowledge-poor manner (Lappin, 2003). It uses only syntactic information to replace referents with their target anaphora.

6.6.1 Bindings
Bindings are the rules specifying which referents (3rd person personal pronouns and possessive pronouns) are allowed to be used to refer to named-entity types.

<table>
<thead>
<tr>
<th>Named-Entity Type</th>
<th>Referents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>he, him, his, they</td>
</tr>
<tr>
<td>Female</td>
<td>she, her, hers, they</td>
</tr>
<tr>
<td>Organization</td>
<td>it, its, their, theirs, they</td>
</tr>
<tr>
<td>Location</td>
<td>it, its, that, there</td>
</tr>
<tr>
<td>Date</td>
<td>this, that</td>
</tr>
<tr>
<td>Money</td>
<td>this, that</td>
</tr>
<tr>
<td>Percent</td>
<td>this, that</td>
</tr>
<tr>
<td>Facility</td>
<td>it, its, she, her, hers, there</td>
</tr>
<tr>
<td>GPE (Geo-Political Entity)</td>
<td>it, its, that, there</td>
</tr>
<tr>
<td>GSP (Geo-Social Party)</td>
<td>them, they, their, theirs</td>
</tr>
</tbody>
</table>

Table 6.3: Bindings for anaphora resolution

6.6.2 Query Rewriting
Once an allowed binding has been found the anaphor replaces the referent in a reformulation of the original follow-up query. If multiple allowed bindings come from a single previous utterance then the first anaphor is chosen over the others.9

Example of a rewrite occurring:

9There is no reason for this behaviour other than there is currently no functionality to account for multiple reformulations of a query using just one previous utterance.
User: What is the distance between London and Newcastle?
Sys: Approximately 485.92 kilometers.
User: How many people live there?
Rewrite: How many people live in London?

As you can see from this exchange as London is the first named-entity (or anaphor with respect to “there”) that can be bound to the referent “there” it is used to rewrite the follow-up query.
Chapter 7

Testing

Testing of the programme was carried out at various stages of the project lifecycle. The findings from these tests are described below.

7.1 Regression Testing

Regression testing was used throughout all stages of implementation. Every time code was changed, added or deleted tests were carried out to ensure quality of output. This also allowed for the testing of fixes and to make sure new bugs were not being introduced.

This type of testing worked very well in the projects environment as the modules, classes, methods and functions in IQABOT are very small; doing only what they need to do. In this way it was very easy to spot where bugs were emerging or fixes were simply not working at all.

The stack viewer and debugger in Python Shell\(^1\) were used to great effect; providing a means to see what was being stored in variables and what functions were returning at any stage. This meant there was no need to litter the code with print statements for debugging purposes.

7.2 Black-Box Testing

In order to implement black-box testing the assistance of two Computer Science students was arranged. They were shown various dialogues and asked to come up with their own queries in order to test as much of the system as possible. It was through this testing that a bug which tagged follow-up queries as not being so was discovered and fixed.

Another bug was also discovered. This particular bug threw an IndexError whenever a referent could not be resolved using any of the previous utterances (up to 8 places).

\(^1\)Python Shell is an interactive development environment allowing for Python code to be written and run all in the same window.
7.3 White-Box Testing

In the same way the Python stack viewer and debugger were used for regression testing they were also used for white-box testing. Through careful selection of dialogues we were able to test all predicate paths in the programme. This had the benefit of showing where some unhandled exceptions had occurred. Namely in the handling of xml data from the question-answering service.
Chapter 8

Evaluation

8.1 Anaphora Resolution

This section documents an evaluation of the anaphora resolution module in IQABOT. The module deals with the identification of follow-up queries and rewrites for those queries. It is able to rewrite queries using previous utterances from both the query and answer stacks - a feature noted as being missing in YourQA (Manandhar and Quarteroni, 2007).

8.1.1 Evaluation and Results

The module was tested on 16 follow-up queries generated from randomly selected questions as used in the TREC 2008 Question-Answering Track. The results can be seen in appendix B.1.

Although the test set is small we can quickly recognise that the main problem occurs at the named-entity recognition phase. The incorrect classifying of entities means that bindings from referent to anaphor are unable to occur within the defined ruleset (see table 6.3). This has the effect that when a user types in what they believe to be, and indeed is, a correct referent for their intended target pronoun the system is unable to correctly rewrite the follow-up query. This can lead to grammatically incorrect rewrites, incorrect answers or no rewrite at all.

As mentioned, the resolution module can sometimes return grammatically incorrect rewrites. Part of this is to do with incorrect named-entity classification but part is also due to a lack of extensive rewrite rules. This is noted as something to be seen to in future work.

In brief we can see that:

7 follow-up queries were rewritten correctly.
1 follow-up query was rewritten with a grammatical error.
4 follow-up queries were rewritten correctly but missed out on a rewrite from either the query or answer stack owing to an incorrect named-entity classification.
1 follow-up query was rewritten correctly but demonstrated the programmes inability to rewrite a query using multiple named-entities from a single previous utterance.
3 rewrites were incorrect. These were all associated with an incorrect named-entity classification.

With these results we can see that the weakness of the anaphora resolution module lies in poor classification from the named-entity module. If all entities were tagged correctly the total number of correct rewrites would have been 14/16. The final 2 remaining incorrect rewrites could be corrected by introducing more complex rewrite rules.

8.2 User Interaction

Five users were each given a brief explanation of precisely what constituted an anaphoric query and were then asked to evaluate their experience with the system. Of the five users four are Computer Science students and one was a novice computer user. They were each told to choose 1 broad topic of interest get as much information as they could in 5 minutes.

8.2.1 The Questionnaire

The questionnaire the users were asked to fill in once they had completed their session with the system is based on that used in Dang et al. (2007). The questionnaire consists of 6 questions evaluating ease of use, usefulness and effectiveness using 5-point scales in ascending order; so the lower the number the better.

They are:

1. How easy was it to understand how to interact with the service?
   Easy → Difficult

2. How coherent was the interaction?
   Coherent → Incoherent

3. How stimulating was the interaction?
   Stimulating → Dull

4. How much did the interaction help you think about your topic in new ways?
   A lot → Not much

5. How much did you learn about your topic during this interaction?
   A lot → Not much

6. Overall how would you rate the quality of the interaction?
   Excellent → Poor

Users were also invited to make comments.

8.2.2 Evaluation and Results

Questions are shown here by their number. Refer to above list for actual questions.
Table 8.1: User questionnaire on use of IQABOT

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>&quot;No answer&quot; returned a few times.</td>
</tr>
<tr>
<td>User 2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>Could use a nicer GUI. Did not know the answer to most of my questions.</td>
</tr>
<tr>
<td>User 3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>Had to rewrite some of the queries.</td>
</tr>
<tr>
<td>User 4</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>Would have given higher overall mark but a few of my questions weren’t answered.</td>
</tr>
</tbody>
</table>

8.3 Name Genders Classifier

8.3.1 Features

An important part of being able to correctly assign the most likely gender of a name are the use of features which are uniquely common to male and female names. As such I experimented with several different feature sets; starting with an over-filled feature set and using the show_most_import_features() method of NLTK’s Classifier class to identify which features were most useful. Using this methodology I quickly ascertained that the 3 most useful features were the ones looking at the last, last two and first two letters in a name.

8.3.2 Evaluating The Classifier

The figures given here are for classifiers trained on 9500 names in the extended corpus.

```python
def book_features(self):
    features = {}
    features['suffix1'] = name[-1:]
    features['suffix2'] = name[-2:]

    return features
```

```python
def iqabot_features(self):
    features = {}
    features['first-two'] = name[:2].lower()
    features['last-two'] = name[-2:]

    return features
```

To evaluate the accuracy of the feature sets, the test data, consisting of 500 randomly chosen, unseen names, was generated 100 times. The classifier was then evaluated on each iteration. The book_features feature set achieved
an average accuracy of 79% whilst the iqabot_features feature set raised the accuracy to 83% (415 names guessed correctly out of 500).
Chapter 9

Conclusion & Future Work

9.1 Conclusion

IQABOT has proven that it is possible to apply simple, knowledge-poor pronominal anaphora resolution on queries and expect accurate rewrites to occur on follow-up queries so long as there is accurate classification of named-entities in previous utterances.

We have shown that by enabling interaction between the programme and the user we can enhance the users overall experience but this must be done carefully so as to not alienate novice users and not be too restrictive for experienced users.

IQABOT also demonstrates the importance of using answers from a previous adjacency pair to enhance the interaction between user and system in encouraging follow-up queries and natural dialogue. Previous systems such as YourQA explore use of just the query stack in follow-up resolution. In our system the user is allowed to engage the programme in a manner similar to how they would engage another human when attempting to explore a topic.

9.2 Future Work

There are a few weaknesses in IQABOT that require further refinement. They are discussed below.

9.2.1 Interaction

As interaction is at the heart of this project it was disappointing to see that the evaluation scored it so low. One tester went so far as to say it “could use a nicer GUI.”. In order to remedy this problem I would finish the GUI shown in figure 6.2. This should allow for a richer interaction and will also remedy some of the problems presented with the use of inaccurate named-entity classifiers.

9.2.2 Named-Entity Recognition

The named-entity classifier included with the Natural Language Toolkit was essential to the main operations of anaphora resolution and query rewriting and so as it failed to classify entities correctly the other modules could not work to
their full effect. Given time I would investigate other named-entity classifiers in an effort to increase the accuracy of this important tool.

9.2.3 Follow-up Query Rewriting
Some query rewrites require the use of more complex algorithms in order to perform a correct, grammatical rewrite. This aspect of the interaction is important so as to ensure that when rewritten queries are sent to the question-answering service they are correct.

9.2.4 Anaphora Resolution
As it stands the system is able to detect the presence of elliptic queries but it is unable to resolve them. As elliptic queries do occur in natural dialogue it seems fitting that in the future this functionality should be implemented.

9.2.5 Question-Answering Service
A common complaint from almost all the testers was that the question-answering service performed rather poorly on certain questions. IQABOT has the capability to plug in to any existing question-answering service but this functionality was not implemented in time for the project deadline. This feature would be an integral part of any future developments in order to ensure that the final product achieves the projects aims.
Bibliography


Appendix A

User Guide

A.1 Prerequisites

The following list features software that IQABOT requires in order to function. You should install the software in the order presented. The web page at http://www.nltk.org/download has clear, step-by-step instructions, for several platforms, on installing the programs mentioned herein.

IQABOT was written with and tested on the following software platforms:

- Python 2.6.4
- PyYAML 3.09
- NLTK 2.0b8

A.1.1 Python

Python is a platform-independent, dynamically typed, interpreted language designed with the intention of being easy to read for the developer. It is available for Windows, Linux and Mac operating systems.

Please see the Python download page here: http://www.python.org/download/. Please note that IQABOT has only been tested with Python version 2.6 on Windows and Ubuntu Linux and will not work with Python 3.0 or above.

A.1.2 PyYAML

PyYAML is a Python implementation of the YAML (YAML Ain’t Markup Language). YAML is, in turn, a human-readable data serialization format. It is outside the scope of this project to go in to any detail but it is a prerequisite for using the Natural Language Toolkit (see below).

You should install the latest version available that will work with Python 2.6. See the downloads page here: http://pyyaml.org/wiki/PyYAML.

A.1.3 Natural Language Toolkit (NLTK)

The Natural Language Toolkit is a set of open-source tools written in Python for natural language processing. It has been designed from a teaching perspective
in order to make it easier for students to acquire a working knowledge of natural language processing. This does mean that it may not be as memory and speed efficient as other systems but it is easier to read and comprehend the source code. IQABOT makes use of NLTK version 2 and for best results this or a more recent version should be used. Please see the downloads page here http://www.nltk.org/download.

A.2 Running The Program

IQABOT can be launched using either a command-line interface or the Python Interpreter.
Appendix B

Anaphora Resolution

B.1 Evaluation Results

1. What does the Peugeot company manufacture?
   ANS: NO ANSWER
   FQ: What was their total revenue for last year?
   RW: What was Peugeot's total revenue for last year?
   RES: Correct

2. Who was President of Costa Rica in 1994?
   ANS: Rafael Angel Calderon Fournier and Jose Maria Figueres
   FQ: Where was he born?
   RW: Where was Costa Rica born?
   RES: Incorrectly identifies Costa Rica as a male person and fails to identify either of the people in the answer as persons.

3. Why did David Koresh ask the FBI for a word processor?
   ANS: NO ANSWER
   FQ: What is his nationality?
   RW: What is David Koresh's nationality?
   RES: Correct

4. What debts did Qintex group leave?
   ANS: NO ANSWER
   FQ: When were they founded?
   RW: When were Qintex founded?
   RES: Correct

5. What is the brightest star visible from Earth?
   ANS: NO ANSWER
   FQ: How big is it?
RW: How big is Earth?
RES: Correct.

6. Where was Ulysses S. Grant born?
   ANS: Point Pleasant, the place in Clermont County, Ohio, USA
   FQ: When did he die?
   RW: When did Ulysses S. Grant die?
   RW: When did Point Pleasant die?
   RES: Correct but Point Pleasant incorrectly identified as male person.

7. What is the capital of Germany?
   ANS: Berlin, the large city in eastern Germany
   FQ: What is its population?
   RW: What is Germanys population?
   RES: Incorrectly tagged Berlin as person. Rewrite ok.

8. Where is the actress, Marion Davies, buried?
   ANS: NO ANSWER
   FQ: When did she die?
   RW: Error resolving query. Incorrect gender.
   RES: Incorrectly tagged Marion Davies as male.

9. What is the name of the highest mountain in Africa?
   ANS: Mount Kilimanjaro
   FQ: How big is it?
   RW: How big is Africa?
   RES: Incorrectly tagged Mount as person and Kilimanjaro as organization.

10. Who is the president of Stanford University?
    ANS: John LeRoy Hennessy, the American computer scientist and academic
    FQ: How old is he?
    RW: How old is John LeRoy Hennessy?
    RES: Correct

11. When did Lucelly Garcia, a former ambassador of Columbia to Honduras, die?
    ANS: NO ANSWER
    FQ: How many people live there?
    RW: How many people live Columbia?
    RES: Grammatically incorrect.
12. When did Nixon die?
   ANS: April 22nd 1994
   FQ: Who was his wife?
   RW: Who was Nixons wife?
   RES: Correct

13. Where is Microsoft’s corporate headquarters located?
   ANS: Microsoft Headquarters is a building or building complex in Redmond, Washington.
   FQ: How many employees do they have?
   RW: How many employees do Microsoft Headquarters have?
   RES: Incorrectly recognises Microsoft (from query) as a person.

14. Where is the Taj Mahal?
   ANS: Taj Mahal is a mausoleum in Agra, India.
   FQ: Who designed it?
   RW: Who designed Agra?
   RW: Who designed Taj Mahal?
   RES: Incorrect tagging of Agra.

15. What is the capital of England?
   ANS: London, the capital of England and the United Kingdom
   FQ: How big is it?
   RW: How big is England?
   RW: How big is London?
   RES: Correct