The University of York Department of Computer Science

Submitted in part fulfilment for the degree of BEng.

Bringing Knowledge Through AI and SMS

Sam Heather
sam@heather.sh

16th April 2015

Supervisor: Dr. Lilian-JL Blot

Number of words = 999999, as counted by wc -w.
This includes the body of the report only.

Abstract

In remote Africa there are millions of disadvantaged and uneducated individuals, who in the vast majority do not have access to the internet the knowledge that this brings. Outside of their immediate friends and family, individuals cannot access the information they need to further their education, or just for general interest.

In many parts of the world, the number of people without access to the internet but access to a mobile phone is significant. This project aims to research and develop a system capable of bringing knowledge through a question and answer based interactive system, in the language natively spoken by the user, through the use of a simple Artificial Intelligence and an SMS interface. The system will be expandable, such that it can be adjusted to handle questions on any knowledge area.

This project raises ethical issues relating to the responsibility of providing accurate information when in a position of trust, the ethics of machine translation and maintaining user privacy.

It is early days so it is normal that many things are missing from the abstract. You should keep in mind that it should include later on few sentences on method/experiment, result and conclusion.

Contents

Contents

List of Figures

List of Tables

Statement of Ethics

Pretty much the most important part - write this! - data protection - users information irreversibly obscured and protected in the database to protect them in case of database hack. - data protection - supervisor had root password to the server. - experiment - participants were asked to sign consent forms, these were stored with supervisor. - after experient, all participants were anonymised by a unique participant id, and the only mapping of these to names is the consent form, which the supervisor has - ethics of providing information - steps taken to try and prevent wrong info been returned

Acknowledgement

The author would like to acknowledge the following individuals for their contribution to this project:

- SpazioDati (http://spaziodati.eu/) for providing extended access to the Dandelion Text Semantics API at no extra charge.
- Rackspace for providing server infrastructure.
- Julie Markham and Nicholas Hopper, with whom the author collaborated with on Shy, the mobile application that inspired this project.
- Lilian Blot, for his time and effort spent supervising and providing guidance for this project.

1 Introduction

1.1 Background of this project

The inspiration for this project originates from a project the author undertook in September 2014 whilst attending Yacht Hack, a week-long hackathon on a Yacht in Croatia[1]. The author co-created a project called Shy to prototype a mobile application that would facilitate the immediate answering of questions that fall within certain categories using a knowledge base, simple machine learning and profiling of users based on their usage history.

One of the initial goals of Shy was to bring knowledge via m-learning[2] as opposed to e-learning[3]. This is because m-learning systems are available to a much larger demographic of people, because the hardware (mobile devices, commonly phones) is becoming ever cheaper and more accessible.

A semi-functional prototype was completed for iOS, as seen in Figure 1.1. The front-end was complete; however in the backend question-answer matching was evaluated without knowledge of previous material the user had viewed. This meant that explicitly targeted answers for a question and suggestions of questions that a given user profile might be interested in could not be made. Up until this point, the service was restricted to working on iOS smart devices only, with poor quality question-answer matching. In the initial project, the knowledge base was built for questions on personal health, sex, relationships and family.

1.2 Motivation for this project

Access to knowledge is a critical part of modern life. It's also a Human Right, under Article 27 of the Universal Declaration of Human Rights [**?**]. As members of a developed country, we have the facility to retrieve information and instantly

[1] http://toughhackers.com/yacht-hack-2014/

[2] M-Learning: learning using tools available on a mobile device

[3] E-Learning: learning conducted online, usually with the help of a computer

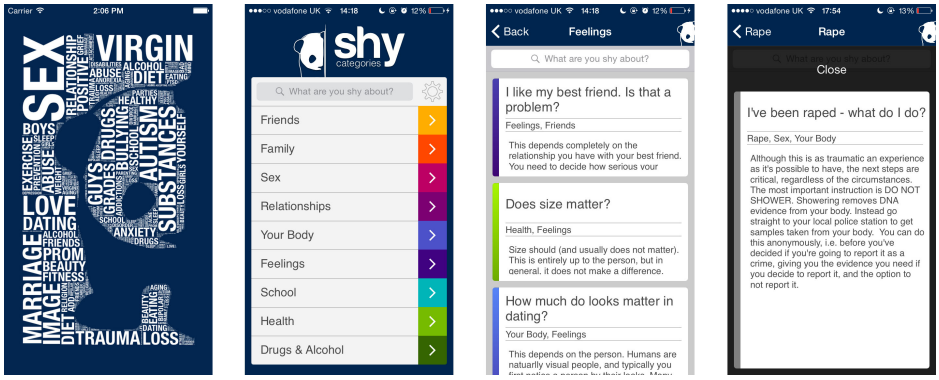

Figure 1.1: Screenshots of Shy iOS App

communicate with each other using our smart phones, and commonly do it up to 200 times a day [**?**] which is a facility that hundreds of millions of individuals in developing countries live without. This project aims to investigate the technical and ethical challenges behind this and to create a technology to give more people access to knowledge, and thus their human rights.

The project involves the use of a number of systems, including machine translation, an SMS input/output system and a system to build a profile on a user in order to facilitate high-quality targeted question-answer matching. The system will also feature machine learning so that as time progresses, the quality of the answers it returns increases. The use of these systems in one application along with complex ethics and privacy issues create an interesting project that draws together many technologies and discussions in a way that can be used as a basis for further research in the future.

1.3 Aims of this project

The first goal of this project is to assess and address the ethical issues that arise from the software that will be created. These include the responsibility that the software and its developer has because of its position of trust to provide accurate information and to protect the privacy of its users by storing only necessary information, among others. This project will do this by researching ethical issues relating to the technologies that the project will use, for example machine translation and user information storage. This will then be used to feed the design process of the software and to specify the expected use-case of it. Finally, the

resulting software will be evaluated through experimentation, with volunteers asking the software a set of questions within a topic area, in a non-English language, and evaluate the relevance of answers returned.

Initially, the goal for this dissertation was to focus on questions relating to personal health, sex, relationships and family, which was as a direct result of the focus of 'Shy'. Although a health and diseases database was located in the early stages of the project (Diseases Database, www.diseasesdatabase.com), it was targeted at professional uses for the data and came with a significant licence fee, as well as complex documentation that the provider customised specifically for each use-case. An attempt was made to acquire a free academic research licence via email but this was unsuccessful. Additionally, as the project progressed in the early stages it became clear that the author's original two-page document discussing ethical considerations with the project did not cover all possible situations. Indeed, further significant ethical issues were still been discovered as far as eight weeks into the project, which would have required a complex ethics panel review, critically delaying this project. One major issues was that any application that dispenses health related information that a user might act upon in a life changing manner has to always provide correct information in response to a query, since an incorrect response could lead to a user taking an action which leads to them harming themselves. No information source containing knowledge of a high enough quality with a licence that allowed it to be used in this project could be found. Another issue was that privacy, which is critical to a project in the area of personal health, couldn't be guaranteed since information would be transmitted in an unencrypted form via SMS.

As a result of the above issues with locating a reputable knowledge base and the complex ethical issues, a decision was made in early December 2014 to switch the target topic for questions to Geography and to use Wikipedia as a datasource, since it is openly available, well populated and multiple methods exist for extracting information from it. In future work, it should be possible to extend this project to work with medical questions, if a high quality data-source is located and a solution to the ethical issues is found. A full discussion of possible future work is included in section 9.

1.4 Structure of this report

This report starts with a literature review in chapter 2 where existing software, tools and services of a similar type to those that will be used in this project are

presented, along with research into the ethical issues of the creation and use of the software under development. Comparisons between different software development life-cycles are also described and discussed.

Chapter 3 describes the method that will be taken to develop the software and service. This covers the software development life-cycle that will be followed, and sets out a plan for when development will take place. Requirements will also be identified, described and categorised as either functional or non-functional. A method of evaluating the success of the software will also be discussed and chosen.

The design of the software, driven from information collected from the requirements, will be set out in Chapter 4. The individual components that make up the technology will be described and discussed, followed by a set of annotated diagrams showing how these modules will interface with each other to build a complete system. Finally, this section will include a description of the platform, language and other tools that will be used.

Chapter 5 contains details about the implementation of the software. This includes information on the tools that will be used to create components such as the SMS interface and question-answer matching system.

Results from the evaluation of the system will be presented in Chapter 6. A discussion of these results and their implication on this project is then presented to the reader in Chapter 7.

Chapter 8 contains an evaluation of the software produced and the process taken to complete it. Comparisons will be made to the success criteria, set out for the project in section 3.5.

Thoughts on potential future research that builds on the work produced in this project are then displayed in chapter 9.

Finally, the main achievements of this project are concluded in chapter 10 with closing thoughts.

1.5 Assumptions Within This Project

This project will make the assumption that users of this service have basic literacy skills in a language supported by the project. Although this is not a correct assumption world-wide, expanding the remit of this project to include a 'graphical' user interface that works over SMS would scale this project outside of the practical

Lilian: I no longer have language support, and you said to keep research in about it. Should I remove it from introduction / abstract etc?

limitations of this project.

The main use-case for this project is going to be parts of the world with limited internet access. Although these exist all over the world, the African continent is typical of this problem so I will focus my research in demonstrating a need for this tool there.

extend this

2 Literature Review

The point of this is to show that I 'own' the material and subject (competence). Show different points of view, choose a side, express a view on which I prefer and if I agree, critique. Have objective goals.

A significant amount of work has previously been undertaken in the area of making knowledge quickly accessible to us in the form of questions and answers. To take one example, imagine that an individual needs to find the population of London as quickly as possible. The initial step would be to search for information. In the connected world, this is easy: a simple Google search returns the answers without even having to click through to any resources, as shown in Figure 2.1.

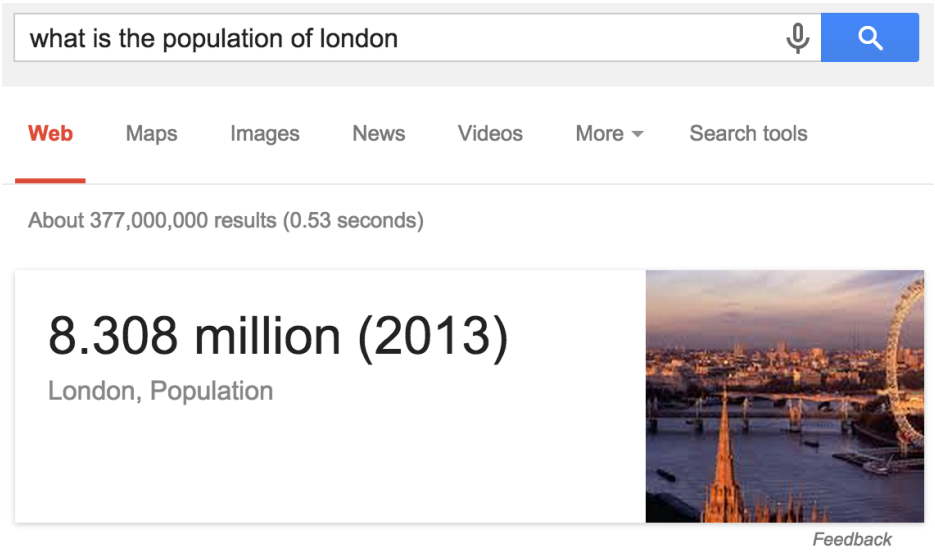

Figure 2.1: Google Now information box. Screenshot taken on 3rd February 2015.

This is easy to do in the developed world where up to 75% of the population are connected to the internet [**?**], with the population of elderly people been largely

responsible for the remaining 25% [**?**]. This contrasts strongly with the situation in the African continent, where in 2013, internet usage had only reached 16% [**?**], a figure largely inflated by South Africa where 5% of the African population generate 2/3 of internet traffic from the African Continent [**?**].

2.1 Previous work using SMS

Although the above suggests that the African continent is majoritively disconnected, this is not the case. Because of restrictions in the electricity available, the ways in which a non-smart phone can be used in Africa have far surpassed those in the developed world [**?**], in part due to the longer battery life of a non-smart mobile phone compared to a smart mobile phone. Interesting examples of SMS use in Africa include an automated service which sends an SMS to HIV/AIDS sufferers to remind them to take their medication, and a system for farmers that gives them the current market prices for goods, saving them from making regular long-distance trips to the market or relying on out-of-date information from a weekly radio broadcast [**?**].

Interactive systems have also been developed to operate over SMS. One successful example is mobile money platforms. These allow for users from any background to pay and be paid for goods and to transfer money across long distances at negligible cost [**?**]. One of the most highly adopted services is M-Pesa, which in Kenya alone was responsible for £5.7 Billion in transfers in 2012 [1]. M-Pesa gives users a balance linked to a national ID number from which they can pay for goods by sending an SMS with a cashier (recipient) number or pay outstanding bills in a similar way. Non-subscribers can also use the system by depositing money with a M-Pesa cashier in exchange for an access code, which can be sent via SMS to a contact who can subsequently redeem it with their local cashier [**?**].

This difference in standard use of mobile phones is demonstrated in the International Telecommunications Union's 2013 report [**?**], which shows that in Europe, for 790 million mobile subscriptions, 53% of subscribers have mobile internet access (422 million), compared to 17% in Africa (93 million have mobile broadband, out of 545 mobile subscribers). This is due to the prohibitively high cost of accessing data services, regardless of the hardware that the user has. In 2012 in Europe, 500 MB of data per month for 12 months cost 1.2% of the

[1] Data from Safaricom, M-Pesa operator in Kenya - actual value 817,085,000,000 Kenyan Shillings, converted to GBP on 1st November 2014 at rate of approximately 0.007.

average Gross National Income Per Capita (GNI pc). In Africa, the average price was 30x this, at 36.2% of an individuals GNI pc [**?**].

2.2 Ethical Issues

This project raises a number of ethical issues related to translation accuracy, providing information to people in an ethically acceptable way (with a focus on information accuracy) and maintaining user privacy.

2.2.1 Ethics of Providing Information

One significant issue for this project stems from providing information that may affect an individual or lead them to take a harmful action.

In a similar way to that which a teacher has a responsibility to teach accurate information to a pupil due to their position of trust, any service relied upon by a user must equally provide accurate information. The result of not doing this could be providing inaccurate information that leads to a user carrying out an action that causes harm.

this needs to be waaaayyyyy expanded, but I struggled to find information that was useful. There's lots on remote education, and on basic ethics of teaching, but I struggled to find anything specifically relating to ethics of ensuring information you provide is correct.

teachers code of conduct, wikipedia ethics to provide infomration that contributos believe to be correct. doctors and gp's provide to the best of their knowledge

2.2.2 Ethics of Translation

A significant part of this project is represented by the support of multiple languages. It is clear that translation on demand, at scale, needs to be automated by some kind of machine or algorithmic translation.

This project involves two blocks of translation. These are:

- The translation of the user's input into the language of the system (English)
- The translation of the answer to a user's question from the system language to their local language.

The latter of these raises some issues that do not apply to the former. To understand

these issues it is first necessary to understand the two categories of machine translation.

Rule-based machine translation effectively treats human language in a similar style to programming languages. Formal grammars and lexicons are used to represent words that exist in either the source or target language, structures representing the translation of individual words or groups of words. Map structures contain mappings between individual or groups of words to their translated counterparts, sometimes with multiple results (a 1-to-many map), from which rules decide which is selected. Maps and rule sets are created by trained computational linguists [**?**].

More commonly, Statistical Machine Translation (SMT) is used, for example, by Google and Microsoft in their translation products [**? ?**]. SMT learns mappings between strings of words of potentially unequal lengths from pre-existing original texts and their trusted human translations. The accuracy and breadth of language support for translation increases as more source material is analysed by the system, as potentially erroneous or low quality translations can be identified and flagged. SMT is also dependent on the quality of the human translation on the input material [**?**].

A significant issue that is present in statistical machine learning systems comes from the knowledge we assume an individual has and derives from a word; in this project, this affects the translation of the answers returned to a user. In human translation this is solved by the translator's knowledge of the difference in material culture, allowing them to append necessary information to the resulting translation that the recipient might find useful. In statistical machine learning, cultural awareness of material knowledge is a separate problem on its own. **formal reference to this, probably again from Kenny, but should be able to find better**. To take one example from Melby (2006):

> "when translating a French menu, a human translator might stop to think that an English speaker in France would appreciate being told that a steak tartare is served entirely raw, even if this information is not contained in the original text (because French people might be assumed to know this already). Such a translator would be aware of differences in material culture, and would be able to empathise with the English speaker who might choose to avoid the dish, given more information" [**? ?**]

This issue is a result of the translation engine not being capable of taking and using a complete representation of the expected cultural differences between those

who speak the input language and those who speak the output language [**?**]. This does not apply to the first block of translation within the application where a user's input is translated into the language of the system, since the application will only need to parse text to identify geographical entities and the property been queried, making context unnecessary.

below should really be expanded and be in the discussion. Explain the decision to keep translated answers in the database, which are pre-approved, to ensure no cultural misunderstandings.

The potential uses of this system include distributing information that may be used by individuals making important decisions, regardless of whether that is within this project or in further work based off of this project, as cultural confusion such as this could have potentially catastrophic effects.

2.2.3 User Privacy and Data Protection

This project uses a simple Artificial Intelligence (AI) to keep a record of information that has already been returned to a user, to attempt to save sending repetitive information (saving the cost of additional SMS). This raises another ethical issue though of privacy. The material that a user has researched is likely to be sensitive; such is the nature of health related questions. This means that the information on a user has to be kept securely, in a way that an individual user can not be identified.

Lilian: Need to re-write this for new question subject, or can leave as a mark of the old subject topic?

One way of doing this is using a technology called hashing. Hashing is the technique of taking data as an input and generating an output value (called a 'hash') that is unique to an input. Hash functions are ideally one way functions where given a piece of input data X, the output hash will always be a constant value Y, whenever and however the hashing algorithm is executed. This allows a user's identifying information such as their phone number to be obscured, in such a way that the mobile number can not be recovered from the database. When a question is received, the phone number (represented by variable X in the above example) can be hashed and the data for this number looked up from the database without the database application ever being aware of the phone number of the user. In the example in Figure 2.2, a phone number (X) is used as input to an imaginary hashing function and the output hash (Y) is shown on the right hand side. As just described, this number can then be used as the key for the user information database, shown in Table 2.1 to retrieve the information needed for the AI.

reference the above from security engineering ross anderson, chapter 10

userId	userData
8f64B2	{âĂŸpreviousQuestionsâĂŹ:[13,301,170,577]}
9ac5e3	{âĂŸpreviousQuestionsâĂŹ:[441,56]}
f9dd7e	{âĂŸpreviousQuestionsâĂŹ:[301,623,89,280,364,621,209]}

Table 2.1: Example of software database: hashed phone numbers paired with a list of previous returned answer IDs.

and 21?

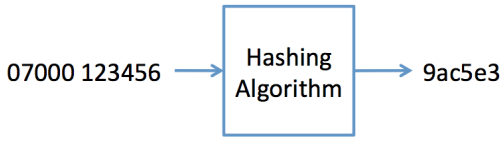

Figure 2.2: Hashing a phone number using an arbitrary hashing algorithm

In practise it is possible for the output of a hashing algorithm not to be unique to a particular input. This is known as the collision resistance property of a hashing algorithm and formally is the property of a hashing algorithm that determines the probability of the same output hash for two distinct random inputs [**?**]. It occurs because the range of possible input values is much larger than the range of output hashes, and because all valid inputs must map to an output there is repetition in outputs [**?**]. This property of hashing algorithms can be a security risk for some use-cases, where hashes are used to verify that data has not changed since been hashed. In this case, a third-party may be attempting to generate alternative, malicious data with the same hash as some pre-existing data, to swap them undetected [**?**]. In the use-case of this project there is no security risk since hashes are only used to anonymise phone numbers; however a high collision resistance is necessary to ensure that two users of the system with distinct phone numbers don't get mapped to the same row in the database, resulting in poor quality question-answer matching or in the extreme worst case, privacy issues originating from returning to one user answers that are targeted to another.

Irreversibility is a necessary property of the hashing algorithm chosen for this project. This ensures that the key used in the database cannot be used to retrieve the phone number of the user, thus protecting their identity. Commonly used hashing algorithm families, such as the MD family (e.g. MD5) and SHA family

are all designed to be one-way [**?**]

Lilian: Is the wording of Bruce Schneier paragraph ok? Should I perhaps remove the last sentence?

It should be noted that Bruce Schneier is a fellow at Harvard's Berkman center for internet and society. He has been posting regularly for his newsletter and then blog since 1998 (over 16 years) and has published material related to this field throughout this time. He has been involved in the creation of other cryptographic algorithms, such as the Skein hashing algorithm and blowfish block cipher, so may have a vested interest in discrediting other algorithms such as MD5 and SHA. However, the essay cited above is written as a result of the Crypto 2004 Conference in California, where the algorithm that Schneier suggests should be used at the end is not one that he has involvement in. For these reasons, his material should be treated as reliable.

Although hashing algorithms are computationally one-way, they can suffer from another type of vulnerability affecting the security of the original input data. When hashes are used for short strings of information (passwords, for example), a simple way to try and retrieve the original data is to compute a dictionary of the hashes of lots of common passwords, from which matches can be found. This idea has been extended to produce Precomputed Hash Chains and subsequently Rainbow Tables. These are highly efficient data structures that consist of chains of processed input messages and their hashes with a reduce function to reach the next item in the chain, allowing original input data to be looked-up from its hash. Both Precomputed Hash Chains and Rainbow Tables tend to be Terabytes in size, and as such they only exist in a significant form for the most commonly used hash functions (including MD5 and SHA) [**?**]. Within this project, this raises privacy implications, as hashes within the database would be 'convertible' to phone numbers.

In password authentication systems, this can be solved by the use of a technique called Salting [**?**]. Salting is where a 'salt' - a piece of additional unique data - is added to each piece of input data before hashing, and then this salt is prepended to the output hash. This nullifies the use of Precomputed Hash Chains and subsequently Rainbow Tables [**?**]. When checking if a plaintext password is equal to the hashed and salted representation, the salt from the stored hash is added to the plaintext password, the hashing algorithm applied and then the output compared to the stored obscured password. However, in this application this would prevent efficient lookup of users from phone numbers in the application, as the application would have to compare an input phone number against every user object in the database, which is computationally expensive and slow since a salted hash has to be computed for potentially every user record [**?**].

As such, it has been decided that the best way of protecting a user's phone

number in the event of a database hack without significantly degrading performance is to salt phone numbers with a constant, large and complex salt. This means that a rainbow table would have to be specifically computed specifically for this application and would be extremely large in size.

2.3 Software Design Life Cycles

There are many software development life cycles that a developer or company has available to choose from. Four common ones include the Waterfall Model [**?**], Iterative Model [**?**][2], Spiral Model [**?**] and the Agile Development Model [**?**][3]. In order to choose which life cycle suits this project best, research into all four has been carried out.

Lilian: remove these two footnotes? Not always necessary to reference the original

2.3.1 The Waterfall Model

The waterfall model consists of 5 phases of work, where each leads directly into the next. It is a plan driven development model, which means that all process activities such as the implementation of the project must be planned and scheduled before work on them begins [**?**]. The five layers are shown in Figure 2.3 and described below:

1. **Requirements**. In this stage, both user and system requirements for the project are determined after examining the business goals for the project. These form the system specification.
2. **Design**. Here, a plan for the software development is created. The overall architecture for the software is created, the external libraries to be used are chosen and the interaction between different modules is modelled.
3. **Implementation**. In this stage, the software is implemented to match the design and architecture set out in the Design phase. Unit tests are written to ensure that each module matches its specification.

[2] Iterative development came together from many concepts dating back to 1957. There was no single paper that initially presented the entire concept, but this paper gives a summary of how the various concepts came together into the current Iterative Development model.

[3] The Agile Manifesto brought together a group of Agile Methods which had been in use since the mid-1990's into a single concept, that of Agile Development.

4. **Testing**. In this section the system is tested as a whole to check for any bugs that might not be picked up when testing individual modules in isolation. The software is then delivered.
5. **Maintenance**. Finally, the software is deployed and maintained. This includes making changes to the requirements and subsequently the software implementation as the business goals change. [**?**]

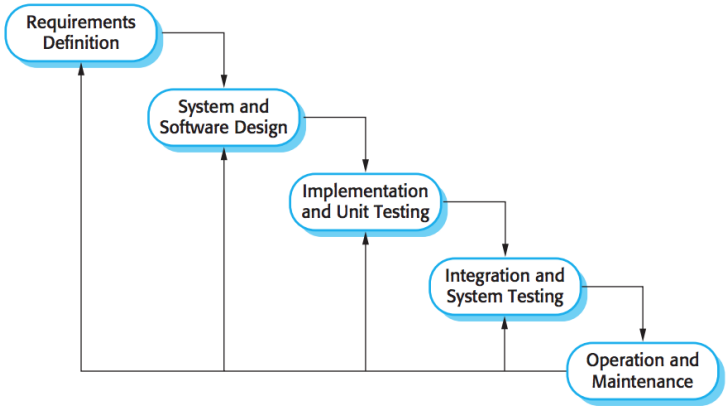

Figure 2.3: Waterfall Development Model [**?**]

A key advantage of the Waterfall Model is its simplicity: it is easy to understand and to use, saving time at the start of the project that would otherwise have been spent on learning and preparing to use a more complex model. The main disadvantage of the waterfall model is that once development has moved from one stage to another it cannot go back because the model is fixed and progresses in one direction only, until the maintenance step.

2.3.2 The Iterative Model

Iterative development is the practise of splitting the overall development of a project into multiple independent and distinct blocks. A block contains a sequence of tasks which are essentially a mini life cycle, such that each iteration can really be thought of as its own project. Each iteration tends to focus on a specific set of features such that when the iteration is complete the overall project is stable and the modules developed to date can be tested as a whole system [**?**].

There are a multiple variations of the iterative development method. Two

of these are Timeboxed Iterative Development where all iterations have a fixed duration, and Client-driven Iterative Development where each iteration contains the features that the client considers have the highest business value [**?**].

A common variation is based on the waterfall model as shown in Figure 2.4. The first three tasks in the cycle represent the first four stages in the waterfall model (Requirements, Design, Implementation and Testing), at which point there is the option to either evaluate progress so far and start another iteration, or to deploy the software and end development. One significant difference between this and the waterfall model is that maintenance is not represented.

A key advantage of the Iterative Model is that it allows for the user to revise their requirements as they progress, making it dynamic and flexible which is ideal for small development teams. This is also its disadvantage, in that the lack of fixed structure can be a weakness unless the project is planned well.

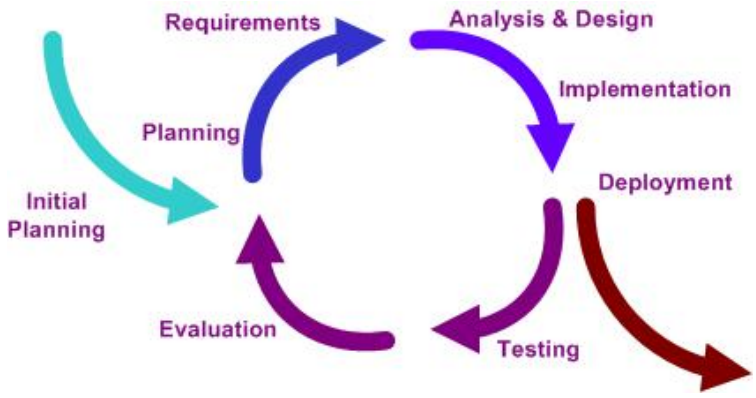

Figure 2.4: Iterative Development Method
source: https://commons.wikimedia.org/wiki/File:Iterative_development_model_V2.jpg

2.3.3 Boehm's Spiral Model

The Spiral Model [**?**] was originally developed as a result of the adjustments that were commonly made to the waterfall model in large government projects. The Spiral Model is a general model and can be used as a generator for other more specific models, given the parameters for a project such as its risks [**?**]. Models such as the waterfall model are specialisations of the spiral model [**?**].

In the Spiral Model each loop represents a stage of the software development, from planning on the inside loop to system testing and verification on the outside,

as shown in Figure 2.5. The cyclic nature of the model reflects the evolution of development of a software engineering project: the displacement from the origin shows an increase in the definition of the software, progress with the implementation and a decrease in the overall risk [**?**]. With each spiral cycle in the model, stakeholder objectives are reviewed, risks identified and stakeholder approval and commitment sought before commencing the cycle [**?**].

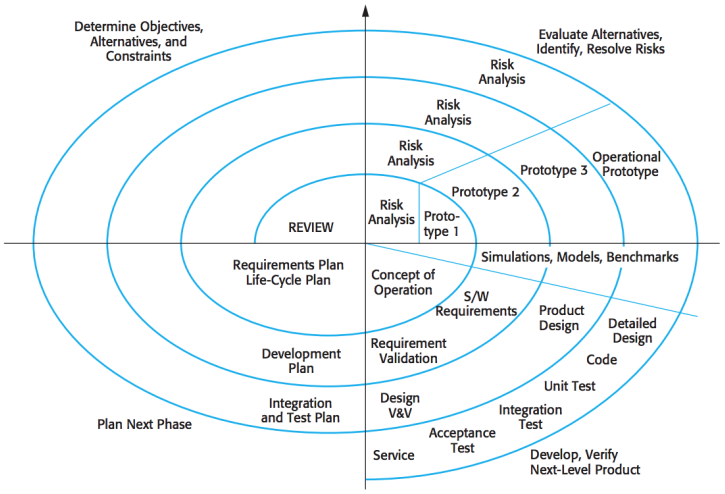

Figure 2.5: Boehm's Spiral Model [**?**]

A key advantage of the Spiral Model is the level of detail it models. The stage of a sub-task within the entire development process is represented by the distance from the origin and each loop is split into small, well described sections. The main disadvantage is that the planning and use of the model is largely driven by the results of complex risk analysis on the sub-tasks [**?**]. Although this makes it favourable to large organisations, it has the opposite effect for smaller projects.

2.3.4 Agile Development

Agile Development essentially represents a group of development techniques targeted towards either the development of a small to medium sized product for sale or the development of a custom system within an organisation, where the customer has a high degree of involvement, for example 'Extreme Programming' [**?**].

Agile development is highly targeted towards teams with components such as

Pair Programming, Collective Ownership, regular meetings and employee health (avoiding overtime) playing an integral part in the model [**?**].

2.3.5 Development Life Cycle Summary

Iterative development has been chosen as the model that this project will use, based on the following factors:

1. Boehm's Spiral Model and Agile Development are both targeted towards development teams. A significant part of these models relates to team interaction, so they are not appropriate to this project.
2. The Waterfall Model is rigid and inflexible. It does not allow for the changes to the requirements and implementation that might be proposed after evaluating the quality of the first implementation.
3. The Iterative development model can be implemented effectively by a single developer, and supports the adjustment of requirements and implementation in each iteration of the cycle.

3 Method and Requirements

3.1 Plan for Software Development

The Iterative Development Model was selected in Section 2.3.5. Therefore, the development of this service will follow the principles set forth in this process.

3.2 Requirements

3.2.1 Use Cases / User Goals

Two personas were created to represent individuals who may have a need for the work resulting from this project, which can be seen in Appendix A. These are Nanjala, who is from Uganda, and Cedric, who is from Switzerland, and it is from these that user goals are extracted. These describe what a general user aims to achieve with this service.

1. Nanjala wants to be able to read general information on any given topic.
2. Cedric wants to be able to retrieve facts from direct questions.
3. Both Nanjala and Cedric want to be able to achieve their goals entirely via SMS.
4. Cedric needs a simple and efficient user interface that allows him to retrieve facts quickly.
5. Cedric wants to be able to feedback into the system when he finds an answer particularly useful or unhelpful.
6. Nanjala wants to be able to show the service to her friends without explaining how to use it every time.
7. Cedric wants the system to protect his privacy, obscuring his identity.
8. Nanjala wants to be able to ask questions as she would ask a friend without

having to learn a syntax.

3.2.2 User Requirements

User goals are distilled into formal user requirements. Each user requirement corresponds directly to the correspondingly numbered user goal.

1. Users must be able to request general or descriptive information on a geographical entity.
2. Users must be able to ask questions requesting a fact or statistic about an entity and get the answer or number they are looking for.
3. Users must be able to interact with the system over SMS.
4. Users must be able to make their queries and get responses quickly.
5. Users must be able to feedback to the system when an answer is of noticeably low or high quality so the system can learn and improve.
6. Users must be able to learn how to use the system themselves and this learning process should be quick.
7. Users must be comfortable that their privacy is been protected whilst using the service.
8. Users must be able to interact with the system using natural language.

3.2.3 System Requirements - Functional

System requirements are derived from the capabilities that the software needs to have to be able to support the user requirements. To support the user requirements in section 3.2.2, the application system must fulfil the following system requirements.

1. The system will be capable of processing queries in natural language form.
2. The system will receive queries via SMS, process them and then respond via SMS.
3. The system shall use a datasource that contains both facts and descriptive text on geographical entities to answer queries.
4. The system must be able to parse queries to identify the entity that is being

asked about and the specific parameter of that entity that should be returned as the answer to the user's query.

5. As well as giving direct facts and statistics, the system should be able to give general descriptive information on an entity.
6. The system must be able to match words of similar meaning to the parameters in the data source. For example, the word 'long' must be matched to the parameter 'length'.
7. The system must learn from user feedback when answers that are returned are of high or low quality an apply this knowledge to future queries.
8. The system must be self-explanatory, with the facility to offer instructions to a user.
9. The system must use user state so that it can accept queries that are dependent on past questions, for example a query asking for more information or feedback on the quality of a past question.
10. The system must be easy to implement and flexible, to accommodate future projects that make use of adaptations of this project.
11. The system must obscure a user's phone number in the database.

3.2.4 System Requirements - Non-Functional

Non-functional requirements are listed in table 3.1 and are accompanied by a Fit Criterion, which will be used to identify whether a requirement has been met successfully.

3.3 Data Sources

A number of possible datasources were identified from which data could be drawn from for this project. These were all based off of Wikipedia (a large, linked source of knowledge), and include the Wikipedia API and various representations of the DBPedia Ontology. In this section, both will be discussed and a decision will be made with regard to which ones will be used.

Non-Functional Requirement	Fit Criterion
The system behind the SMS gateway must respond in a timely manner.	From an SMS been received, a response must be sent within 20 seconds. SMS transmission times on the cellular network and gateway are out of the control of this project, so are not accounted for.
The answers returned must be of high quality.	In testing, the system returns the correct information for at least 90% of queries initially (before learning starts) if an answer is returned at all.
Answers that are consistently rated poorly are are less likely to be returned.	In testing, after receiving a small number of negative ratings for an answer, a different answer is returned.
Instructions to use the service should be clear and concise, with no ambiguity.	In testing, having explained the purpose of the system, a user should be able to use the system with just the help information, without needing to ask for assistance.
The system should accept and be able to process questions in natural human language as opposed to a delimited message.	In testing, a user should be able to use the system for general questions without having had a syntax explained to them.

Table 3.1: List of non-functional requirements and their Fit Criteria.

3.3.1 Wikipedia API

The Wikipedia API is a standard MediaWiki API and allows applications to query a page or resource for various parameters from a page, including the page contents, infobox parameters, citation list and links to other articles in various flat or multi-layered data structures [**?**]. Interactions are made via a HTTP GET request to a URL containing the parameters, and responses can be received in JSON. This makes the API easy to use and widely compatible, especially since JSON can be converted to a dictionary in most languages.

3.3.2 DBPedia Ontology

The DBPedia Ontology is the result of a project that aims to build an OWL ontology representing all of the knowledge within Wikipedia. The ontology has been manually created from the most commonly used infoboxes on Wikipedia and covers 685 classes with 2,795 properties, providing structured information on 4.2 million places, people, work, species and organisations [**?**]. This makes it especially good for use in applications since the data is both curated and well structured.

There are two ways of querying the DBPedia Ontology. One is using the standard SPARQL ontology query software to make a query form the ontology structure; however this is complex as queries must be structured correctly and it is necessary to know the class structure of the entities being queried [**?**].

In the event that the class structure is not known, queries can be made by querying the top-level representation of an entity in the ontology, which presents data in a similar format to that of the Wikipedia API, albeit in a 'cleaner' form (with human readable keys) as a result of the data been processed by hand. Although the keys are of a more human-readable format, some material is abstracted within the ontology and so is inaccessible with a basic query. An example of this is page for 'London' (accessible at: `http://dbpedia.org/page/London`), which does not have a parameter 'Mayor' with the value 'Boris Johnson'. Instead it refers to a number of 'Leader Titles' and 'Leader Names', which include a reference to Boris Johnson

3.3.3 Choosing the Source and Interface for this Project

This project has a limited scope and time period available to it, and creating a system capable of identifying the relevant classes and then constructing SPARQL queries for looking up an abstract property is an example of an approach that is out of scope, as it involves natural language processing and semantic analysis of the names of the ontological classes, not just the query being made. As such, this project will make use of a combination of the data from the Wikipedia API and DBPedia top-level entity representations, collecting keys from both and working with the key with the highest semantic relatedness match to identify the data-value that most likely is the answer to a user's question. The DBPedia Ontology also provides access to the abstract at the same level as the infobox, so general descriptions of entities will be sourced from there.

3.4 Processing Natural Language Queries

Because of the nature of its audience, the system needs to be able to process input in natural language form rather than enforcing a formal grammar on users as described in User Requirement 8 and System Requirement 1. This involves constructing a natural language processing system to process queries and identify their meaning. In this project, queries are of a simple form: if a user is asking a question they are requesting a property of information about a given entity. An assumption is therefore made that each query consists of: a token representing the geographical entity been queried; a token representing the property been requested of that geographical entity; and some noise in the form of stop words.

Lilian: Ok to have this quite descriptive definition in the main text following straight on from the last paragraph?

Stop words are words that are frequently used in in language but do not carry any significance themselves [**?**]. It is common practise in Natural Language Processing to to remove these stop words before attempting to process an input [**?**]. An example of a stop word is the word `the`.

It necessary to be able to identify the geographical entity in the query. This will be done using a Named Entity Extraction tool, of which there many available online including Cortical [**?**] and Dandelion [**?**]. These tools extract keywords from a text and in some cases are capable of categorising keywords according to an ontology. For example, Dandelion links entities to the DBPedia Ontology [**?**].

The property about the geographical entity will be established by removing stop words, punctuation and the name of the geographical entity from the string. In the input 'What is the height of the Eiffel Tower?', this would result in the

string 'height'.

The result of this simple natural language processing should be that a sentence such as 'What is the Height of the Eiffel Tower?' is reduced to a structure indicating that the geographical entity is the 'Eiffel Tower' and the property is the 'height'.

3.5 Evaluating Success - Experimental evaluation of the system

Success will be evaluated by establishing if the project fulfils its original requirements, based on the scenarios. Many user requirements such as protecting the privacy of a user will be demonstrated through the software developed, whilst some that involve user interactions must be tested with users. As such, user testing will be undertaken to evaluate three specific properties of the software through a trial usage and subsequent questionnaire: the usefulness and quality of the service, its usability, and the quality of the machine learning system.

Prior to running the experimental evaluation, a dry run will be undertaken with two participants. No data will be stored or collected - the purpose of the dry run is purely to establish early on any issues with the running of the experiment that would cause a problem during the real experiment.

3.5.1 Client software for the experiment

The goal of this project has been to construct a piece of software that interfaces over SMS, as described in the project title. However there are practical limitations to running the experiment via SMS. During testing of the various services it was observed that HTTP-SMS Gateway Services added approximately a 5 second latency to both the sending and receiving of SMS, extending the length of time that a user would need to partake in the experiment to collect the same number of data points as could be collected when working straight over HTTP. In addition, the sending and receiving of SMS is costly and the savings made by not consuming SMS can be used to encourage participants to partake in the experiment.

The interface is also an important consideration for the experimental evaluation - a key goal is to collect as many data points from participants as possible, whilst still maintaining some of the context in which the software will be used - on mobile devices. With regard to input speed, it is assumed that typing speeds are

reference?

slower on mobile keyboards than full-sized physical keyboards, so data points can be collected more rapidly when working on a full-sized physical keyboard. In order to do this, a mobile app will be developed that interfaces with the software, and this will be run on a laptop with full-size keyboard. This gives participants use of a physical keyboard, allowing them to query the system more quickly and therefore making it acceptable to ask them to do a larger set of queries, and maintains the mobile context. poor wording?

3.5.2 Questionnaire Design

The questionnaire will aim to evaluate the usefulness of the system, the usability of the system and the quality of returned answers, and is based off of the IBM "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology" questionnaire [**?**]. Questions were modified to suit the software being used in a situation without internet as opposed to a general workplace environment, and two questions were added: one to investigate the overall quality of answers returned to a user, and another to discover how trustworthy users felt the answers returned by the application were in general.

Each question is formed by statement and a measure of how much the participant agrees with the statement. This is measured by selecting a value on a five-level Likert scale [**?**]. The scale ranges from 1 through 5, where 1 is `strongly disagree` with the statement given, 3 is `neutral` and 5 is `strongly agree`.

A copy of the questionnaire that will be used is included in Appendix D.

3.5.3 Typical experiment plan

At the start of the experiment, participants will be given both the consent form, included in Appendix B and a document explaining what the software aims to achieve and what the participant should do, included in Appendix C. The consent form sets out exactly what participants will be asked to do, the information that will be collected from them, and reminds them that they can withdraw at any time. The instruction sheet sets out some topic areas for the participant to research using the software, with a varying level of freedom within that topic. For example, they may be asked specifically to `identify the height of the Eiffel Tower`, or to identify a property of an entity of their choice such as `identify the population of a country of`

`your choice`. Participants may also be asked to locate general information about a geographical entity. Constraints are placed on the area in which participants can query to limit the range of keywords used, thus focussing the machine learning onto a smaller group of words to allow a demonstratable improvement to be shown in the short space of this study.

After the experiment, participants will be asked to complete the questionnaire. Any additional comments arising during the post-experiment debriefing will also be noted beneath the questionnaire.

3.5.4 Assessing success quantitively

As stated in section 3.5, the experiment aims to evaluate three goals: the usefulness and quality of the service, its usability and the quality of the machine learning system. In the questionnaire, questions 1 through 6, 13 and 14 investigate the usefulness and quality of the system, whilst questions 7 through 12 investigate the usability of the system. Bearing in mind that this project is heavily constrained with regard to time due to constraints on the nature of it been a university project, it is not expected that the software will be as good as is possible. As such, a positive mean rating for usefulness and quality, and usability, would be deemed a success.

The machine leaning will be tested by inspecting the change in usefulness and quality rankings of the system made by users in the questionnaire as more participants use the system and provide ratings. The implementation of the machine learning system will be considered a success if a statistically significant positive trend is found in the usefulness and quality ratings when the ratings are sorted by the number of proceeding participants.

4 Design

In this section, the design of the system will be described, including information on: the structure of the server application and database; interactions with third-party services; systems used to protect user-identity; and the language, platform, tools and libraries used for development.

4.1 System Design

In this section, the high-level design of the system software and its interactions with third-party services is described. The system design was driven by requirements listed in sections 3.2.2, 3.2.3 and 3.2.4, which were generalised to:

- The system must be modular with points to add additional plugins, for example for additional data sources.
- The system must be easy to maintain, with distinct boundaries between functional components and behaviour.
- The database must be local to decrease the risk of exposing data.
- The system must have clear interfaces for interacting with the external world.

Based majoritively from these requirements, the formal aforementioned requirements and the application of an informal gap-analysis between each iteration of the architecture, the system architecture shown in Figure 4.1 was created. This shows the structure of the system, from the level of hardware and third-party services through to individual software modules.

4.1.1 Micro-system for Protecting User Identity

In section 2.2.3, the need to protect the privacy of a user and the general technique that will be used (involving irreversible hashing) are described. The design of this

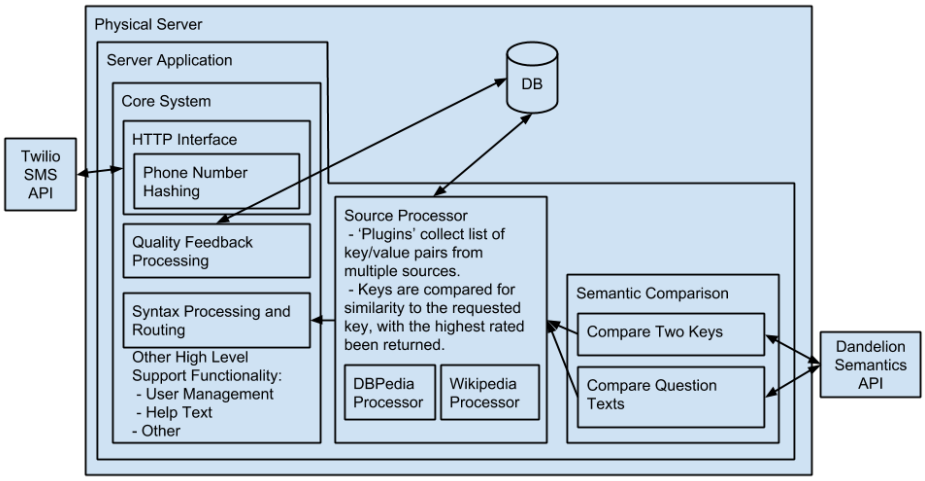

Figure 4.1: System Architecture

system will now be discussed.

The main gateway into the system will be an HTTP call to a Python function called `sms()`, which takes the phone number of the user and the question being asked as its parameters. This function represents a core part of the privacy protection system, by generating the hashed phone number and only using that when calling the function that computes the answer. The user's unobscured phone number is only subsequently used when that function call has returned and it's time to send a reply SMS to the user. After that message has been sent the HTTP connection will close and the original number will be lost when the memory is released. As the computation of an answer only requires a general constant representation of user identity, not necessarily the user's real phone number, the specific value of the users phone number is not required within the computation of the answer. As such, a hash is passed into the answer-computing function. This hash will be used within the database to store a user's history and profile. This means that only minimal user information (described in the physical data model in section 4.2.3) can be identified from the database, not the actual identity of a user.

4.2 Language, Platform and Tools and Libraries

4.2.1 Language

The language Python has been selected to be the main language in project for four reasons:

1. Python is universally compatible and simple to execute. Once the server application is written, it will be simple to migrate it between machines and adjust the application's configuration or environment.
2. Python is commonly used and has significant community support in the form of libraries for all the tools that will be used in this project, for example libraries for querying common databases and for sending text messages. This is useful as it will save time implementing common functionality like database calls.
3. Python is an extremely flexible language in which it is easy to implement applications of this type without syntax issues. This fulfils the system requirement in section (3.2.3, point 10) that the application is easy and quick to implement.
4. The author has significant expertise in the language.

A testing utility will also be created to query the application from a personal computer without using SMS. A simple OS X Cocoa application will be created that queries the HTTP interface with JSON in the expected format and displays an answer. Objective-C and and Mac OS X were chosen as the language and platform for this as the author has experience with Objective-C and uses a Mac. There was no requirement for universal compatibility because the application is only for testing and development purposes.

4.2.2 Platform

The development platform of the utility will be Linux. This is for three reasons:

1. Linux manages Python and its libraries in a simple way without complex installers with tools like the Python Package Index (a library repository) [**?**].
2. Common libraries that will be used for creating a HTTP server, creating a REST API, making database queries and sending SMSs tend to be written

Lilian: should I explain what this is somewhere?

primarily on the Linux platform.

3. The author has familiarity with Linux and has access to significant Linux infrastructure under a grant from Rackspace.

As previously stated, the interface to the application will be over the SMS platform.

4.2.3 Database

There are three tables of data that need to be stored. These are the tables for: user profiles, to allow the application to handle queries that make use of 'state' with users; a list of keyword matchings between the requested property in the question asked and the property returned from the data source with quality ratings from user feedback, which the application uses to learn from and improve its answers; and a record of all asked questions, their answers and the ratings provided, to allow for questions to be answered based on highly rated previous answers.

In the following section, the conceptual model is presented and then this is used to choose a database technology. Once the database technology has been chosen a logical data model and a physical database design are given.

Conceptual Data Model

There are four entities in the conceptual data model for this project. These are:

- **Users:** An entity that represents a user. The user is identified via an obscured representation of their phone number.
- **Last Question Asked by a User:** An entity containing the parameters of the last question. This allows the system to handle stateful queries that are based upon a previous query.
- **Keyword Pair Quality Ratings:** An entity that contains the property requested by a user, the property returned to that user, and the number of times a matching was ranked as being either one, two, three, four or five stars.
- **Ranked Previous Answers:** A representation of a previously asked question, the answer text provided, and the rating given to the answer.

These entities are shown in the UML diagram in Figure 4.2, which also shows the entity relationships.

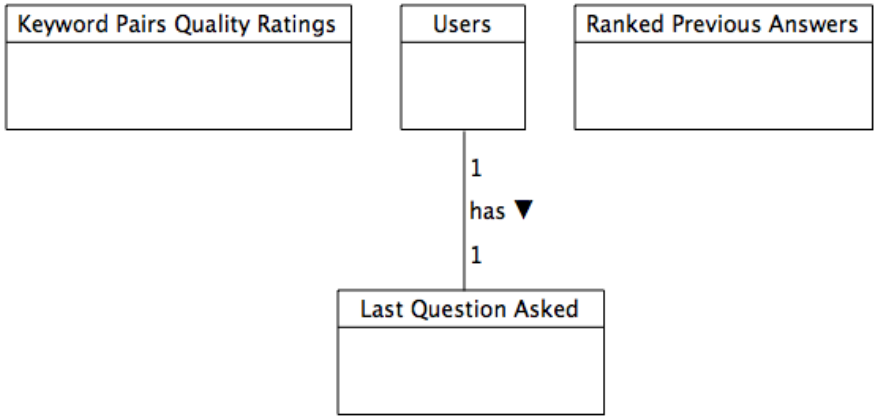

Figure 4.2: Conceptual Model of Data

Database Technology

The database technology requirements for this project are:

1. The selected database must be as simple in implementation and interface with as possible, whilst still supporting the system requirements. This is to allow for quick development and to avoid adding unnecessary complexity.
2. The selected database must allow for storage and retrieval of records with multiple parameters.
3. Support for records that are related to each other is not required, so relational databases are not needed, but not ruled out.
4. The database must have a library for interfacing with it in the language of this project.
5. The database should have built-in support for Sharding and Replication/Mirroring if possible to allow the application to scale. Sharding is the theory of distributing data and hence load across multiple servers [**?**], and replication is where data is stored on multiple machines to provide redundancy [**?**].

There are a number of database technologies available that would fulfil the database requirements of this project. Specifically, these can be split into two categories: relational databases and non-relational databases. Relational databases are used to store highly structured data in a table format [**?**]. Each table uses rows to represent a single record, and columns to represent the variables within that record [**?**]. These tables may contain constraints on the contained data to ensure integrity, such as limiting the range of a variable or enforcing that a variable may not be null [**?**]. Relational databases also encourage the use of table manipulation operations, where multiple tables are used to derive new tables with a processing stage in-between [**?**]. Relational databases also require a schema to be devised to formally structure the data [**?**].

Non-relational databases are specialised towards unstructured data, or where the data may be structured but there is a desire to not enforce a structure [**?**]. Records are stored in collections which are similar to a table in a relational database, and collections then contain documents, which are similar to rows in a relational database [**?**]. Unlike a relational database a collection does not have a set of columns or preset list of variables for each record [**?**]. Instead, each record is a dictionary. Non-relational databases generally support storing relational data (entities that refer to other entities) by nesting documents inside of each other for one-to-one and one-to-many relationships [**?**]. They also support referencing another document via its unique ID (which is assigned to all documents) [**?**]. A downside of this latter technique is that the first object must be retrieved from the database to retrieve the ID of the referenced object before the referenced object can then be fetched in a relational-like query.

The two common databases used for each of these database categories are MySQL databases [**?**] (relational) and MongoDB databases [**?**] (non-relational). In a Python application a MySQL database can be queried with an Object-relational mapping tool (ORM), which maps queries from the syntax of your programming language to a query in the SQL (a query language) and then maps the returned data to objects compatible with your programming language[1]. MongoDB comes with a simple pre-existing client to query the database natively [**?**].

To fulfil the database requirements, the database chosen should be flexible with no fixed schema, querying should be easy in the Python language, and either support for relational queries or nested documents will be required. Additionally, it would be good if the database natively has sharing and replication/mirroring

[1] ORMs also commonly apply optimisations to queries and validate them to ensure they are not malicious.

	MySQL	MongoDB
Fixed Schema	Yes, reducing flexible	No, giving flexibility
Querying Simplicity	Requires complex ORM and awareness of transactions, or construction of SQL queries.	Very simple with included MongoDB client.
Records support nested parameters	No	Yes
Supports relational queries natively	Yes	No
Built in Sharding support	Yes	Yes
Built in Mirroring support	Yes	Yes

Table 4.1: MySQL & SQLAlchemy vs MongoDB

support so that in future work a stable and reliable system can be built. MongoDB meets all of these criteria, with no fixed schema, a simple method for querying the database, support for nesting documents and sharding and mirroring support for future expandability. In contrast, MySQL forces a fixed schema for data and requires a complex ORM for querying, although it does support relational queries natively and has support for sharding and mirroring [**?**]. These properties of the two databases are shown in Table 4.1.

From this comparison, the decision to use MongoDB has been taken because MongoDB has more flexibility than SQL meaning that development will be less time consuming, and writing queries is easier in Python with the built-in MongoDB Library [**?**].

Logical Database Design

Having decided that the database technology will be MongoDB which supports nesting in section 4.2.3, the logical design of the database is able to reduce the number of entities from four in the conceptual model to three. This is because there is a one-to-one relationship between the `User` and `Last Question Asked by a User` entities, so the `Last Question Asked by a User` entity can be nested inside of the `User` entity as it only contains a small number of user-specific fields. This will simplify the implementation, meaning that a smaller number of collections will be needed to contain the entities.

Table/Collection	Parameters
Users	(id, cellNumber, history, (lastQuestionText, lastQuestionRequestedProperty, lastQuestionReturnedProperty))
KeywordPairingRatings	(id, givenProperty, returnedProperty, oneStarRatings, twoStarRatings, threeStarRatings, fourStarRatings, fiveStarRatings)
RankedPreviousAnswers	(id, question, answer, rating)

Table 4.2: Logical Database Design. Underlined fields represent primary keys.

The logical database requirements are listed in Table 4.2 and represented graphically in Figure 4.3.

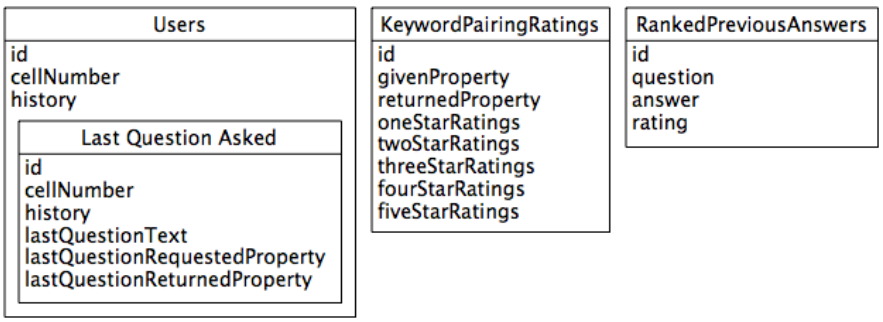

Figure 4.3: Logical Model of Data

Physical Database Design

As previously discussed section **??**, MongoDB separates groups or types of entity into collections (which are similar to a Table in an SQL database) but does not place restrictions on the structure of objects within a collection. The general structure of each document in a collection for this application is shown in Table 4.3. A small change was made between the logical and the physical database designs with the storage of ratings in KeywordPairingRatings. In the logical model these were individual counts of ratings at the five different levels (for example, a

Users	{
	'_id',
	'cellNumber',
	'history',
	'lastQuestion' : {
	'text',
	'givenProperty',
	'returnedProperty'
	}
	}

KeywordPairingRatings	{
	'_id',
	'givenProperty',
	'returnedProperty',
	'ratings'
	}

RankedPreviousAnswers	{
	'_id',
	'question',
	'answer',
	'rating'
	}

Table 4.3: Typical Document in each MongoDB Collection

count of one-star ratings, a count of five-star ratings, etc) however in the physical model this was converted to a list of ratings received (e.g. `[5,5,3,4,5]`). This made the implementation more compact as five different variables did not need to be addressed every time the ratings were read or written to.

4.2.4 Libraries

Database Querying

MongoDB comes with a Python client called PyMongo [**?**], which is developed by the same developers as MongoDB. It is the recommended tool to interact with a MongoDB database [**?**] and has good community support, so it will be used in

this project.

Creating HTTP Server and REST API

The application needs to have an HTTP interface to allow interactions with it via SMS. As a result of the non-functional requirement "The system behind the SMS gateway must respond in a timely manner" in section 3.2.4, the application will need to be able to process multiple HTTP requests concurrently to prevent requests queueing up and responses been significantly delayed.

To create an HTTP Interface, the latest Web Server Gateway Interface (WSGI) specification (version 1.0.1, published as PEP 3333) [**?**] will be followed as it is the industry standard. The WSGI specification describes a general interface between web servers and web applications or frameworks (such as Twilio) in the Python language, thus describing how the web server should be constructed.

In this project, two tools are needed: a WSGI server, and a framework for writing WSGI applications. Flask will be used for the framework as the author has extensive experience using it and it is compliant with the WSGI standard [**?**].

With respect to the WSGI server, many implementations exist such as a simple WSGI server included in Flask [**?**] called Werkzeug [**?**], and the Gevent WSGI server [**?**]. Support for multiple concurrent HTTP connections is not a requirement in the WSGI specification, and so some WSGI servers such as the implementation of Werkzeug in Flask do not support this even though Werkzeug does natively support processing multiple HTTP requests concurrently. It does this by specifically enabling threading and increasing the default maximum number of processes from one [**?**]. The implementation of Werkzeug in Flask has these disabled in the source code because Flask uses thread-local storage (memory that is local to a single thread) and so cannot support concurrent HTTP requests on new threads [**?**].

A library called Gevent will be used to replace Werkzeug in Flask as it does support concurrent HTTP connections [**?**] using using the Greenlet library [**?**], which provides a notion of micro-threading (coroutines). When a new HTTP request is received, Gevent creates a new Greenlet, thus preventing the main thread been blocked and allowing multiple concurrent HTTP requests. The creation of the Greenlet is described in the preceding reference([**?**]).

Lilian: Should I work these footnotes into the main text?

Sending SMS

There are a number of SMS providers with HTTP gateways available to choose from, all of which are similar. For this project, the provider must match the following criteria:

- Forwards received messages to a REST API to enable easy development.
- Supports sending messages via a REST API which also enables easy development.
- Has good coverage of countries so that users from many countries can use the service.
- Has a 'good' reputation. This is because user privacy is still an issue, since data is unencrypted when been sent over SMS, so the provider should be trustworthy.

It is difficult to measure the reputation of a provider accurately, but one possible way of doing this is by looking at the number of significant clients they have.

Two services considered were that matched these criteria. These were Twilio [**?**] and Plivo [**?**]. Both use REST APIs for handling incoming messages and for sending responses. Both Twilio and Plivo have comparable coverage as well: Twilio state that they send and receive SMS with numbers in 198 countries [**?**], and Plivo state that they support 207 countries [**?**][2]. The difference is trivial, as the number of recognised independent states in the world is between 190 and 200 depending on listing; the US Government recognises 195 [**?**].

Lilian: remove footnotes about bias? Also, ok to mention a RAW URL?

With regard to reputation, for this project it was decided that Twilio is preferable to Plivo. Although it is difficult to measure reputation accurately, Twilio was founded in 2007 and has an group of clients including PayPal, Uber, Sprint and Airbnb, compared to Plivo, which was founded more recently in 2011.

Semantic Analysis - Measuring semantic relatedness between texts

Semantic comparison of text will be used for two purposes within the project: to match keywords from datasources, for example, if a user asks `how tall` something is, the property `height` would be appropriate to return; and to match questions with identical meaning to each other to save the application doing a

[2] This is referenced on a marketing page (https://www.plivo.com/twilio-alternative/) where Plivo compare their product to Twilio's, so is likely to be biased

full analysis of the data sources again. The requirements for these use-cases are different: for the former the tool needs to compare pairs of either single words of short strings or words for semantic similiarity; and for the latter the tool needs to be able to compare full sentences for semantic similarity.

Semantic similarity is a measure of how semantically alike entities are in their meaning, for example, `bank` and `trust company` [**?**]. A more general measure exists called Semantic Relatedness. Whereas semantic similarity compares entities by their meaning, entities may be semantically related by additional lexical relationships such as antonymy (the opposite of an entity, for example, `tall` and `short`) and meronymy (entities that are a part of another entity, for example, `car` and `wheel`) [**? ?**]. Both use-cases in this application will be using semantic similarity as semantic relatedness is too general. An example using the datasource keyword matching usecase is that using semantic relatedness, the word `tall` would be related to `width` as well as the target word of `height` as a result of antonymy, potentially cluttering results. In the second use-case, consider a highly rated question in the database of `What is the population of England?` and then the question of `What is the population of London?` being processed. London is a part of England, so there is the potential for the population of England to be returned instead of the population of London.

A key requirement for the algorithm and method used to compute semantic similarity is that it can be implemented in compliance with the system requirement in section 3.2.3, point 10, that the system is easy to implement. A number of algorithms were discovered for calculating semantic similarity based on the WordNet lexical database [**?**], such as the Leacock-Chodorow and Resnik algorithms. Although mathematical representations of these algorithms are published [**? ?**], to fulfil the aforementioned requirement a pre-implemented semantic comparison tool is needed as implementation of these algorithms would be a non-trivial software engineering project. Both of the Leacock-Chodorow and Resnik algorithms are available in a Perl package as part of WordNet::Similarity, a set of Perl modules with an optional web interface for integration into applications. No other open-source libraries were found, so during application design, an attempt was made to use the WordNet::Similarity tools by calling the Perl scripts from Python code. This proved complex as there were numerous out of date dependencies to trace and install.

An alternative to using a specific algorithm in a library was to use an online semantics package. Two were identified: the Cortical Compare API [**?**]; and the Dandelion Similarity API [**?**]. Both take two pieces of text and return a value representing the semantic similarity of the texts. The Cortical API was found

to be better suited for comparing single word or short multi-word terms using its `term` type [**?**], often returning errors when queried with two texts with a similar syntax, and in contrast the Dandelion API is better suited to single or multi-sentence blocks of text [**?**].

Both the Cortical and Dandelion APIs interface via a HTTP REST interface, so will be simple to implement. They are also in active development with support teams. For these reasons, they have been selected to be the semantic comparison tools used in this project.

Semantic Analysis - Discovering Named Entities in text

A named entity extraction tool is required to establish the geographical entity ('place') in a users' question. The requirements of the tool were:

1. Accepts a string of text and returns a list of keywords representing the place.
2. Keywords are categorised by a group classes, allowing for differentiation between a keyword representing a place, concept or person.
3. Accessible via a HTTP REST API for ease of implementation (fulfilling functional system requirement 10 in section 3.2.3).

During initial research, two keyword extraction APIs were discovered: the Cortical Keyword Extraction API [**?**] and the Dandelion Named Entity Extraction API [**?**]. However, the Cortical API does not fulfill requirement 2, simply returning a list of keywords. In comparison the Dandelion API returns a complex data structure which includes a list of DBPedia classes that the keyword is an instance of.

Research was done to identify a way of using the Cortical API to extract keywords and subsequently categorise the keywords after they have been been identified; however as there was no significant benefit of the Cortical API over the Dandelion API this was discounted and the Dandelion API chosen.

Results from the Dandelion API were checked against the `Place` class in the DBPedia Ontology by checking their `types` list (a list of DBPedia classes that the keyword fits into) for the string `http://dbpedia.org/ontology/Place`.

5 Implementation

This section will contain information on how data is stored, the structure of databases, and the format of and method through which data is sent to an SMS provider.

Code used within this project and discussed in this section is located in the following Git repositories:

- `https://github.com/samheather/dissertationServer` - This repository contains the code for the server application.
- `https://github.com/samheather/dissertationClient` - This repository contains the Xcode Project and code for the Mac OS X testing client, used in development of the application.
- `https://github.com/samheather/dissertationIos` - This repository contains the Xcode Project and code for the iOS client application, used in the experimental evaluation of the application. It has been constructed for an iPhone 5 screen size.

5.1 Implementation Overview

As discussed in section 3, the implementation was completed whilst following the iterative development process. Software development took places in stages which represented each iteration, and after each iteration a version of the software was demonstrated in a weekly meeting. In these meetings, the software's capabilities and success rate were discussed, and requirements were re-evaluated based on progress. Ideas for expansions to the software (for example, the Natural Language Processing, which was not in the original plan) were also discussed.

The software has been constructed in a modular manner to allow easy future expansion and modification.

5.2 Tools and services used

The following tools and services were used in the implementation. These were chosen to fulfil the requirements detailed in section 3.

- Python was the programming language used in this project. This was due to a number of reasons, including that the author had experience with writing an application with some similar characteristics as those in this project.
- MongoDB version 2.6.6 was used as the database software in this project.
- Both the application server and the database server were hosted on an Ubuntu version 14.10 "2 GB General Purpose v1" Linux server, hosted by Rackspace in London.
- No integrated development environment (IDE) was used in the development of the server application. The author chose to use textural editors due to familiarity with them, and it was felt that the assistance provided by an IDE with managing modularity/abstraction/semantic checking would not be of significant benefit, bearing in mind the size of the server application.
- An IDE was used in the development of the client applicatios to reduce the use of expensive SMS during testing. This was written in Objective-C for Mac and iOS, and created in the Xcode IDE, which is the industry standard.
- The Twilio SMS Service [**?**] was used to facilitate the SMS interface to the server application.
- The Cortical [**?**] and Dandelion [**?** **?**] APIs were used for semantic analysis of text.

In addition, the following Python libraries from the Python Package Index [**?**] were used within the server application:

1. Greenlet [**?**] - provides a notion of micro-threading (coroutines), which allows the server to process multiple HTTP requests concurrently.
2. gevent [**?**] - a coroutine based networking library for Python. gevent provides a Web Server Gateway Interface (WSGI) server that supports processing multiple requests concurrently, using Greenlet.
3. PyMongo [**?**] - Python library containing tools to work with MongoDB databases.
4. ujson [**?**] - Twilio uses JSON as the format for HTTP requests, so it makes

sense to use it for all requests. ujson allows for effecient encoding and decoding of json to and from Python data structures.

5. Twilio [**?**] - Twilio library used for receiving and sending SMS.
6. lxml [**?**] - Used in the 'Wikipedia Utils' library, which handles interactions with the Wikipedia API. lxml is used for processing XML/HTML in Python.
7. Flask [**?**] - a micro-framework for web applications, used for its simple syntax for defining HTTP routes (URLs that trigger Python functions). Flask also comes with a WSGI server, but it does not support processing multiple HTTP requests concurrently, so it is replaced with the Gevent WSGI server in point 2.
8. NumPy [**?**] - Numerical Python (NumPy) is used for outlier detection in the rating system. It provides a function call for calculating the standard deviation of a set of numbers.
9. re - a Regular Expression matching library built into Python which is used in the 'Wikipedia Utils' library and in application code for converting CamelCase text to Snake case text.
10. Wikipedia Utils [**?**] - a library used for abstracting the Wikipedia API.

5.3 Implementation of natural language processing module

In this section, a description of how the natural language processing module was implemented is given.

It was decided that a module should be constructed specifically for the parsing of natural language questions in the application. This module, called nlp, processes question in natural language form, and is able to output the two parameters needed by the application for answering a question.

The first step in the process is to remove punctuation from the question text, which is of no meaning in the context of this application. Consider an example sentence `How tall is the Eiffel Tower?`. In this sentence, the question mark will be removed to give `How tall is the Eiffel Tower`.

The next step in the process is to identify the geographical entity that a user is investigating. This is done by passing the original question into a Named Keyword Extraction tool, which extracts a list of keywords representing entities

from a string and and returns them. The Dandelion Named Entity Extraction tool was selected as in addition to providing a list of words representing entities, it categorised them according to the DBPedia Ontology. By using this, it was possible to identify the geographical entity by simply checking each keyword for the type `http://dbpedia.org/ontology/Place` in the types associated with each keyword. This is done by the `extractEntities()` function in the nlp module.

The final step in the process is to identify the property that a user is requesting about the entity, for example, the height. This is done in the `stripToProperty()` function in the nlp module. This function identifies the property by assuming it will be the only text in the sentence with meaning, excluding the place name. As such, the first action of this function is to remove words relating to the name of the geographical entity being queried, which it does by checking each word in the question text against each word in the place string, and removing the word if they are equal. After this process, our example question will have been changed from `How tall is the Eiffel Tower` to `How tall is the?`. The next step is to remove meaningless words that do not identify the property that the user querying. These are stop words, such as `and, the, of`. As with removing words relating to the geographical entity, this is done by checking each word in the question text against a list of stop words and removing them if they match. In our above example, the question text will now be `tall`. This is the property determined to be the subject of a user's query.

Having called both `extractEntities()` and `stripToProperty()`, the nlp module packages these results into a dictionary and returns them as the processed representation of the question text.

5.4 Implementation of answer location system

In this section, a description of the process followed to answer a question about a property of a geographical entity is presented.

After having identified the place and the property of that place in the question text using the nlp module, the application must locate the value of that property for that place. This is done in the `sourceProcessor` module, which collates data from the data sources and then attempts to identify from those sources an answer to the user's question.

The entry point into `sourceProcessor` is the `findArgumentOnPage()`

function which takes the place name and the property being queried as input parameters. The first step of `findArgumentOnPage()` is to query the data sources for a dictionary of properties about the geographical entity, pulling down all information available on the entity in question. Interfaces to data sources are factored out into modules, such as `dbpediaParser` and `wikiPageParser`, which handle queries to these data sources and return the data to the application in a normalised form (a dictionary of keys and values about the geographical entity). The modules controlling the data sources have a common interface: a function `getInfobox()` which returns a dictionary of keys and values from a source. Additional data sources can easily be added to the application by conforming to this interface.

After having retrieved a list of properties about the entity, the application then iterates through the datasets (one dataset per source) checking the keys in the dictionary against the requested property in the question text. If any key in the keys for a dataset is an exact, case-insensitive match to the requested property, the value for that key is returned as the answer. Otherwise, each key in the dataset is compared to the requested property text for semantic relatedness. This tries to match key strings given in different contexts and is useful in the case that a user asks a question using a different word to represent the property they require than the one in the data source. After having compared each key to the property given by the user, the key with the highest semantic similarity is returned.

An example of where the semantic comparison and matching of keys is useful is the question `How tall is the Eiffel tower?`, where the property string is `tall`, and in the data source the key might appear as `height`. The word `tall` has a high semantic relatedness to the word `height`, so the value for `height` will most likely be returned, unless another key with higher semantic similarity is found.

This semantic comparison is done by the the `compare` module, which has a function `similarityOfProperty()`. This function takes two strings as an input, and returns a value representing the semantic similarity of the two strings. The Cortical API [**?**] is used for the semantic comparison.

The Cortical API is not configured for recognising text in CamelCase, so helper functions such as `camelCaseToSnakeCase` and `camelCaseToSpace` are also included for converting CamelCase to either Snake Case or space-separated words.

Initially it was planned to compare questions to historical highly rated questions with extremely high semantic similarity. If the database contains the highly rated

question `What's the height of the Eiffel Tower?` and then the new question of `How tall is the Eiffel Tower?` is asked, the answer that was provided for the first question would be sent for the second question. Unfortunately there were issues with implementing this such as how to efficiently check a new question against every historical highly rated question for semantic similarity. Although it would be possible to implement using keywords for a question to limit the number of pairs that were compared, and by storing semantic representations locally and therefore doing the comparison locally, this would have taken too much time. As a compromise, the current version of the software does collect question texts and their ratings to begin building a database. In the future when this feature if this feature is implemented, this historical database will be useful as it means there will already be data for the application to use rather than having to wait to build up a dataset of full question texts and ratings. wording?!

The comparison of question texts is also done in the `compare` module which contains a function `similarityOfQuestion()`. This function uses the Dandelion [**?**] similarity API to compute the semantic similarity of two longer pieces of text such as questions to the application.

5.5 Implementation of rating and learning system

5.6 Implementation of Testing Client

To aid development of the server application, a desktop client application was developed which interacted with the HTTP interface of the server application to allow queries to be tested quickly as the software was developed. The application, shown in Figure 5.1, allows both the user's phone number and the question parameters to be set and sent to the server. The raw returned output is then displayed regardless of whether this was an answer or an error, thus making testing and debugging both quicker and easier.

The application was built in Objective-C using the Xcode IDE, as described in section 5.2. The application used the phone number and question text variables to construct a HTTP query which was then sent to the server. The responses was then displayed in the application. The simplicity of this application makes it easy to port to other platforms, as was done in section 5.7.

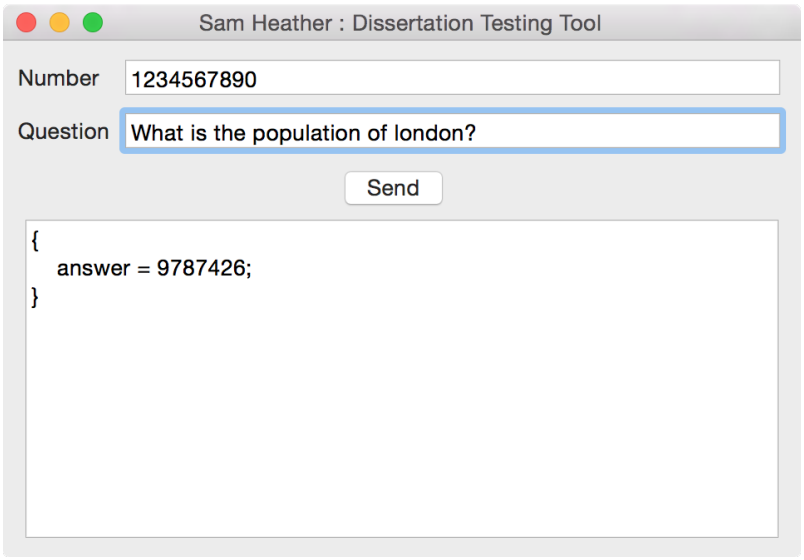

Figure 5.1: Development and testing client for Mac OS X showing a sample query

5.7 Implementation and iteration of iOS app used in experimental evaluation

As described in section 3.5 an iOS app was created to act as a query interface for the server application for the experimental evaluation. This was written in Objective-C meaning it was trivial to port the implementation used in the testing client in section 5.6.

After running the dry run of the experiment three usability issues were identified resulting from the use of the iOS Simulator on the Mac platform: the on-screen keyboard remained visible covering text on some longer answers; the participant ID field at the top of the screen caused confusion with one dry run participant attempting to change the value; it was slow to clear the question field to send a new query. Also, both users would press `return` as opposed to clicking the `Go` button, which wouldn't send a question.

Fixes were made to the software before running the full experimental evaluation such as:

- Questions are now sent both if either the `Go` button is tapped or the `return` button is pressed on the keyboard.

- When a query is sent, the on-screen keyboard is hidden as seen in Figure 5.2. It can be brought back by tapping on the question field.
- When selecting the question field, all the text is selected by default making it easier to type a new query without spending time deleting the old text. A 'clear field' cross button which is standard on the iOS platform is also shown. This is shown in Figure 5.3.
- The field to enter participant ID is hidden, as shown in 5.2. This is now requested in a modal alert when the application starts: the ID is then entered by the researcher.

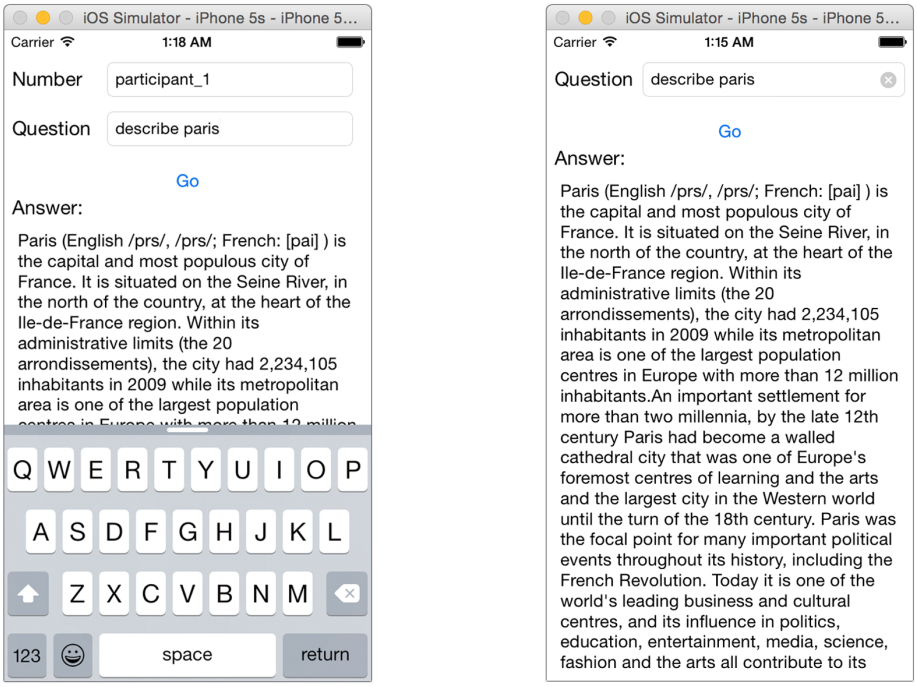

Figure 5.2: Left: Original application. Right: Revised application with keyboard hiding.

5.8 Discussion of Implementation

- Tweaking the constants for best results - Excluding wikipedia due to dirty data

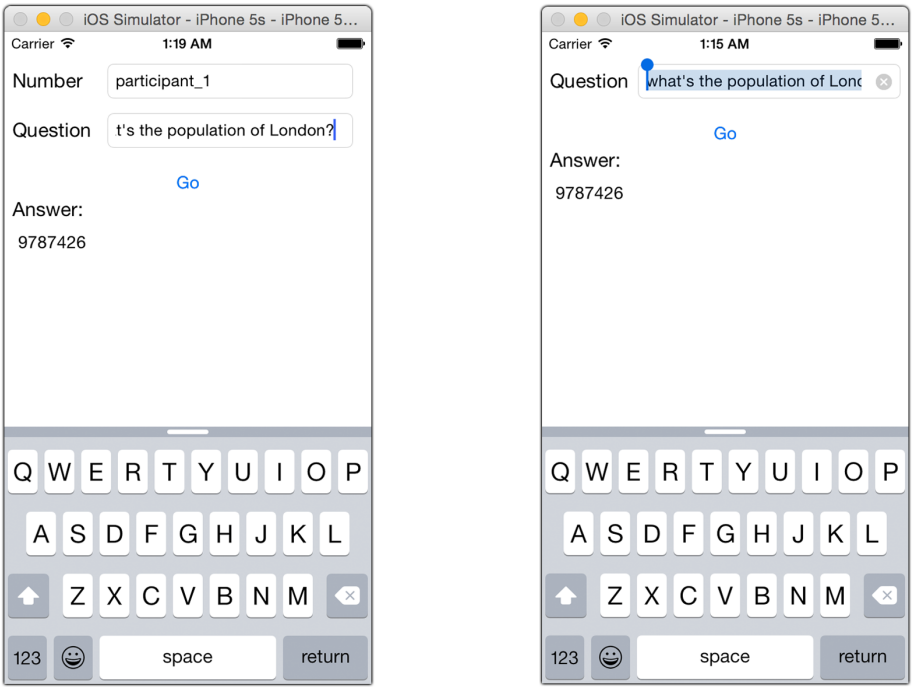

Figure 5.3: Left: Original application. Right: Revised application with hidden participant ID field and Question field with a clear button and auto-selection of text when it's tapped to bring the keyboard up.

Key issues:

- can't handle units

- sometimes wikipedia attaches other garbage

- can't do subjective stuff 'depth of the deepest ocean'

- can't do properties that are stored deep within the ontology.

6 Results

This section's content...

7 Discussion

This section's content...

8 Evaluation

This section’s content...

9 Extending this project

Section not complete If the author was able to expand this project, areas for expansion include:

- Addressing the issue of literacy
- Automating the expansion of the database
- Units for metrics.
- Make it work with all of wikipedia (all pages with infoboxes anyway)
- make it work with all languages
- use full on random sparql queries to do magic
- nlp assumes property is the only other bit of text with meaning. e.g. thank you would break everything.

10 Conclusion

This section’s content...

Part I

Appendices

A Personas

A.1 Nanjala

A.1.1 Demographics and Profile

- Female, aged 15
- Living in the village of Amuria, Uganda, 300km from the capital Kampala. Internet access is difficult and expensive to obtain in Amuria, though there is good mobile network coverage.
- Currently in general education at her village school.
- Nanjala is curious and wants to be able to ask questions and retrieve and read more information at her own leisure. She prefers to be told about something than to learn facts.

A.1.2 Average Day

In a typical day, Nanjala will rise early to help her family with housework chores, before she prepares for school. Nanjala wants to become a teacher. At school, she studies Science, Maths and Geography, and works hard for her classes and on her homework. Nanjala's school has a small library and a large body of students, meaning that books are often unavailable when Nanjala would like to look something up. This increases the amount of time that her homework takes, and can even result in her not having access to information when she goes home.

A.2 Cedric

A.2.1 Demographics and Profile

- Male, aged 27
- Lives in Zurich, Switzerland, and goes on regular hiking trips, from 3-4 day trips around Europe to longer trips to the South American Andes and southern Asia.
- Works in a service role in the city and so has little disposable income to spend on guide books for the many places he visits.
- Cedric is a competent albeit not professional hiker, often going off-track and summiting 'difficult' peaks or reaching famous camping grounds. He likes to be spontaneous, only deciding which peaks he will attempt when he arrives.
- Cedric is very conscientious and wants to contribute his own knowledge to the world, especially to projects that help him.
- Cedric values his privacy is concerned about how much information he gives out online.

A.2.2 Average European Trip

A typical trip will involve an intercity train from the Zurich Hauptbahnhof (the main station) to the city closest to the range Cedric will be hiking. He takes the train because it tends to be cheaper than flights. Once in the area, Cedric will take local trains and buses to get into the mountain ranges. There, he will use the local hiking trail network signs and conversations with the locals to select a peak or route. Once Cedric has got moving and is reaching higher ground, he uses a combination of his view and the signage to identify other peaks that he would like to wander around.

Cedric takes a non-smart phone with him because a) the battery life on his smart phone is not long enough for his whole trip, whereas his non-smart phone lasts a week at a time, and b) his smart phone would not be useful anyway, since the internet is so intermittent. He still wishes he could get information such as whether there are maintained routes on the slopes of a given peak or the height of a peak that he sees without having to phone friends back in the city.

B Sample Consent Form

System to access knowledge using Machine Learning and SMS - Usability and Usefulness Study

Informed Consent - Introduction

This study aims to investigate the usability and usefulness of a system developed to provide access to knowledge using machine learning and SMS. The system aims to provide high quality answers to questions asked of it, where the questions relate to either a description or property of a geographical entity (for example a city, river or mountain). The system aims to learn where mistakes have been made and improve by receiving feedback from the user on the previous answer.

Form: please complete this section before the study.

I, _____, agree to participate in this study. I have been briefed on and understand the purpose and goals of the project.

I understand that information will be collected from me during this study in the form of my answers to a questionnaire. Additionally, I understand that my interactions with the application will be recorded, specifically the questions I ask, answers returned and my quality ratings of answer. I understand that this information will be treated confidentially and only Sam Heather and the project supervisor (Lilian Blot) will have access to the original data in its original format. When the information is processed, shared, described or interpreted I will not be personally identified.

I understand that I can withdraw from the study at any point without giving a reason.

Date: _____Signature: _____

Researcher's contact details: Sam Heather, sam@heather.sh

Supervisor's contact details: Lilian Blot, lilian.blot@york.ac.uk

Post experiment consent form - please complete this section after the study.

□ I have been suitably debriefed about the system and my performance.

□ I understand I can withdraw my data from the study at any time.

□ I have not been forced to take part in this study.

□ The Researcher has treated me with respect and answered all of my questions.

For the researcher: Participant ID: _____

C Experimental Evaluation Instruction Sheet

System to access knowledge using Machine Learning and SMS - Usability and Usefulness Study – Instructions

Introduction

This study aims to investigate the usability and usefulness of a system developed to provide access to knowledge using machine learning and SMS. The system aims to provide high quality answers to questions asked of it, where the questions relate to either a description or property of a geographical entity (for example, a city, river or mountain). The system aims to learn where mistakes have been made and improve by receiving feedback from the user on the previous answer.

Your Task

With consideration to the goals of this study in mind, you should attempt to locate specified information using this tool. You should ask questions in natural language form as you would a friend, for example:

> "How tall is the Eiffel Tower?".
> "What's the population of Mexico city?"

You will either be asked to locate a specific property of information (such as the height of a monument), or to locate general information. You should note that if the primary method of finding information (that of asking a question) should fail, you can always ask for a general description of the entity using the syntax:

> describe X

The system should learn which answers are of a particularly high or low quality. To allow it to do so **please rate every answer you receive** with a star rating between 1 and 5 by replying with a message such as:

rate 1	(for a poor quality answer)
rate 3	(for a neutral quality answer)
rate 5	(for a high quality answer)

After the experiment you will be asked to fill in a survey about your experience using this application.

Questions to Ask

Please locate the following information using the application. You are encouraged to construct your questions using a variety of terms, not just those provided below. You are not required to write down or record the information you find. If no answer is returned, you may ask again in a different way, however if an answer is returned you should not ask again, even if the answer is obviously incorrect. Please remember to rate every answer you receive.

Please identify:

1. the height of the Eiffel Tower
2. the height of another monument of your choice
3. the number of people living in London.
4. the depth of the English channel
5. the year that the Eiffel tower was built
6. general information on Paris
7. the elevation of mount Everest
8. the currency that we use in England
9. The height of the Leaning Tower Of Pisa
10. the length of the amazon river
11. a general description of Snowdon mountain.
12. the population of a country of your choice?
13. the GDP of Russia
14. the country in which Barcelona is a city
15. the area of a country of your choice
16. the discharge of a river of your choice

D Sample Questionnaire

Please fill in the following questionnaire with regard to your experience of the system that has been developed.

#	Question	Agreement 1 = strongly disagree 3 = neutral 5 = strongly agree
1	Using the system would enable me to locate information more quickly when without access to the internet.	1 2 3 4 5
2	Using the system would improve my performance when working or researching without access to the internet.	1 2 3 4 5
3	Using the system would increase my productivity when performing casual research without access to the internet.	1 2 3 4 5
4	Using the system would enhance my effectiveness at finding information without the internet.	1 2 3 4 5
5	Using the system would make it easier for me to find information without access to the internet.	1 2 3 4 5
6	I would find the system useful in day-to-day life when without access to the internet.	1 2 3 4 5
7	Learning to operate the system was easy for me.	1 2 3 4 5
8	I would find it easy to get the system to do what I want it to do.	1 2 3 4 5
9	My interaction with the system would be clear and understandable.	1 2 3 4 5
10	I found the system to be flexible to interact with.	1 2 3 4 5
11	It would be easy for me to become skillful at using the system.	1 2 3 4 5
12	I found the system easy to use.	1 2 3 4 5
13	The information returned was useful, relevant, and answered the question asked.	1 2 3 4 5
14	I trust the answers returned by this application.	1 2 3 4 5

Please turn over for demographic questions.

15. Your Age:

☐ 18-24 years old
☐ 25-34 years old
☐ 35-44 years old
☐ 45-54 years old
☐ 55-64 years old
☐ 65-74 years old
☐ 75 years or older

16. What is your occupation?

☐ Employed for wages
☐ Self-employed
☐ Out of work
☐ A homemaker
☐ A student
☐ Military
☐ Retired
☐ Unable to work
☐ Other: ……

17. Frequency of Internet Usage:

☐ Usually more than once per hour
☐ Multiple times per day
☐ Once per day
☐ Multiple times per week
☐ Once per week
☐ Less than once per week

For the researcher: Participant ID: ____