Generating Route Instructions with Android Devices based on the GRUVE Challenge

Brian Migot
H00138659

August 2013

Computer Science
School of Mathematical and Computer Sciences

Dissertation submitted as part of the requirements for the award of the degree of MSc in Software Engineering
Generating Route Instructions with Android Devices based on the GRUVE Challenge

MSc Software Engineering
Dissertation

Author: Brian Migot (H00138659)
Project Supervisor: Oliver Lemon
Second Reader: Roger Rist
Declaration

I, Brian Migot, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: .................................................................

Date: .................................................................
Abstract

This report describes a spoken language system that provides pedestrian navigation directions from a smartphone. This program is using a speech recognition system to allow the user to interact with the device. One of the main aims of the project is to generate natural language from the device to provide directions to the user. The second aim of the project is to develop two language generation algorithms, one that is using a “Baseline” algorithm to give instructions along the way, the other one giving the user a “Summary” at the beginning then giving direct instructions along the way. Both of them are also using landmark when giving directions.

The algorithm for the system is based on the GRUVE project which is for now located and testable on a web browser. This algorithm has been adapted and transposed to an Android application. Some adaptations have been made to have a good interface on the smartphone. The evaluation has been done with the web browser version. What has been tested is the Baseline algorithm against the Summary algorithm. For both of them, the criteria tested are the length of the answer, which one was easier, more efficient and user preferences.

The program has then been tested with candidates from outside the project to see how it is working with an outsider point of view. The results have then been analysed to see a comparison of the two algorithms, if the system was functional and what is the better one depending on the situation. The Baseline NLG was judged more effective at the end of the evaluation.
Acknowledgements

I would like to take a little part here to thank everyone that supported and helped me realize this project.

I would like at first to thank my supervisor, Oliver Lemon, who guided and supported me during the project, giving me valuable information for continuing and improving what I was doing. It is also my pleasure to thank Srinivasan Janarthanam who helped me a lot understanding the Gruve Challenge and helped me setting up the experiment. I would also like to thank Xingkun Liu for setting up the server which was used to run the experiment.

I would like to thank everyone else that supported me during this project even if their name was not mentioned here. Thank you everyone.
# Table of Contents

1. Objectives .......................................................................................................................... 9

2. Literature Review .............................................................................................................. 12
   2.1. Introduction .................................................................................................................. 12
      2.1.1. State of the Phone industry .................................................................................. 12
      2.1.2. Application of Speech Recognition ...................................................................... 13
   2.2. Speech Recognition and Natural language generation ............................................... 15
      2.2.1. Speech Dialogue System (SDS) ......................................................................... 15
      2.2.2. Natural Language Generation Overview .......................................................... 17
      2.2.2.1. Surface Realization ......................................................................................... 18
      2.2.2.2. Discourse planning ......................................................................................... 20
      2.2.2.3. Micro planning ............................................................................................... 22
      2.2.3. NLG Planning ...................................................................................................... 23
   2.3. Geolocation .................................................................................................................. 25
      2.3.1. The GIVE Challenge ........................................................................................... 25
      2.3.2. GRUVE Challenge ............................................................................................. 26
      2.3.3. Web-based pedestrian navigation system (for GRUVE challenge) ..................... 27
      2.3.4. Other Location Based Games ............................................................................. 29
   2.4. Android ....................................................................................................................... 30
      2.4.1. Overview .............................................................................................................. 30
      2.4.2. Google API ......................................................................................................... 31
   3. Tools and Methodology .................................................................................................... 33
      3.1. Tools .......................................................................................................................... 33
      3.2. Methodology ............................................................................................................. 33
   4. Requirements Analysis ..................................................................................................... 36
   5. Professional, legal, ethical, and social issues ................................................................. 39
6. Project Plan ........................................................................................................... 41
   6.1. Overview ........................................................................................................ 41
   6.2. Project Schedule .............................................................................................. 41
   6.3. Risk .................................................................................................................. 44
7. Implementations ..................................................................................................... 45
   7.1. Technical Decisions ......................................................................................... 45
   7.2. Organisation of the GRUVE system (baseline version) ................................. 45
   7.3. Android Program Design ................................................................................ 47
   7.4. Android Program Implementation .................................................................... 52
      7.4.1. General Description .................................................................................. 52
      7.4.2. Location .................................................................................................... 54
      7.4.3. Voice recognition + TTS .......................................................................... 55
      7.4.4. GRUVE implementation .......................................................................... 56
   7.5. Summary algorithm design and implementation .............................................. 56
8. Evaluations .............................................................................................................. 60
   8.1. Method ............................................................................................................. 60
      8.1.1. Designs ...................................................................................................... 60
      8.1.2. Participants ............................................................................................... 60
      8.1.3. Procedure .................................................................................................. 61
   8.2. Results .............................................................................................................. 64
      8.2.1. Task Completion ...................................................................................... 64
      8.2.2. Time completion ...................................................................................... 65
      8.2.3. System usefulness ..................................................................................... 66
      8.2.4. Ease understanding of the instructions ...................................................... 66
      8.2.5. Accuracy of instructions .......................................................................... 67
      8.2.6. Length of the instructions ....................................................................... 68
Table of Figures

Figure 1 "Conversation example between the system and the user." .............................................. 10
Figure 2 “Example of a design interface on the phone” ................................................................. 10
Figure 3 “Web Browser version of the project” .............................................................................. 11
Figure 4 “Data from Emarketer [22]” ............................................................................................. 13
Figure 5 “Typical Architecture of a Speech Dialogue System” [9] .............................................. 15
Figure 6 “Natural Language Understanding frame example” [9] ................................................ 16
Figure 7 “Simple Natural Language Generation Architecture” [8] ............................................ 18
Figure 8 “A simple Systemic Grammar” [8] .................................................................................... 19
Figure 9 “Example of Functional Unification Grammar list” [8] ................................................. 20
Figure 10 “Text Schemata Architecture” [8] .................................................................................. 21
Figure 11 “Example grammar in the sentence generation domain” [12] ........................................ 23
Figure 12 “Derivation of the white rabbit sleep” [12] ................................................................. 24
Figure 13 “Example of a screen view of GIVE-1” [11] ............................................................... 26
Figure 14 “GRUVE Architecture” [7] ............................................................................................ 27
Figure 15 “GRUVE Evaluation Architecture” [6] ........................................................................ 28
Figure 16 “City Explorer Representation” [13] ............................................................................. 29
Figure 17 “Iterative Waterfall Model” [26] .................................................................................. 34
Figure 18 “Terms of agreement” ................................................................................................... 39
Figure 19 "Gantt Chart of the Project" ............................................................................................ 43
Figure 20 “Class Diagram of the GRUVE System (BaseLine)” ................................................... 46
Figure 21 “Title screen of the Android application” ................................................................. 47
Figure 22 ”Main screen: beginning” ............................................................................................ 49
Figure 23 “Main screen: Manual Mode” .......................................................................................... 50
Figure 24 “Main screen: Voice Mode with Character” ............................................................... 51
Figure 25 “Android Application Class Diagram” ......................................................................... 53
Figure 26 “Azimuth Angle Description” [29] ............................................................................. 55
Figure 27 “HWUNLG description” ................................................................................................ 59
Figure 28 "Evaluation Screenshot" ................................................................................................ 61
Figure 29 “Profile information” ....................................................................................................... 63
Figure 30 “System Questions” ......................................................................................................... 63
Figure 31 “Task completion” .......................................................................................................... 64
Generating Route Instructions with Android Devices based on the GRUVE Challenge

Figure 32 “Time completion” ................................................................. 65
Figure 33 “Average score for system usefulness” ............................................. 66
Figure 34 “Average score for ease understanding of the instructions” ................. 66
Figure 35 "Average score for the accuracy of the instructions" ......................... 67
Figure 36 “Average score for the length of the instructions” ............................. 68
1. Objectives

The general idea of my project is to develop a pedestrian navigation system on an Android phone using the web-browser game already developed by the GRUVE project as a foundation. The entire algorithm will be analysed and then incorporated and adapted into an Android application. Another algorithm will also be created, this one will be giving a summary at the beginning of the route planning, and then it will be compared to a “Baseline” direction like what was done on the GRUVE project. For both of the algorithm landmarks will also be added while giving direction.

The two algorithms will then have to be tested by people outside the project and then the result of the tests will be analysed. The main research question is what people think of the summary algorithm. Is the summary something they want or do they simply prefer direction going straight to the point?

From what has been written previously, my project can be easily separated into 3 big stages. The first stage will be to implement the Android application using the pedestrian navigation algorithm from the GRUVE project. For now the algorithm is working on a web-browser and “Google Street view” being the view field of the user [6] (Janarthanam et al. 2012). The design will have to be easy to use and intuitive for the user. No map should be seen on the screen, the phone will directly give the direction by voice. Then a second algorithm, the summary one mentioned previously should be implemented, it should work on the Android version but more importantly on the web browser version since it shall be tested there.

Here is an example of what should be a simple conversation between the user and the system during the game.
Generating Route Instructions with Android Devices based on the GRUVE Challenge

Below is another example of what could be a good design for the application at the end of the project. In the end the final design is different but it is always nice to see the first prototype.

![Image of a design interface on the phone]

The phone may vibrate or ring when the user will arrive close to a character or event while walking around.

Click on the button below to talk to the phone.

Figure 1 "Conversation example between the system and the user."

Figure 2 “Example of a design interface on the phone”
Figure 3 is showing what to expect from the web browser design.

Figure 3 “Web Browser version of the project”

Once the implementation will be completed, the second step will be to test the application. What will be tested is the Summary algorithm; it will be compared to the Baseline algorithm. Both of them will be provided with landmark, this way, only the summary part is different and it will be this that user are comparing. This will be tested in the web browser interface. It will be uploaded on a server then it is simply accessed from an Internet link. Two games shall be implemented to let the tester be able to experiment each system.

Finally, after the test is completed, an analysis will have to be done. A questionnaire will be given to the testers at the end of each game. They shall judge if the system helped them find their way, did they like the length of the answer (knowing if the summary algorithm was too long or Baseline too short) and other questions. After this, the analysis shall determine what the users thought of both algorithms.

So the main hypothesis of this application is that users getting a Summary NLG will have faster and better task completion and they will also prefer this system compared to the Baseline system which doesn’t provide a summary.
2. Literature Review

2.1. Introduction

2.1.1. State of the Phone industry

Speech recognition is a domain in expansion, a lot of new technologies are developed with speech as their main features and the algorithms are becoming more complex and efficient. Smartphones are using this in a lot of different applications. The first important step has really been with Apple when they launch Siri [17] (Apple). This has then expanded to Android and now a lot of different companies are using this for their own application. In general Speech Recognition is becoming a subject gaining a lot of presence with the advance of different technologies. Now that there are web-based language techniques, different machines using dialogue system and speech synthesis becoming really efficient, it is becoming more and more used by everyone.

Smartphones in general are also gaining a lot of success compared to others normal phone devices. According to the article from the eMarketer website [22] (eMarketer 2013), using data from Portio Research, Smartphone users should continue to increase in the years to come. At the end of 2012, the United Kingdom had approximately 83 million mobile subscribers, 36.8% of them are smartphones. From their research, it is a really big progression, since in 2010, only 20% of the phones in the market were smartphones. The rest of the world is not far behind, and smartphones are becoming the type of phone that will be the most used. According to Portio Research, in 2016, 78% of the mobile phones users in UK should be using smartphone; so nearly everyone.
Generating Route Instructions with Android Devices based on the GRUVE Challenge

Brian Migot

Figure 4 “Data from Emarketer [22]”

About the state of the OS industry of smartphones at the beginning of 2012, Android has already more users than Apple in the UK with 36.9% of Android smartphones users against 28.5% of Apple smartphones users, according to an article from the Guardian [24] (2013). The competition is really fierce, since not too long ago Apple was clearly ahead in the smartphone market.

2.1.2. Application of Speech Recognition

There are a lot of different applications of Speech Recognition on the market. In this report, the focus will be on the different applications and technologies available for smartphone. One of the first main important application in this field is Apple’s Siri [17], Apple Siri has been introduced to the iPhone 4S in 2011. It was the first application on smartphone using dialog system. It could check the weather for you or create a text showing what you were just saying.
Another example is that you could ask it to transform your oral message to a text message then simply say to send it to a contact in your list. However Apple is not stopping there and Apple is continuing to improve Siri along the newer versions of IOS.

The direct concurrent of Apple, i.e. Google also have its own “Siri” application, even though it is not completely similar. The application is called “Google Voice Search” [18]. At the beginning it was much simpler than Siri and was only use to transcribe voice messages to texts. However, now Google Voice Search is able to hold a dialog more or less like Siri. It’s also faster than Siri when searching for information on the web [27]. This application is not only for Android but also available on IOS, this is why the previous comparison is possible. The main difference here is that Google Voice Search doesn’t have access to the inside of the IOS, so the application can’t communicate with others applications and Siri still has the advantage on IOS devices. Google also has “Google Now” [19] which is an application that gives you the required information whenever you need it, to take the catchphrase used by Google: “Google Now gets you just the right information at just the right time”. This application is not really specific to Speech Recognition since it can be completely used only with screen inputs but it is compatible with Google Voice Search so the information can be directly asked by voice. With the combination of Google Voice Search and Google Now, Google is really effective to bring a lot of information for a smartphone user.

There are a lot of other personal assistants able to give you information or try to hold a conversation with the user like Nina from Nuance [20]. Another example presented here is Utter [25] (xda-developpers). It is a new independent application and it seems really promising. For now the application is still in beta so all the different options are still not present but what is available is already working fine and really quickly. The application can handle the basics, like asking the weather, posting a message on Twitter or Facebook, calling someone in the contact list or on Skype. The user is also able to associate some keywords to execute some actions. For example if the user associates the word “supernova” to give the weather of the day, the phone will be able to understand and give back the information when the user says it. For now Utter is available on Android smartphone in its beta version for free.

The last personal assistant presented here is about “Indigo” from Artificial Solutions [28]. This personal assistant has been available recently on Windows Phone and Android. The main advantage of Indigo compared to the other applications is that the application is cross
compatible. This means that the application is compatible between your phones, tablets and computers since the user data is stored on the cloud and not directly on one of the devices. For example the user will be able to begin a search on a device and then continue on another. Furthermore, the device is also able to remember user preference and what complicated research they could have done. Of course all these capabilities are voice compatible and that make Indigo a really promising application.

2.2. Speech Recognition and Natural language generation

2.2.1. Speech Dialogue System (SDS)

![Figure 5 “Typical Architecture of a Speech Dialogue System” [9]](image)

This figure is a typical architecture for most dialogue systems [9] (Jurafsky and Martin 2009). This architecture has six components. Speech Recognition and Natural Language Understanding get the input and transfer it to the Dialogue Manager. The Dialogue Manager and the Task Manager control everything in the process but also monitor and remember what the current domain is. The last two components are used to create the speech. All of these components will be seen in more detail later.

The first component is the Automatic Speech Recognition (ASR), it takes an audio input and transforms it to a string of words. It is not necessarily the same sentence from what has been said from the input. The ASR could directly summarize the sentence to get the necessary
Generating Route Instructions with Android Devices based on the GRUVE Challenge

information to help the next steps. This is called “restrictive grammar”. The ASR should also have real time response to be quicker since the user doesn’t want to wait a long time for the system to respond.

The next component is Natural Language Understanding (NLU). This part computes a semantic representation of what has arrived from the ASR to make it understandable to the dialogue task. To illustrate this, here is an example taken from Jurafsky and Martin [9] (2009), “Show me the morning flights from Boston to San Francisco on Tuesday” could correspond to a frame like this:

<table>
<thead>
<tr>
<th>Show:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flights:</td>
</tr>
<tr>
<td>Origin:</td>
</tr>
<tr>
<td>City: Boston</td>
</tr>
<tr>
<td>Date:</td>
</tr>
<tr>
<td>Day of the week: Tuesday</td>
</tr>
<tr>
<td>Time:</td>
</tr>
<tr>
<td>Part-of-Day: morning</td>
</tr>
<tr>
<td>Dest:</td>
</tr>
<tr>
<td>City: San Francisco</td>
</tr>
</tbody>
</table>

Figure 6 “Natural Language Understanding frame example” [9]

NLU should be able to create this frame directly when the previous sentence is heared. There are a lot of different way for a NLU to realize this, for example using “semantic grammars” or the semantic HMM model of Pieraccini et al. [15] (1991).

The next module is about Generation with the TTS Components (more details about Natural Language Generation later). The generation chooses the concept to pass to the user and how to say the sentence, so it will have to decide the syntax and structure. The TTS will take this information and synthesizes this into sound. The language generation can be done in different ways but at the beginning there is always a content planner. The first part of a content planner is to decide “what to say”, for example if the user will need an answer or if further questions are needed. The second part is “how to say it”. To do it, a lot of different technologies are available; however the next section (2.2.2.) will be only focusing Natural Language Generation since it is the main domain of the project.
The last part is the Dialogue Management. This part takes the input of the first two sections, so ASR and NLU and it will pass this information to NLG and TTS components. There are four different kinds of architectures to do this. This report will just look at the simplest one which is also the most used in the industry. A system that will control the conversation is called “system-initiative” or “single-initiative” system. For example, it is used for system that will ask a lot of questions one by one and the user will have to answer them every time to continue the conversation. This can obviously make the conversation a little annoying on the user side, sometimes a user will want to go quicker and skips some parts or he will not know how to answer some question and it could lead to some problems. This can be annoying and this cannot be called a dialog. To solve this problem, most of the time there are “mixed-initiative” systems to avoid this. In this state, the conversation can switch to the user side and will not only be guided by the system. The system will often divide the necessary information in “slots” that the user will have to fill and when enough slot are filled, the system will be able to formulate a response.

2.2.2. Natural Language Generation Overview

As seen previously Natural Language Generation (NLG) is an important step in an SDS system, this part will present natural language generation in more detail. The definition of NLG from Jurafsky and Martin [8] (2000) is “NLG is the process of constructing natural language outputs from non-linguistic output”. NLG can be the seen as the exact opposite as NLU since this time the process is to transform the meaning to text but there are still two big differences. Every generation can have a different input specification opposed to NLU where there is always one type of input and NLG will always have to search for some details about what it will have to generate.

The second point is that NLG have to make choices. NLG has to choose a “content selection” so NLG should take the appropriate information and express the information depending of the goal but also about the future user, for example the answer could have fewer details about the subject since the user is already knowledgeable about it. NLG also has a “lexical selection”. A lexical selection means that the system must choose the most adequate words for a given situation. NLG also has to take care of having a good “sentence structure”. To do this, most of the time there should be an aggregation which means to separate and gather everything in phrase, clause and proposition. NLG should also know how to address the
subject of the conversation. Furthermore, it should also be able to discern the “discourse structure”, for example if there are two propositions, the system should be able to discern those two parts.

To be able to resolve all of these problems, here is a simple architecture for NLG systems.

![Figure 7 “Simple Natural Language Generation Architecture”](image.png)

The next chapters will give more details into this architecture; the most important components are explained.

### 2.2.2.1. Surface Realization

This component will receive the complete discourse plan and then generates simple sentences appropriately. For this, surface realization can be divided in two parts, “linguistic realisation” and “structure realisation” [3] (Hjalmarsson 2006). Linguistic realisation means that it will translate abstract representations into grammatically correct text and structure realisation means that it will convert the sentence to an external output like HTML, XML or simply plain text. There are two big approaches to realize this according to Jurafsky and Martin [8] (2000): “Systemic Grammar” and “Functional Unification Grammar”.

Systemic Grammar separates sentences as collection of different functions and some mappings are used to link all of those functions together with grammatical rules. A “system
network" is used to define the necessary elements, called clause, in a sentence. An example below illustrates this with mood, transitivity and theme. Furthermore, for each of those clauses, different elements are chosen to fill them. Those different elements are defined with “realization statements”. Realization statements will constraints the sentences with their specific features. Here is a figure to illustrate this organization.

![Diagram of a simple Systemic Grammar](image)

**Figure 8 “A simple Systemic Grammar” [8]**

The main idea of Functional Unification Grammar is to create a list of alternative grammars to create a sentence and then unify those different alternatives with a specified input necessary for the structure of the program. This list is using different levels to specify different actions in the sentence. For example, the highest level provides alternatives for sentences, phrases or verb phrases. Then this list is divided in three more lists to specify more details about each type of sentences. The figure below is illustrating this approach.
To conclude, it is easy to see that both of them are using function categorizations which seem to be a good way to generate language.

2.2.2.2. Discourse planning

From what has been seen before, surface realization just generates single sentences with a specified input, so it does not do anything about the discourse structure or the content of the sentence, and this is the goal of discourse planning. The word “discourse” is also used because there is sometimes a need to produce multiple sentences, the system could simply generate many sentences one by one but it will seem unnatural because people are often using “link
“words” between propositions and people are not always ending sentences to separate everything.

This phase is divided into two processes which are “content determination” and “document structuring” according to Hjalmarsson [3] (2006). Content determination creates objects that will be later used by micro planning and a surface realizer. This is more or less to separate the sentence into objects which are easier to insert in categories like what has been explained before. Document structuring is, as the name implies, to order and chunk the text. Most of the time, this depends on the domain.

Now two mechanisms will be presented to do this: “text schemata” and “rhetorical relations” [8] (Jurafsky and Martin 2000).

Text Schemata can be modelled with this figure:
Generating Route Instructions with Android Devices based on the GRUVE Challenge

So as seen on this figure, the first node S0 puts any number of preconditions inside then node S1 is the action itself and S2 represents the side effect during or after the action. This representation presents how the information is divided. With this, the structure can be provided and the surface realizer will be able to generate the sentence.

Rhetorical relations can represent more complex structures than text schemata. This time it should be able to structure exceptions, for example “I like reading old novel. However my preferred novel is from 2012”. The “however” is the important word that should be in the structure; without it, the sentence does not have a lot of sense and that is something that text schemata cannot do. To do this, a rhetorical structure is built hierarchically to put some levels between pieces of information. On the same level they can be linked together with a definition, for example if the next sentence will elaborate on the first one the definition will be an “Elaboration” and in the example before, it would have been a “Contrast”. The different levels can associate all of that information together if for example there is more than one definition. These approaches make rhetorical relations more powerful than text schemata and they can handle more complexes sentences.

2.2.2.3. Micro planning

Micro planning is also really important and it is placed in the middle of the architecture. This part represents the process of mapping the discourse plan to the surface realizer. This includes “lexicalisation”, “aggregation” and “referring expression generation” which have been quickly presented before. In accordance with what has been seen before, lexicalisation put a concept into words. Most of the time it will take care of the context, which means taking care of who is the future user but also about what has been said before. Aggregation is the process of the structuring the document together. Here is an example from Hjalmarsson [3] (2006), “John has a book” and “Mary has a book”, it could be linked together with “John and Mary have a book” and that is what aggregation should do. Aggregation can also change the order of paragraphs or sentences or even remove some parts if necessary. Referring expression generation is about how to address an entity. So for example adding more pronouns to replace a word and leaving the word intact if it is used for the first time to understand what is happening. For example the sentence “The Edinburgh Castle is on your left” could be replaced by “It is on your left” if the Edinburgh Castle had already been mentioned before.
2.2.3. NLG Planning

In this section, a specific example will be presented about how to generate NLG from what has been described previously. A lot of different techniques can be used to generate words and the one presented will be “sentence generation in the sense of Koller and Stone” (2007) [10]

According to Koller (2011) [12], this planning can be modelled with a “lexical grammar” in which everything is composed by “lexicon” and it will describe how each “lexicon” are interacting with each other’s, what is their meaning and when can those lexicon be used. An example of this lexicalisation can be seen on the figure below.

As can be seen on the figure 11, this representation is using “elementary tree” and they are all contributing to a “semantic content”. To illustrate this, here is an example from Koller’s work (2011) [12]: first there are three elements called r1, r2 and e1. We will suppose that r1 is a white rabbit, r2 a brown rabbit and e1 represent the action where r1 sleep. To represent the action {sleep(e,r1)} which is e1, part of the elementary tree have to be combined in a “TAG combination”. This is illustrated on the figure below.

![Diagram](image-url)
In the figure 12, the dashed arrow represent a “substitution”, a part of a tree is plugged into another tree from its leaf, then the action “white rabbit sleep” can be read on this new tree. This cannot represent the action “a rabbit sleep” since there are two elements corresponding to this description, r1 or r2 and this tree is specific to r1 so the white rabbit. This planning is also particularly useful since it is already doing the work for sentence planning and surface realization. However this method can be long to achieve since it is always searching from up to bottom and this can be hard to use in real application.

To avoid all these steps, Koller proposed to use SPUD (by Stone et al) with an alternative approach by Koller et al. [12] which changed the sentence generation to a planning problem. In this planning, the initial state is with an atom “subst(S,root)”, this atom means that a sentence “S” must be created and it will begin from the node “root” with a second atom “reference (root,e)”, this sentence will describe the action “e”. For the previous example “the white rabbit sleeps”, the action “add-sleeps(root,e,r1)” will be put at the place of the atom “subst(S,root)” with “subst(NP, subj(root)”. This “subj(root)” represents a new individual. With this, the operator knows that the action “sleep(e,r1)” has been created and all necessary individuals are now present. Everything must then be eliminated apart from the white rabbit r1. To do this, different operators are used to remove them. In this example, what could be used is:

1. add-sleep(root,e,r1)
2. add-rabbit(subj(root),r1)
3. add-white(subj(root),r1)

From [12]
The sentence “The white rabbit sleeps” can always be recreated with this plan. This is one of the many techniques used to generate natural language. In the next chapter (2.3), the GIVE challenge is presented and this one uses another technique for generating natural language.

2.3. Geolocation

2.3.1. The GIVE Challenge

The name GIVE means “Generating Instructions in Virtual Environments”. The GIVE challenge has begun in 2009 with GIVE-1 [11] (Koller et al. 2010), which is the first experiment done for the GIVE challenge, now they already are up to GIVE2.5 [16] (Striegnitz et al., 2011). The main focus of the GIVE challenge is to evaluate natural language generation which is hard to do. Natural language generation does not have just one answer for a problem; a lot of different outputs can be good answers.

To be able to cope with this challenge, GIVE uses a virtual 3D environment where a human user has to play a treasure hunt game. To find the treasure, the users will only have natural language instructions provided by the system. These instructions will be displayed on the screen, the test will always be in a closed environment and the user will have to navigate in this environment, e.g. open door, disable alarm... The ultimate goal is finding a trophy in a safe. All the GIVE challenges are using those set of rules for the game. The user will have to connect to the internet to have access to the game; this will allow the supervisors to get easily some data about what has happened during the game session. It is the easiest way to get a lot of information, for example asking human to take notes about each game would be too time consuming and too costly.

The first GIVE challenge (GIVE-1) divided the world in a grid so the user was only able to move in four different directions: north, east, west and south. Those type of moves limit the player compared to the real world but it was easier to see if NLG was working since it could say to simply move three steps forward which in the game would be three squares in front of the user position, there would be no way for the player to misinterpret and it is more easy for NLG to determine what to do. In the end GIVE-1 was a success and a lot of data were collected across 1143 game runs and five NLG systems.
GIVE2 (and GIVE2-5) are the next step in this challenge, both of them have the same rules as before but this time, the player can now move continuously and not discretely like before. For both of those new GIVE, new NLG systems have been tested.

2.3.2. GRUVE Challenge

Similarly to the GIVE Challenge (2.3.1.), the GRUVE challenge [5] (Janarthanam and Lemon 2011) is also about a system generating instructions to the user which will provide landmark or important key points to help him find his way to the goal. But unlike the GIVE Challenge, GRUVE challenge will be placed in a real environment so the instructions will refer to real landmarks and others real entities to guide the user to the correct destination.

This is something hard to realize, even when using GPS, the location of the user is not completely precise which means it is impossible to know where he is exactly and it will be even harder to know which direction he is facing. So without knowing where the user is looking, it is hard to know if he can see a landmark from his position. So the main goal of the challenge is to overcome those problems to create a good pedestrian navigation system usable in the real world.
2.3.3. Web-based pedestrian navigation system (for GRUVE challenge)

The main difference of this pedestrian navigation system from what is already existing is that the user won’t have to look at the screen when navigating [7] (Janarthanam et al. 2012).

The architecture of the application is described below.

![GRUVE Architecture](image)

Figure 14 “GRUVE Architecture” [7]

From what is seen on this picture, the Interaction Manager is the central part of the architecture. It is taking the input of the user from the Utterance Parser, his location from the Pedestrian Tracker then it is able to respond. The utterance generator is there to translate this information into words. The Pedestrian Tracker is working with Global Navigation Satellite Systems (GNSS). The City Model is a database with a lot of information about Edinburgh since it will be the town selected for the test. The Visibility Engine is used to determine what the user is supposed to see and then what landmark can be used to help the user find his way.

To test this application and see if it is working, a web-based game has been created [6] (Janarthanam et al. 2012). The evaluation architecture can be seen on the picture below.
The purpose of the game is to find a treasure. To reach the treasure, the user will have to find a series of clues, to find those clues they will have to interact with characters that will give them information about their location. After finding everything, the user will finally be able to reach the treasure, the player also has an energy gauge to put some challenge while playing.

To describe a little the architecture, the broker is used to connect the web server to the internet navigator of the client. The navigation system is the main part of this architecture but also one of the main part of the game. It will represent the player “buddy”, which is the system that will generate natural language and help the player move around the city. The TTS engine converts the text from the program to speech. The city model and city engine are the same as seen just before. The web client is on the user side, it represents what the user is seeing and his interaction with the game. Google Street view is used to create the user’s vision and the interaction panel represents what the user can do. For this game he can move around with the arrows and there are some buttons, working as shortcut, that work as some quick responses like “Yes”, “No” or “Repeat”. The user also has a textfield that replaces what the user is supposed to say; he will have to type his question on a keyboard during the game. For the Baseline system, since the system creates instruction with natural language, it gives instruction one by one till the user reaches the destination. For example, if the user has to continue walking on the street, the current instruction would be “continue walking straight on streetA”. Then when the user reach the turn he has to take, it is at this moment that the system will say...
“turn right on streetB”. So the baseline system is creating the route at the beginning but it is giving information to the user only for its current position.

This game is important for the project since it is the one the Android application will be based on. The main difference will be that the user won’t be navigating from google street view but he will have to move around himself and the phone will be his “buddy” to help him find his way and the treasure. The Android application will also use speech recognition in place of keyboard inputs to ask direction.

2.3.4. Other Location Based Games

There are a lot of other location based games; some are just games without any purpose apart from letting the users have fun.

The first one presented here is City Explorer [13] (Matyas et al. 2008). It is presented as a game with a purpose, so the player will not only have fun but they will be contributing by giving geospatial data for other applications. They wanted to create an application like this because they think that having such a huge numbers of people on the internet playing some game just for fun and without a purpose is, in a sense, kind of waste of potential so their game should be fun and it should contribute to something. This game is inspired by the board game Carcassonne. The game will be handled in a part of a town that is contained in a square; this square will then be divided in more squares. See the picture below and the map on the left to illustrate this example.

![City Explorer Representation](image)

**Figure 16 “City Explorer Representation” [13]**

A player will have to define a certain location to find before beginning the game (for example a church, shopping store or restaurant) then other players will have to find those
locations during the game, take a photo and upload it to a website. If this photo is validated because the location is good, a tile will be attributed to the team, the team having the more tiles in a square wins this little square. When the time is over, the team with the most squares wins the game. So since each location must be validated, this game can help localise a lot of landmarks in many different cities.

Another location based game that is popular on the internet is Geocaching [21]. This game consists of hiding a box often containing something non valuable somewhere in the world then uploading the coordinate (longitude and latitude) on the geocaching web site. After this step, other players will be able to search for the container with their GPS. This game has begun in 2000 and is still really played with a lot of containers hidden across the world [14](O’Hara 2008). This game is interesting since it is a location-based game that has become really popular. In the game, there are two variants to find a container. The first one is when a player hides a lot of different containers that will help the participants find the final container. The other variant is to solve a puzzle first that will give the coordinates for the cache. So the one who hides the item can become some sort of creator since they have to hide it and try to create some fun puzzles to make people want to play the game and find their “treasure”. It could be a reason why this game has become so popular.

There are also a lot of other location based games like Mobimissions [2](Grant et al. 2007) where a player takes some photos then puts a description with those photos. The description should be something like a challenge, a game or a question, and that will be the mission. So it is a really open ended game and players can do more or less whatever they want. There are still others theories being developed about this subject, for example some people are trying to create a framework for location based games with augmented reality [4] (Jacoba et al.).

2.4. Android

2.4.1. Overview

Android is a Linux-based operating system (OS) owned by Google. It is primarily designed for Touch-Screen mobiles devices like smartphones and tablets. Android is also open-source which allows it to be distributed by a lot of different manufacturers. This fact also leads to the popularity of this OS across the phone industry as what has been seen in (2.1.1).
To know Android programming better, some key points will be mentioned about its history. It is in 2005 that Google bought Android [1] (Gargenta 2011) but it was only in 2007 that Android become open sourced. In 2008, Android 1.0 was released; it will signify the beginning of Android on smartphone. In 2009, Android was beginning to be on a lot of different devices and a lot of new versions are released. What has made Android really popular was the fact that it was open sourced so a lot developer decided to try and develop something for Android.

This report will focus on the different framework and libraries available in Android that give a lot of different functions. In the application framework, there are a lot of java libraries built for Android. This lets the developer have access to most of the features inside the phone, like the contact list of the users, the Wi-Fi and different applications. There are also a lot of different APIs that have been created by Google and those APIs can be easily re-used everywhere.

2.4.2. Google API

APIs are tools that are really easy to use. Google made a good documentation on their website [23]. The API interesting for this project should be the one that can be linked to voice recognition. The first API is “Android.speech”, this API is used to recognize voice and translate it into text. It is easy to use, most of the time some functions just have to be called to realize the necessary action, and speech is not an exception. This API is also available for a good range of Android versions. This can be used from an API level 3, so one of the first version of Android, and all the different functions of this API are used below an API level 10, which approximately an Android version 2.0.

The other interesting API linked to the speech API seen previously is the “Android.speech.tts” API. This function is able to use the TTS included in an Android phone. So it will allow the program to create sound from text. Since it is part of the previous package, it is also a low level API and it can be used from Android version 1.6 to newer version.

The next features that would be interesting to use with this project would be the GPS. To interact with the GPS, the API that should be used is “Android.location”. This API has been implemented for a long time so the basic functions of this API are only level 1, this means that all Android systems can use this and all of the functions are available for API level 5 or more, which means for Android version 1.0 or more. This API can give the current location of the user easily and it also has some interesting features like a “proximity alert”, this function will trigger
the device if the user enters a certain area that has been decided with longitude, latitude and the radius.
3. Tools and Methodology

3.1. Tools

To realize the project, a number of different methodologies and tools are used. The first important point is that the project is developed in JAVA. To develop in this language, an integrated development environment is the best thing to use to program efficiently and easily, so Eclipse will be used. The program will also be an Android application, so it should support Android APIs. Eclipse is also a good choice for this requirement since it has got an Android plugin which gives an Android Emulator to test the program and some graphical interface when creating the design. Furthermore, this plugin can detect error while using functions from Android APIs.

The different Android APIs that will be used are speech, speech.tts and location. The first one is used to detect the user voice and translate it into text. The second one which is part of the first package is used to generate voice from text with the TTS in an Android phone. The last one is an API that allows a developer to interact with the GPS. The program will be tested with an Android Smartphone and sometimes with the Emulator included in Eclipse but it is easier and better to use a smartphone to test it more quickly and in real conditions.

The Evaluation will be done with the Web Browser version. It will be uploaded on a computer that will be used as a server. People will then be able to access it from the outside. The questionnaire will directly be implemented along the test and the answer will be registered in log files on the server.

3.2. Methodology

For this project, an iterative waterfall model [26] (FCLS 2006) will be used. An iterative process, contrary to a waterfall model, goes through every step in the process a lot of times during the development. So when both of those methods are combined, it just means that the iterative process will go through every step in the waterfall mode a lot of times during the development. Iterative processes are driven by risk, which means that the most challenging step of the project will be created first. With this, it is easier to make some changes earlier if there are some problems about it. The main foundation of the system also has to be created in the beginning to be able to test it. This methodology is really effective to advance the project.
globally step by step and to be able to see how the project is working all the time. The process can be illustrated by the figure below.

1. **Requirement Analysis**: This step consists of defining everything about the system. The main goal and future users should be clearly defined. Some questionnaire, survey and research can be used to define the requirements before the process begin.

2. **System Planning**: In this step, the project planning is defined. For example the Gantt Chart or PERT graph should be defined during this step.

3. **System Design**: The program itself is designed in this step. The different units in the system are decided, how and with who those parts are interacting with each other. Everything here must also respect what has been decided previously during the requirement analysis.

4. **Implementation of the system**: This is the technical part of the system. The coding begins here and everything is created in this step.

5. **Testing and Evaluation**: The system is tested in this step. The test should check if the program is working and delivering the expected results. The evaluation is the beta-test
phase and it will evaluate the program to see if it respects the requirements and if the program in general is good to use.

This method has some advantage for this project, since it allows people to follow more closely the development of the program and see every time if new functions are working. Also the foundation created will be more solid which will help create something that could directly be shown to other people.

The downside of this method is that it could be hard to go back to a previous step when something has to be improved or changed if this methodology is not used like it is intended to do, time could also be lost by trying to reiterate the steps every time.
4. Requirements Analysis

The requirement of the project will be divided by groups and themes. On each line it will be show if these requirements are mandatory or optional.

The different requirements of the project are listed below:

- **Voice recognition**: The user should be able to speak to the phone; this will be translated to text in the application. The user will also have to touch a button to be able to speak to the phone. *(Mandatory)*

- **Find the User location**: The user location should be found with the GPS activated. *(Mandatory)*

- **Generate a route to reach a destination**: two algorithms should be implemented to do this. *(Mandatory)*
  - The first algorithm is called the “Baseline” system, the one currently uses by the GRUVE Project. It gives simple instruction to help the user reach his destination. For example it will always tell simply “Continue and turn left when reaching Princes Street” and then only when the user will reach Princes Street, a new instruction will be generated. *(Mandatory)*
  - The “Summary” system should be more complex and gives more information. For example it could give an overview of the route before giving any instructions: “The distance to the destination is two kilometres, you will have to pass through 5 streets, first you will have to go through Princes Street then turn right to Charlotte Street [ ... ] For now continue and turn left when reaching Princes Street” *(Mandatory)*

- **Program is working as a game**: Different features should be included to make the application feel like a game *(Mandatory)*
  - The player could have some statistics, for this game it is an energy gauge and some money. When the player moves around in the street, his energy could diminish till it drops to 0. When the energy has fallen to 0, it is a game over. The money will serve to buy information or others thing from characters in game. *(Optional)*
The interface should also show the number of keys collected for the player. *(MANDATORY)*

The game should signal when a player appears close to a character, for example the phone could be vibrating to show that something happened. The phone will know if the player is close to the position of the game character with the GPS position. *(MANDATORY)*

The character will then give information to the player on the screen *(MANDATORY)*

The image of the character should also appear on the screen with the text. *(OPTIONAL)*

**Summary Algorithm:** The new Summary algorithm must be created and at least working on the web browser version *(MANDATORY)*

- The summary should give the user an estimate distance and time to reach destination *(MANDATORY)*
- The summary incorporates the number of turns the user will have to take to reach destination *(MANDATORY)*
- The Summary algorithm will use landmark when giving directions *(OPTIONAL)*

**A test will be conducted to see the performance of the new Summary algorithm:** The test will check if the summary is appreciated by the user, a point of comparison will be used with the Baseline algorithm. *(MANDATORY)*

- The test will use approximately 30 persons from outside the project *(MANDATORY)*
- All the participants will test both algorithms with the help of two games; each algorithm will be randomly assigned to each game. *(MANDATORY)*
- The candidates will have to answer a questionnaire when the test is over. *(MANDATORY)*

**The results of the test will be analysed:** The result will have to be analysed. The report should for example highlight if the game was good, which algorithm has been more
appreciated, if they have been able to complete the game and if the guidance was good. *(MANDATORY)*
5. Professional, legal, ethical, and social issues

During the project a test will be conducted with candidates from outside the project. The results and feedback are taken anonymously so no leaks of personal information are possible and there will be data protection. The test will be done online and candidates will have to consent at the beginning that the data obtained from them will be used for the test and that they can withdraw from it at any moment. When the research report is completed, the candidate shall receive a copy of it via e-mail to see how their data have been used.

Concerning the development of the project, some software will be used and none of them are illegal. The development of the project will be done with Eclipse which is a free IDE and the smartphone use for the test is my own Android smartphone.

The figure below shows the terms users have to agree to.

Questions

Email Id: email@gmail.com

Terms

I give consent for this data to be used for research purposes. I understand that I can withdraw from this experiment at any time.

By checking the following check box, you agree to our terms. ☐

Figure 18 “Terms of agreement”

At the beginning the choice was not to use the web browser system to realize the test but to use the smartphone application to test the summary algorithm by walking around the streets. However we decided to use the web browser. There were several reasons for this choice. The first reason was that it was dangerous to walk around the street with a smartphone giving directions for a simple test like this. If some accidents would happen, it would have been a real problem for us. The second point was that it would have taken a really long time to test this. If we needed 20 persons, it would have been necessary to each time lend them the phone then they would have to walk around the streets to realize the test. It
could have been very long. So in the end, the web browser seemed to be the safest choice but also the one where it would be easier to find people doing the test to get more results.
6. Project Plan

6.1. Overview

The main aim of this project is to realize a pedestrian navigation system inspired from the GRUVE project. The two main focuses is to put the program on an Android phone and to develop the summary system. To test the algorithm, the program will be implemented as a game.

The stakeholders of the projects are pedestrian wanting to navigate in a town. So it could be tourist or even some people living in the town. The other group of stakeholders could be people interested in natural language generation and wanting to see the application of the project and also the results of the Summary algorithm against the Baseline algorithm.

6.2. Project Schedule

The project was divided into 4 main parts. The first part is the design and implementation part, so it was focusing on defining the project then coding it. For this stage, the main focus was to translate the GRUVE project to an Android phone and create the summary algorithm. So the first focus was to understand what has already been created for the GRUVE project, then what could be reused from this and what had to be changed or adapted to fit an Android application. The Baseline system also had to be understood to create the Summary algorithm. The next step was to design and create the GUI. It had to be intuitive and easy to use for a normal user. At the same time, the program had to be implemented. The summary algorithm had to also be created during this time frame; it had to at least be implemented on the web browser version for an easy testing.

When each implementation was finished, they had to be linked together. Then a testing phase took place where the different bugs had to be resolved and the software could be upgraded, it could be stronger and more robust if there was still some part possible to improve and some time remaining.

The third part was about the evaluation. The main focus on the project was to organize the test then find people to participate in the test.
The last part of the project was to organize and analyse the results. First the results had to be organized and separated to make it easier to analyse. After the analysis is finished, the conclusion can be draw and the difference between the two algorithms can be highlighted.

So below on figure 19 is the Gantt Chart illustrating the project.
Figure 19: Gantt chart of the project
### 6.3. Risk

The table below presents the different risks that could be encountered in the project.

<table>
<thead>
<tr>
<th>Risk</th>
<th>Impact</th>
<th>Likelihood</th>
<th>Contingency Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRUVE project not adaptable for an Android application</td>
<td>High</td>
<td>Low</td>
<td>If the problem should occur, only some part of the algorithm should be changed to resolve this problem</td>
</tr>
<tr>
<td>Android API not suitable for the GRUVE project</td>
<td>High</td>
<td>Low</td>
<td>The other plan will be to use what is already used in the GRUVE project</td>
</tr>
<tr>
<td>Summary Algorithm not compatible with the web browser</td>
<td>High</td>
<td>Low</td>
<td>If it should happen, the summary algorithm should be remodeled to be usable in a web browser or the web browser interface should change to accommodate it</td>
</tr>
<tr>
<td>Impossibility to work due to illness or any other reasons</td>
<td>Medium</td>
<td>Low</td>
<td>If it occurs, the project should not be late on schedule and it should have some float time</td>
</tr>
<tr>
<td>Any devices used for the project are broken or malfunction</td>
<td>High</td>
<td>Low</td>
<td>Should try to use a replacement for the time being or having free time waiting for the repairs</td>
</tr>
<tr>
<td>Supervisor unavailable due to sickness or any other reasons</td>
<td>Medium</td>
<td>Medium</td>
<td>Should try to do some other things that does not need the supervisor intervention</td>
</tr>
<tr>
<td>Schedule not realistic</td>
<td>Medium</td>
<td>Medium</td>
<td>Should try to not rely too much on schedule and try to do everything early</td>
</tr>
<tr>
<td>Any task taking longer than expected</td>
<td>Medium</td>
<td>Medium</td>
<td>Should try to have more free time to catch up</td>
</tr>
</tbody>
</table>
7. Implementations

7.1. Technical Decisions

The goal of the project is to create an Android application based on the GRUVE project. The GRUVE system was created using JAVA and Javascript, so for commodity purpose, JAVA has been chosen to continue with the same language. This way, it would be easier to modify part of the GRUVE system if necessary and JAVA is also the most used language for Android programming.

The next step of the project was to determine what to use for language recognition, it would have been possible to use something different but I went with what was already available for Android since one of the goal of the project was to develop this program using Android tools.

7.2. Organisation of the GRUVE system (baseline version)

Before going into details about how the Android application has been developed. A summary about how the GRUVE system is already working and how it is organized could be interesting to determine what was necessary to change and add to make the Android version work. The interface and content of the GRUVE project has been explained previously in 2.3.2 and 2.3.3.

The algorithm is based on route-plan nodes. Each intersection and other part of a street are divided into nodes. The different nodes are linked together so it is easy to see from there how the street are connected together, the route is also planned directly from this network of nodes.

To go into more details about the program itself, the program is mostly coded in Javascript and JAVA. The Javascript part is taking care of the client side (so what is going on the web browser) and it is the link to the system. The system is for its part coded in JAVA. Here is a UML showing an organisation of the java files of the system which is the interesting part of the system to see how instructions are generated.
To define the process a little further, the part getting the input from the client is “GruveServlet”. From this class, the server gets all the necessary data. Those are the user e-mail (which is used as an ID), his position, his orientation and the request, for example that he wants to go at a certain location or that he wants the system to give an update on his location. Those data will be stored in a JSONObject, which is an unordered collection of value in pairs. With this object, it is easy to store the data and pass it to other classes. Once the JSONObject is created and filled with all the necessary data, GruveServlet can send the information to GruveIM which will generate the answer of the buddy. Some other class like LogWriter are only used to create text files to store the information about what happened. From the information send by GruveServlet, GruveIM will determine what utterance it will need to create to answer the client.
Generating Route Instructions with Android Devices based on the GRUVE Challenge

GruveIM will contact the NLG class which will return the utterance. Only if the program wants an update of direction, then HWUNLG is called from NLG. HWUNLG will check the route to see the next step for the user. According to the user orientation, it will also be approximately able to tell the user if he will have to turn left or right or even turn around and go back. GruveIM will then have enough information to create the utterance for the buddy. The utterance will be sent back to the web browser and shown on the screen and/or voiced by the computer. More details about the algorithm itself will be seen later with the creation of the Summary algorithm.

7.3. Android Program Design

In this section, it will be presented how the application has been designed. This application is supporting a lot of different Android version, till API 8, so fairly old device should be able to make this application work. As explained previously, the system is giving directions to the user to reach a certain destination. The Android application is just the interface for the user; it is in a way replacing the web browser of the GRUVE project. The utterance is coming from an online server. The game of the GRUVE project is completely implemented inside the application for testing purpose. Now, to concretely begin, here is the screenshot of the “title” screen.

Figure 21 "Title screen of the Android application"
This is the first screen that the user will be able to see when he will launch the program. An ID is necessary to be able to begin using the program; in this case the email address is use. The ID could be used to remember what the user did before and adapt the system answers by using this knowledge. For example, if the user has been going a lot of time at a certain location, the system will know the user already has knowledge about how to go to this location. Let us say that this location is George Street. If the user is asking to go to Princes Street, the system could simply tell the user to go to George Street without going into the details of the route to this place. Then, when the user reaches George Street, the system will speak in more detail telling him for example: “Now turn left on Hanover Street to reach Princes Street”. Unfortunately, there was not enough time to implement a feature like this, but the ID stay here this feature should be implemented in a future version.

To facilitate the navigation, the last email used to login will directly be remembered by the phone, so the user will not have to take a lot of time to type his email every time he wants to use this program. For the button, as the name implies, start is to launch the program and instruction takes the user to a new screen showing the detail about how to use the application, so what the user can do and the details of the game. If the manual textbox is checked, the program will launch in manual mode and not voice mode. More details will be seen about each mode later.
Figure 22 “Main screen: beginning”

Figure 22 is showing the first screen that the user see when going into the game in voiced or manual mode, both are similar. As can be seen on the top, it is always possible to switch to the other mode on the go. However at the beginning, there won’t be any differences, the button “connect to server” has to be clicked first for change to happen. When this button is clicked, the connection to the server is first made and an utterance is brought back to the user saying: “Hi. I am your buddy. I can help you with directions”. This sentence shows the connection has been successful and the server is ready to give directions. In voice mode, the text of this button will simply change to “Ask for direction” now. The user will then have to talk and ask the system for the things he wants. Let us take a look at the manual mode. This way, it is easier to see what the user can also ask to the system in voice mode.
As can be seen on the Figure 23, this is the manual mode so the user doesn’t have to talk to the system but can access everything by clicking on buttons. However, the system’s answer will always be done by sound and not by text so the user always has to listen. The main purpose of this application is to indicate the direction to a certain destination; this is what the buttons on the first line are used for. Since just some streets are available on manual mode, this part is mostly used only for testing and those are also the streets used for the GRUVE Game. The streets are presented in a drop-down list, if the program has to be implemented for a more general use, this could be easily replaced by a text view and work the same way but for the sake of testing, a drop-down list is easier and quicker to use.

The information from the drop-down list is sent when the user click on the send button. The system will then answer by saying that he is looking for a route. A cycle will then be launched inside the application to call the server approximately every fifteen seconds. If the user is in a hurry and wants quickly a new instruction, then the button “Which direction now?” can be used to ask for an update about the current route. The button “Stop directions” is used to make the system stop the cycle calling the server. “Ok” is a “useless” button that is used to thank the system. The system will simply answer by saying that he is happy to help. “Repeat Direction” is used to ask the system to repeat the last direction he gave to the user. Those are all the options available from the system. In voice mode, the user has to ask all of this by
Generating Route Instructions with Android Devices based on the GRUVE Challenge

talking. For example, if the user wants the system to repeat the last information, he can simply say: “Can you repeat what you have just said?” and the system should repeat the last information. If the user says something wrong or something that was not recognized, the system will simply answer that it does not understand and to repeat what the user wanted to say. More explanations about how it is implemented are seen in the next chapter.

About the game used to test this application, it is the same as the one explained in (2.3.3) but adapted for an Android device. The game is staying mostly similar; the user has to get a certain number of clues and keys to get a treasure. So the characters or items are this time appearing on the screen when the user is in a specific location, like in the example below.

![Figure 24 “Main screen: Voice Mode with Character”](image)

When the user has reached a certain area, the character will directly appear on the screen with the text he is saying just below him. The phone will also vibrate to signal that something new appeared on the screen. On this example, it is the pirate presenting the game in the beginning. If it was a key to get or chest to open, an interaction button would have also appear to interact with the item so respectively taking the key or opening the chest for example. If a key is taken a little key picture would have appear just at the left of the manual button.
That is all for a summary of the application design, now the next chapter will go into more details about how the application has been implemented.

### 7.4. Android Program Implementation

#### 7.4.1. General Description

As it has been shown previously the program is divided in 3 screens, with mostly just one screen handling everything. The program can be divided this way, as can be seen on the class diagram below.
Figure 25 “Android Application Class Diagram”
The other class present apart from the three screens is NPC that is used to represent characters or items for the game. Title, Instructions and GameScreen are working similarly.

```java
@override
protected void onCreate(Bundle savedInstanceState) {
```

This method has to be implemented on each screen/Android class. It is inside this method that the program will know which layout he has to use to create the design. It is also easy to initialize everything in this function. As seen on the UML, Title is mostly there to link to the instructions or the game itself. The Instructions class is just a wall of text on a screen so after reading it, the user can just go back to the Title by clicking on a button. Those classes are not complicated. The NPC class is simply a class to store some values for each item/characters. Inside, you can find the position of the NPC, his image, text and visibility. Now the biggest and main part of the program is found inside the GameScreen class.

7.4.2. Location

One of the main functions that the program should be able to do is getting the user location but also update it as he is walking around the city. To do that, the Google location API has been used. Firstly, the program is checking if the user has activated an internet connection and the GPS, if not, a warning box will appear asking the user to activate those functions. The location API can now be used and activated, another function is created to update the user location after a certain time and certain distance is reached. For this application, it was decided to update every meters after 5 seconds.

The only thing that is not obtained with this API is the user orientation. This is calculated with the accelerometer and the magnetic fields available on the Android device. With those tools, it is possible to get an angle in reference to the North Pole. Figure 25 illustrates this angle with the main difference being that the angle of my program is from -180° to 180° and not 360°. So 315° on the figure would represent -45°.
7.4.3. Voice recognition + TTS

For both of those functions, the Android API is also used. Both are easy to use and implement. For the voice recognition, the user simply has to click on a button to call the function that will translate what the user is saying to a string. When the String is obtained with the program, a parser is used to determine what the user was trying to say. Since there are not a lot of options, the parser is not too sophisticated but it is still using regular expressions like this example.

```java
suggestedWords.get(i).matches(".*how can I go to.*")
```

This regular expression is used to determine that the user is asking for a direction and what is say after the “to” should be the destination. For some of those demands like asking direction, update on direction or repeating information, the system has to connect to the server. For the rest, the answer is always the same so there is no need to ask the server. The information needed to send to the server is stored in a simple string. The necessary data are similar to what has been presented in (7.2) but this time not through a JSONObject but a String for an easier transfer from the phone to the server. More information about what has been changed in the server is presented later. But for this part, once the utterance is obtained from the server, the string utterance is simply transferred to the phone TTS so that the user can hear the answer.

The only problem with this method was to know when the answer is finished before putting something else through the TTS. If there was no check, the new message will cut the
old one without letting him finish what it was saying. To avoid this problem, a loop is done for checking if the TTS is finished before putting another message.

### 7.4.4. GRUVE implementation

The game is directly implemented inside the Android application. The server side is still doing the same thing; just the class doing the interaction with the client is different. First let us take a look at the Android integration.

This is really easy to implement, as seen previously the characters are created with the NPC class. At the creation of the program, all the characters and items are initialized. A function to check if the user is close to a character or an item is called whenever the location is updating. This function is checking if the user is within a certain radius to the character or item. To calculate this, latitude and longitude have to be translated to a distance since they cannot be used as they are since earth is not a surface plane. The formula chosen is called “spherical law of cosines” As seen on Movable-Type [30], the formula is written like this

\[ d = \text{acos} ( \sin(\phi_1).\sin(\phi_2) + \cos(\phi_1).\cos(\phi_2).\cos(\Delta\lambda) ).R \]

“where: \( \phi = \) latitude, \( \lambda = \) longitude

\[ R = \text{radius of earth} \]

\[ d = \text{distance between the points} \]

Then it is easy to know if the object is within distance of the user by comparing this distance to the minimum radius wanted. That is all for what is inside the Android application.

Now for the server part, the only class that needed some change was GruveServlet as seen on (7.2). This time a string is coming from the Android device and not a JSON Object. The string has the different information separated by a “/”, it is then easy to divide everything from the “/” to separate all the information. When the data are divided like wanted, the server is working similarly like what was explained on 8.2. The new information is stocked into a JSON Object for creating the utterance that is send back to the phone.

### 7.5. Summary algorithm design and implementation

While the implementation of the GRUVE project on Android was a main task, it was not the only goal. A second algorithm also had to be created; it had to generate language based on
the “Baseline” algorithm. This new “Summary” algorithm should have the same purpose as the other one but it should do it in a different way.

First, let us see how the “Baseline” algorithm is working. This algorithm is completely based on nodes. Every time you can find a corner in a street, a node should be there. So this node is representing intersection between streets. However node could also be placed randomly in some street if the street is too long. A database is containing all the information about the nodes, their position and also to which nodes they are connected. From there it is easy to find a route; you just have to connect the nodes to find a link between the current user positions to the desired node with the shortest distance. This is a simple and quick way to summarize the principle of this method. The “Baseline” algorithm just has to indicate how to go to the next node according to the current user position. To calculate the angle, the azimuth has to be first obtained, like what has been explained previously on (7.4.4) then with another formula, it is possible to obtain the user’s bearing. For example, it is possible to compare the current azimuth of the user to his previous one, this way the bearing can be obtained. With the bearing available it is then easy to know where the user has to turn. Here are some examples output that can be obtained from the Baseline system. If the user has to go straight on the street Potterow, the system will be saying “Continue walking on Potterrow”. If the user has to turn left on West Nicholson Street, the system will be saying “Turn left onto West Nicolson Street”. Another example would be if the user has to turn around, the system will be saying “Turn around and continue walking straight”.

The “Summary” algorithm will have to do different things. This time, a summary of the route will be given to the user at the beginning, so each time the user is asking for a new location. This summary will give the user the distance “as the crow flies”, the approximate time to reach the destination based on an average walking speed, in this case, approximately 84 meters per minutes. The summary will then provide the number of turns the user will have to do. The turn will only be counted if the user is turning while there are other ways. For example, if a node is placed in the middle of a long street, the system doesn’t have to check if there is a turn and which kind of turn, this node is only connected to the previous node (beginning of the street) and the next node (end of the street), so if there was a turn, the node would have to be connected to three or more nodes. Furthermore, if there is a turn but the user will still be going in the same street; this is not considered as a turn. Then the system
provides more detail, like how many of those are left turns and how many of those were right
turns. So each time the system was finding a turn, it had to differentiate left to right turn. The
program will then show where those left and right turn are taking place. For example the
system could say something like this: "turn right onto Princes Street, then turn left onto South
Charlotte Street". However, if the route is too big, it will be annoying to get the description of
the whole route, in this case, the user will just have the number of left and right turns. This
case will happen if there are more than 5 turns. After this, the system will just give the next
direction just like the original algorithm.

This summary will be updated after some time to give information about what is left to do
on the route. It will be exactly the same as what has been explained previously. If there were
also a lot of turns at first and now for the update there are less than 5 turns, the user will now
have the description of what are the turns he still has to do.

The summary algorithm will also be using landmarks when possible, so for example, when
the system will say: “Turn left onto Princes Street”, if there is a restaurant named “XYZ” nearby,
the system will be able to say “Turn left at the restaurant XYZ”. Landmarks will be used
whenever they are available; this will take priority over street names, even for the summary
description. If there are no landmarks, the system will of course use street names.

Here are some concrete examples showing what the system could be saying to present the
whole summary. Here is an example when the user is asking how to go to Nicholson Street
from Potterow: “Your destination, Nicolson Street, is 283 metres away, a little more than 3
minutes walk. You will have 1 left turn. So you will have to turn left onto West Crosscauseway.
For now, continue walking on Potterrow.” Now is another example when the user is asking
how to go to South College Street from Crichton Street: “Your destination, South College Street,
is 288 metres away, a little more than 3 minutes walk. You will have 4 turns before reaching
the destination. To be exact, 3 left turns and 1 right turn. So you will have to first turn left onto
Potterrow, then turn right onto Marshall Street, then turn left at Beruit, then turn left at
Brassfounders Column. For now, turn left onto Potterrow.”

Now for the summary implementation, some files from the GRUVE system were modified.
The files in question are “HWUNLG”, “NLG” and “GruveIM”. They are working exactly as what
was explained on (7.2), but most of times new functions and variables have been added.
“HWUNLG” is where most of the algorithm implementation went to. Here is a diagram showing what is inside “HWUNLG”.

<table>
<thead>
<tr>
<th>HWUNLG</th>
</tr>
</thead>
<tbody>
<tr>
<td>- om: HWCityModel</td>
</tr>
<tr>
<td>- um: UserModel</td>
</tr>
<tr>
<td>- turn: int</td>
</tr>
<tr>
<td>- type: String</td>
</tr>
</tbody>
</table>

```java
+ HWUNLG(UserModel userModel, HWCityModel cityModel)
  + getTurn(): int
  + presentRoute( JSONObject devs): String
  + getRouteDetails( JSONObject dev): String
  + getClosestEntityPosition(nextCoord): String
```

**Figure 27 “HWUNLG description”**

The only function that was already there before was `presentRoute` which is used to show the next immediate direction. This time `getRouteDetails` has to loop through all the nodes to count the number of turns and get the next direction at those turns. The utterance will then be completed and `GruveIM` will be able to recreate the summary utterance from those functions.

Some of the problems encountered when creating the summary was to know if the user was turning left or right. To find this, there is first a function that is checking if the node is linked to multiple nodes, if not, it can be already decided that this node is not a turn so the user can just go straight. Then a function is checking to see if the user is changing to another street name, if not, he is obviously on the same street so he will have to continue following the street. If both of those conditions are true, the only thing left is to determine the angle, so if the user is turning left, right or going straight. To determine the angle, the azimuth will be used again (8.4.2). The previous azimuth will always be kept in memory and it will be compared to the current azimuth, by comparing both of them, it is easy to determine where the user will be turning.
8. Evaluations

This part will describe the experiment for this project, show the results and conclude about what can be obtained from those results.

8.1. Method

8.1.1. Designs

The goal of the experiment is to test the new Summary algorithm against the Baseline algorithm. As can be seen previously, some landmarks have also been added to the system with the summary. We have decided to simply put a focus on testing the summary, it would have been too complicated and too time consuming to prepare another experiment for testing if people like street name or landmark. The labs in Heriot-Watt also already tested that people in general prefers landmarks over street names so it was not a really interesting thing to test.

The second point is also that the Android application won’t be used during the experiment; the test will be done directly from the web-browser version. It would have been interesting to try testing the Android application in real life but it was too time consuming to get good results since it would imply make a lot of people try this application by walking around in Edinburgh, so each person would take a long time to finish their test. It would also have been a little dangerous to walk in the city, so we have finally decided to focus on testing the algorithm through the web browser version.

For simplicity of spreading the test, the experiment has been hosted on a computer in the computer science department of Heriot-Watt. A port has been opened to allow access on this computer from the outside. This way a simple Internet link could be given to a participant allowing him to participate easily. The questionnaires of the test have also to be answered directly online. For the ethical part, the user must check a box at the beginning saying that he is agreeing taking part in a test and using his data for research purposes.

8.1.2. Participants

The experiment was uploaded on a server, so everyone can participate with only an Internet link. There were no special criteria for choosing people. The person just had to fill some information before beginning the experiment. They had to put their age, if they are a native English speaker, how many hours they are playing game per week, if they are familiar with Google Street View and finally if they are familiar with the city of Edinburgh.
8.1.3. Procedure

The experiment was divided into two game worlds. Each game was working essentially the same as the GRUVE Project (2.3.3). The figure 28 below is a picture of what the user is seeing while playing the game.

![Figure 28 “Evaluation Screenshot”](image)

The user has a buddy (the system giving instructions) with him to help find a route. To communicate with the buddy, the user can choose a street from a dropdown box then ask for directions by clicking on the “Send” button. After this, the server will update with new instructions as the user is moving around. If the user does not want to wait for the buddy to respond he can ask for an update by clicking on the “Which direction now?” button. If he wants to hear again what the system was just saying, he can click on “Repeat Instruction”. “Stop Directions” is used to stop the system giving new updates. The “Ok” button is useless and is just used to acknowledge what the system is saying, so it doesn’t have any real meaning. From what has been discussed previously the two algorithms are using landmarks for giving directions. If the user is sometimes lost with the landmarks, he can use the “Help” button to change the current instruction to use street name, this may sometimes be easier for some people.

The game has always the same procedure, first the user has to talk to a character that will welcome the player and tell him to go somewhere to get a clue by talking to someone. Then the player will have to use the buddy to find this place. At this new place the player will get
some more clues till he is able to find the key to get the final chest. When the chest is open, the game is over. There are two games for this experiment, every time the player is put in the city of Edinburgh but he will not go through the same places and he will also not get the same clues. On each game, one of the two systems is randomly assigned to the player, so he may begin on world 1 with the “Summary” system then he will be in world 2 with the “Baseline” system but another player could begin on world 2 with the “Summary” system then world 1 with the “Baseline” system. The second game will always be the inverse of the first one, so the other game world and system. This is a balanced designed within subjects.

The two games have also been created with different mind-set. There is one game with different clues really close to each other while the other one has different clues really far away from each other. This way, it will be easier to see if there are some changes of opinion between the two systems for short or long route.

Every time the user finishes a game, he will be taken to a questionnaire. There, the user will have five questions. For each questions, he can answer in a four points Likert scale, going from disagreement to agreement. The first question is if the buddy has been helpful, the second one is if the buddy’s instructions were easy to understand, the next one asked if the buddy’s instructions were accurate, then if the instructions were of the right length then the last question ask if the player already knew this neighbourhood of Edinburgh, for this question there are only two answers, “yes” or “no”. Finally there is a text box if the user wants to put additional comments. Since there are two games, each participant will have to answer those questions two times.

As can be guessed from those questions, we are implicitly trying to see if the user prefers the Summary system over the Baseline system. All of those questions are asking this question from different angles, to see if a participant could prefer some part of the summary system over others.

The two figures below are showing what each user has to fill for this evaluation. Figure 29 is showing what profile information participants had to fill at the beginning of the test then Figure 30 is showing the questionnaire that each participant had to fill each time they were finishing a game.
Please answer a few questions about yourself

1. Your age: 

2. Are you a native speaker of English? (Y/N)
   ○ Yes ○ No

3. How many hours do you spend playing computer games per week? 

4. Are you familiar with how Google Streetview works?
   ○ Yes. I have used it a lot ○ A little ○ No, not at all

5. How would you rate your knowledge of streets in Edinburgh in general?
   ○ Familiar ○ Not familiar

Figure 29 “Profile information”

What did you think about your buddy who helped you with navigation instructions.

Please rate your agreement with the following statements about the buddy (who gave you directions) assigned to you.

1. The buddy helped me to complete the tasks in the game:
   ○ Agree ○ Slightly Agree ○ Slightly Disagree ○ Disagree

2. The buddy’s instructions were easy to understand:
   ○ Agree ○ Slightly Agree ○ Slightly Disagree ○ Disagree

3. The buddy’s instructions were accurate:
   ○ Agree ○ Slightly Agree ○ Slightly Disagree ○ Disagree

4. The buddy’s instructions were of the right length:
   ○ Agree ○ Slightly Agree ○ Slightly Disagree ○ Disagree

5. How would you rate your knowledge of the streets in the gameworld (the neighbourhood) before you played the game?
   ○ Familiar ○ Not familiar

6. Were you able to hear the buddy speak or did you just read off the navigation instructions from the screen?
   ○ Yes, I heard the buddy ○ No. Just read the instructions from the screen

7. Any other comments about your buddy:

   

Figure 30 “System Questions”
8.2. Results

At the end of the test, there were more than 30 participants, however some people did not fill both questionnaires and other people did not even fill one of them so they will not be counted in the statistics. For this analysis, there were 23 people that tested both systems.

The game has asked participants to fill some questionnaires at the end of each game, but some data has also been obtained directly when they were playing the game. For example it is possible to see if each participant has finished the game and how long they took to do it.

8.2.1. Task Completion

The figure 29 below is showing different statistics showing how many finished one or both games.

![Figure 31 “Task completion”](image)

As can be seen on the graph, approximately half participants completed both games with the two systems and 65% completed only one game, so they were able to finish with only one system. So the majority of people that gave up were people that could not do even one game.

There seems to be others differences since people were able to complete more easily the Baseline system (69%) compared to the Summary system (only 56%). To determine if those data were relevant and if there was a significant difference, a Mc Nemar’s test has been
applied to those data. The p value obtained from this test is 0.250 which is higher than 0.05, so the difference is not significant.

8.2.2. Time completion

Figure 32 “Time completion”

Figure 30 is showing the average completion time taken to finish the game with the Baseline system or the Summary system. From those data, it seems people took approximately two more minutes to finish a game with the Baseline System than one with the Summary system. To see if the difference is significant, a t-test is applied on the data. The p value obtained from the data is 0.53, higher than 0.05. Those data do not have a significant difference.
8.2.3. System usefulness

Figure 33 “Average score for system usefulness”

Figure 31 is showing the average score for the system usefulness. It seems most participants preferred the Baseline system compared to the Summary system. Even though both got good grades, since they are between 3 and 4. For a reminder 4 is representing that the user is agreeing that the system instructions were useful while 1 is showing disagreement implying that the system did not help at all. A Wilcoxon test has been applied to those data to see if the difference is significant. The p value obtained is 0.024 which is lower than 0.05. The difference is significant here.

8.2.4. Ease understanding of the instructions

Figure 34 “Average score for ease understanding of the instructions”
Just like (8.2.3), this is using the same Likert scale. This time the higher the score is the easier it is to understand the instructions. Participants found the Baseline system to be easier to understand than the Summary system, however both got good ratings, higher than 3. Just like before, the Baseline system seems to be a little more appreciated however compared to (8.2.3), the difference between the two systems is not as big as previously, only a 0.2 differences. A Wilcoxon test is also applied on those data to see if the results have significant differences. This time the p value obtained is 0.190 which is higher than 0.05 so the difference is not significant.

8.2.5. **Accuracy of instructions**

![Average score for the accuracy of the instructions](image)

This time, the two ratings are really similar, so most participants found the instructions mostly accurate with a little better accuracy for the baseline system. The Wilcoxon test is also done here and the p value obtained is 0.180. Here again the difference is not significant.
8.2.6. Length of the instructions

![Figure 36 “Average score for the length of the instructions”](image)

This time, the difference is bigger between the two ratings. Most people seem to prefer the instruction’s length of the Baseline system which got a rating of 3.35 compared to the Summary instructions which only got 2.74. People found the length of the instructions to be at the right length when the ratings are high. Since the summary system got an average between 2 and 3 most people got a mixed opinion about the length. A Wilcoxon test is done again to get the p value; it is at 0.015 so the difference is significant.

8.3. Discussions

From what has been obtained in the previous data, it can be seen that the Baseline system is more liked than the Summary system according to the participants. The Wilcoxon test shows that only for the length of the instructions and the usefulness of the system were those differences really significant. However all of the results for the summary system were still at an acceptable level, so people thought that the system was still helpful and it was still easy to understand the instructions. The only problem was the length of the instructions; most participants must have found it too long and it is because of this that they may have found the system a little less helpful. It may also have lowered their appreciation of the Summary system in general.
However, in the end both algorithms were well received, the average score for all the questions were between 3 and 4 apart from the length of the Summary algorithm which was a little under 3.

For all the participants that were not able to finish both games which represent approximately half the participants, here are some ideas about why they stopped. From the t-test it seems to be independent from the system (Baseline or Summary) so it should be something else. From other data, it seems some participants were lost so they have not been able to find their way back. Other participants may had some problems understanding how the game was working since they were not native speakers; other may have not read the help at the beginning, since some people commented that they did not understand what they had to do exactly or they changed the street instructions and forgot where they were going. Some people that only finished one game may also had passed too much time on the first one, so they did not had time to finish the second one. These were what could be deduced from seeing statistics about their game.

Some interesting comments were also obtained in the comment section. Some participants were wondering why the summary instructions were so long. They wrote that they had to go to each place anyway, so there was no need to really know the details of the route. This is a fair comment since everything was really scripted and there was no freedom in the game, however in the real world, people could wonder how long it would take to go to a certain place and how far it is. Knowing if the route is complicated or not could also help people deciding if they really want to go there by foot or they could take transport or even decide to go somewhere else. Another interesting comment available was someone thinking that the summary instructions were good but he always had to wait near the end of the instructions to know where to go. He thought that it would have been better to get the next step directly then the rest of the route as a summary as a bonus. This way, people that had to hurry could go more quickly to their destination. It could be interesting to change the Summary algorithm in the future with this idea. It could change the opinions of some people that like to go quickly to a destination.
9. Conclusions

In the end the project was successfully finished. The Android application is done and working even though only the Baseline system has been implemented, there should not be any problem for the Summary system but because of time constrained, there was not enough time to test this algorithm with the Android application. The Summary system has been implemented and it is working successfully. There were a lot of difficulties along the way, for example defining the summary algorithm, and then make it work. I also got a lot of problems while implementing new functions in my Android application. For example, I got a lot of problem updating the different information on the phone, like user location and orientation, while continuing sending and getting update from the server.

The main hypothesis of the project was that NLG Summary would be more beneficial for people compared to the Baseline NLG. The evaluation made here proved this hypothesis to be wrong, mostly with the question if the buddy was useful and if the instruction’s length were of the right size. For both of them the results were significant and people preferred the Baseline system. As what has been discussed previously, a different test may have given other results, like walking in real life compared to a computer a game may make people prefer the summary NLG.

For the future, some improvements could be made. The Android application has not been tested in the city. This could be tested to get people opinions and advices. From what has been obtained from the evaluation, a new test taking place in the city could be interesting for testing the Summary system. Maybe by moving around themselves and not playing a game, users could find the Summary system more interesting. The system could also interest more people if the objectives are not as precisely declared as what has been tested now, people would have to search and by knowing what the shortest route is with the Summary system, they could have other opinions. Since the text was displayed both time, it could be interesting to just leave the voice, maybe participants won’t feel overwhelmed by the size of the Summary system if they cannot see it. The system voice during the test was also saying every instructions and not cutting anything even when new instructions where coming. So if users were too quick, the system audio could be a little late compared to what the user was doing, this could be improved later by maybe adding a button to stop the system audio and make it...
skip everything to go back to the current instruction. Maybe this way people would prefer the Summary system.

There are also some ways that the Summary NLG could be improved. Maybe changing the order, putting the direction first then doing a summary. The system could also be more interactive, asking the user if he wants a full summary or part of it. The summary update could also simply be asked by the user, no need for the system to tell the update all the time after two minutes, this may have annoyed some participants. The summary could also change a little, no need to tell how many turns but maybe it could be using some key locations to tell the route, like for example to tell that the route will first go near the church X then go to the street Y and it will pass near the statue Z. Here are some ideas to improve the Summary system, but it could be improved in a lot of different ways. Participants were not reticent for having a Summary system from what they have answered in the questionnaires, so it may just be necessary to improve it to meet their expectations in a better way.
Bibliography


Generating Route Instructions with Android Devices based on the GRUVE Challenge


Appendix: important code

Summary Generation

```java
/**
 * Do a description of the route
 */
public String getRouteDetails(JSONObject da)
{
    ArrayList<String> reachNodes = new ArrayList<String>();
    Route route = (Route) da.get("route");
    Integer index = (Integer) da.get("index");
    Double co = (Double) da.get("currentUserOrientation");
    Double turnAngle;
    Double newAngle = 0.0; // Current User Angle
    Double oldAngle=0.0; // Previous Angle
    Position nextCoor;
    Position oldCoor = new Position (0,0);
    Position currentUserCoor = new Position((String) da.get("currentUserCoor"));

    Iterator<RouteElement> ire = route.getNodes().iterator();
    String wayName = "null";
    String previousName = "null";
    String entity="";
    String entityName="";
    String sysUtt = "first "; // prepare utterance
    String oldPosition = "";
    boolean start = false;
    boolean otherStreet = true;
    int left=0; // Number of left turns
    int right=0; // Number of right turns

    while (ire.hasNext()){
        RouteElement re = ire.next();
        if (re.getIndex() == index - 1 && start == false){
            wayName = re.getWayName();
            entity =
            getClosestEntity(cm.getCoordinates(re.getNodeId()));
        } else {
            if (re.getIndex() == index || start == true){
                boolean street = false;
                nextCoor = cm.getCoordinates(re.getNodeId());
                reachNodes =
                cm.getReachableNodes(re.getNodeId());

                //Check if it is the second turn or more
                if (start == false){
                    turnAngle =
                    Tools.convert180(Tools.getRelativeOrient(currentUserCoor, nextCoor, co));
                    newAngle =
                    Tools.getOrient180(currentUserCoor, nextCoor);
                } else {
```
// Compare the previous and current bearing to get the angle
newAngle = Tools.getOrient180(currentUserCoor, nextCoor);
turnAngle = Tools.convert180(newAngle - oldAngle);
}

if (entity.equals("null") || entity.equals(""))
    || entity.isEmpty()){
    street = true;
    entityName = wayName;
} else {
    entityName = entity;
}

boolean turningPoint = true;
if ((turnAngle > 0 && turnAngle < 30) ||
    (turnAngle >= 150 && turnAngle <= 180) ||
    (turnAngle < 0 && turnAngle > -30) ||
    (turnAngle <= -150 && turnAngle >= -180)) {
    turningPoint = false;
}

// Check if this is a node where there is more than one way
if (reachNodes.size() == 1) {
    turningPoint = false;
}

if (otherStreet == true && turningPoint) {
    if ((turnAngle >= 30 && turnAngle < 120) ||
        (turnAngle <= -120 && turnAngle > -150)) {
        right++;
        if (street) {
            sysUtt += "turn right onto " + entityName;
        } else {
            sysUtt += "turn right at " + entityName;
        }
    } else if ((turnAngle <= -30 &&
        turnAngle > -120) || (turnAngle >= 120 && turnAngle < 150)) {
        left++;
        if (street) {
            sysUtt += "turn left onto " + entityName;
        } else {
            sysUtt += "turn left at " + entityName;
        }
    }
}
otherStreet = false;
sysUtt += ", then ";
if (re.getWayName().equals(wayName) == false){
    otherStreet = true;
} else {
    otherStreet = false;
}

entity = getClosestEntity(cm.getCoordinates(re.getNodeId()));
wayName = re.getWayName();
start = true;
oldAngle = newAngle;
currentUserCoor = nextCoor; // Update coordinates
}

this.turn = left + right; // Number of turn

//Utterance for turn
String turnUtt="";

//Only one turn, so there is no "first"
if (left + right == 1){
    sysUtt = sysUtt.substring(6, sysUtt.length());
    turnUtt = "You will have ";
} else if (left > 0 || right > 0){
    turnUtt = "To be exact, ";
}

// Not plurals
if (left == 1)
    turnUtt += "1 left turn";
else if (left > 1)
    turnUtt += left + " left turns";
if (right > 0 && left > 0)
    turnUtt += " and ";
if (right == 1)
    turnUtt += "1 right turn";
else if (right > 1)
    turnUtt += right + " right turns";
if ((left + right) == 0){
    return ", then ";
}
else if (left + right < 5){
    sysUtt = sysUtt.substring(0, sysUtt.length()- 7);
    //Remove final "then"
    return turnUtt + ". So you will have to " + sysUtt + ".";
}
else {
    return turnUtt + ".";
Landmark finding

/**
 * Entity closest to the user
 */

public String getClosestEntity(Position nextCoor){
    String entity = "null";
    int loop = 0;
    Random random = new Random();

    while (entity == "null"){
        type="";
        int rand = random.nextInt(5); // Random type
        if (rand == 0){
            type = "poi";
        }
        else if (rand == 1){
            type = "museum";
        }
        else if (rand == 2){
            type = "chainStores";
        }
        else if (rand == 3){
            type = "atm";
        }
        else if (rand == 4){
            type = "restaurant";
        }
        entity= cm.getClosestAmenityLimit(nextCoor, type, 23);

        if (entity.equals("null") == false){
            if (entity.substring(0, 2).equals("a ")){
                entity = "null";
            }
        }
        loop++;
        //loop 6 times before finishing
        if (loop > 6){
            type = "";
            entity = "";
        }
    }

    return entity;
}