NTNU – Trondheim
Norwegian University of
Science and Technology

OsloTUC - Natural Language Bus Oracle for a new City

An Evaluation of BusTUC's Extensibility

Espen Jacobsson

Master of Science in Informatics
Submission date: May 2015
Supervisor: Bjørn Gamback, IDI
Co-supervisor: Rune Sætre, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

NTNU – Trondheim
Norwegian University of
Science and Technology

OsloTUC - Natural Language Bus Oracle for a new City

An Evaluation of BusTUC's Extensibility

Espen Jacobsson

Master of Science in Computer Science
Submission date: May, 2015
Supervisor: Björn Gambäck
Co-supervisor: Rune Sætre

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

The recent years have seen an increased demand for smart phones and other handheld devices. This gives producers of applications or content an excellent opportunity to reach out to a larger audience than before, and it gives them a chance to make their applications a part of people's everyday lives.

In this thesis, a well-established natural language travel planner for buses was modified. This was done in order to understand what is required when providing support for a new city. In 2013, Marius Qvam Wollamo added support for trams in Trondheim, and provided the travel planner with a map (Wollamo, 2013). His idea of using a map was an inspiration for this thesis, and the recent increase in the use of mobile devices was an encouragement to create a multi-platform application utilizing map features. A prototype of such an application was developed to try to meet the needs of users, whether they are in front of a desktop computer or on the move with their mobile devices. A survey was conducted with the aim to find out how users of such an application ideally want to query a travel planner, and how they want the returned information to be displayed. A questionnaire was created and distributed to a number of respondents who were asked to compare the new prototype with an existing travel planner. The survey showed that the prototype was well received, and that the displayed information gave varying, but overall positive results. The results of the questionnaire gave grounds for discussing the implemented features, and whether the application was accessible to users with disabilities. A tendency to favor natural language over filling in predefined boxes, as well as a preference to see sufficient metadata about the routes being suggested, was observed. Providing the ability to search for bus stop names in real time was found to be preferable with the users. The work done has shown that it is possible to fully transfer the functionality of the travel planner in one city to another, and that a mobile friendly interface is to be preferred.

Sammendrag

I de siste årene har vi sett en stor økning i etterspørselen av smarttelefoner og håndholdte enheter. Dette gir produsenter av applikasjoner og innhold en utmerket mulighet til å nå ut til et større publikum enn før, og gir dem muligheten til å gjøre disse applikasjonene til en del av folks hverdagsliv.

I denne avhandlingen er en veletablert reiseplanlegger for busser, basert på naturlig språk, blitt modifisert for å bedre forstå hva som må til for å flytte et slikt system til en ny by. Marius Qvam Wollamo la i 2013 til støtte for trikk i Trondheim, og utstyrte reiseplanleggeren med et kart(Wollamo, 2013). Idèen hans om å bruke et kart var en inspirasjon for denne avhandlingen, og den store økningen i bruken av mobile enheter de siste årene inspirerte også til å lage en multiplatformsapplikasjon som benytter en kartfunksjon. En prototype av en slik applikasjon ble utviklet for å forsøke å møte brukernes behov, enten de sitter foran en stasjonær datamaskin eller er på farten med en mobil enhet. En undersøkelse ble utført med et mål om å finne ut hvordan brukere av en slik applikasjon ideelt sett vil stille spørsmål til en reiseplanlegger, og hvordan de vil ha svarene presentert. En spørreundersøkelse ble laget og distribuert til et antall respondenter, som ble spurt om å sammenligne den nye prototypen med en allerede eksisterende reiseplanlegger. Undersøkelsen viste at prototypen ble vel mottatt, og at informasjonen som ble presentert ga varierende, men for det meste positive resultater. Resultatene av spørreundersøkelsen ga basis for å diskutere de implementerte funksjonene, og hvorvidt applikasjonen var tilgjengelig og brukervennlig for personer med funksjonsnedsettelse. Det ble observert en tendens til at respondentene foretrakk naturlig språk fremfor å fylle inn i predefinerte bokser når de skulle søke, samt at de ønsket å se tilstrekkelige mengder metadata om rutene som ble foreslått. Det ble også observert at muligheten til å søke etter bussholdeplasser i sanntid ble foretrukket av brukerne. Det utførte arbeidet viser at det er mulig å overføre all funksjonalitet for en reiseplanlegger i en by til en annen, og at en mobilvennlig løsning er å foretrekke.

Preface

This Master's Thesis describes the work done by Espen Jacobsson to fulfill the final part of his Master of Science (M.Sc) degree in Computer Science at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU). It is a contribution to the *Fremtidens Ultimate Intelligente Reiseopplysningssystem (FUIROS)* project.

Acknowledgements

I would like to thank my supervisors Björn Gambäck and Rune Sætre for their guidance and support, and for giving me the opportunity to work on a project I have been using for a greater part of my life. It has been truly interesting to be a part of the FUIROS project. I would like to extend my gratitude toward my wonderful friends Anja Sønstebyseter Rønning, Mari-Jeanne Schjetne and Juni Angelfoss, for their continued moral support. Also, I would like to thank my parents and my sister for always being there and for always believing in me. Finally, I would like to thank my girlfriend for encouraging me when I needed it the most.

"It's dangerous to go alone! Take this."

Espen Jacobsson
Trondheim, May 1, 2015

Contents

List of Figures

List of Tables

Acronyms

AJAX Asynchronous JavaScript and XML. 34

API Application Programming Interface. 11, 30, 33, 34, 39, 42, 68

AtB A til B. 10–12, 14, 15, 20

CBR Case Based Reasoning. 12

CNN Cable News Network. 3

CPU Central Processing Unit. 33

CSS Cascading Style Sheets. 20, 21, 34–36, 38

CSV Comma Separated Values. 22, 30

Difi Direktoratet for Forvaltning og IKT, The Norwegian Agency for Public Management and eGovernment. 67

DOM Document Object Model. 34, 35

FUIROS Fremtidens Ultimate Intelligente Reiseopplysningssystem. iii, 1, 2, 58, 65

GPS Global Positioning System. 10, 11, 68

GTFS General Transit Feed Specification. 18, 21, 22

GUI Graphical User Interface. 7, 9, 63, 65

HTML HyperText Markup Language. 34–36

HTTP HyperText Transfer Protocol. 36

IDC International Data Corporation. 3

IDI Department of Computer and Information Science. iii

IP Internet Protocol. 30

JNI Java Native Interface. 36

JSON JavaScript Object Notation. 30, 39

MVC Model-View-Controller. 37

NTNU Norwegian University of Science and Technology. iii

REGTOPP Regional Trafikk Opplysning. 21–23, 27–29, 31, 39, 60, 63, 68, 69

RWD Responsive Web Design. 7, 13, 14, 20, 21, 35

SIRI Service Interface for Real Time Information. 15

TT Team Trafikk. 20

TUC The Understanding Computer. 1, 20

URL Uniform Resource Locator. 34, 36, 48

WCAG2.0 Web Content Accessibility Guidelines 2.0. 59

Chapter 1

Introduction

In the following section, the problem definition and the concrete task for the project will be described. The motivation for the project will then be explained, as well as the research questions and goals. Finally, a background description and research methods will be presented.

The focus in this Master's Thesis has been to expand the already existing natural language travel system, Bus - The Understanding Computer (TUC), that exists in Trondheim, Norway, to a new city. BusTUC's programming and knowledge base has been modified to accommodate for a new data set provider and public transportation data for the city of Oslo. A new graphical user interface was designed and developed using the latest web technologies. A questionnaire was also administered to better understand how the general public want to use such a travel planner, and how they want the answers to be displayed.

1.1 Problem description

The assignment was presented and given by supervisor Björn Gambäck and co-supervisor Rune Sætre, and has later been modified to meet the final research questions in this thesis. It is a part of the FUIROS-project.

> *FUIROS - Fremtidens ultimate intelligente ruteopplysningssystem.*
>
> *BusTUC is a natural language bus route system for Trondheim. It gives information about scheduled bus route passings for its current domain. It is desirable to examine how a well-established natural language processing system can be moved from one domain to the other,*

and figuring out what challenges this might present. The last years have seen an increase in openness around public transportation data. Using such open data BusTUC can be moved into new unknown domains, which is the scope of this project.

The concrete task is to explore how BusTUC can be moved from one city to another, and to gain insight into the amount of work that must be done in order to achieve this goal. The result will be a prototype of the new OsloTUC, which will be a modification of BusTUC, and an accompanying application based on the ideas of using maps to present travel information. Using this prototype, a questionnaire will be administered to find out how such a system is experienced in a new city. The prototype will also be used as a platform to investigate how users ideally want to ask questions about public transportation, and how they want the information displayed by comparing this new system with an already existing solution providing information about public transportation in Oslo.

1.2 Motivation

BusTUC is a natural language bus route system for Trondheim, and has been in development since 1997. The so called Bus Oracle was installed on Team Trafikk's website[1] in 1998, and ever since users have been asking it questions about bus schedules. In 2009 it had already successfully answered more than 3 million queries, both in Norwegian and English (Amble, 2009). Today, BusTUC is deployed on AtB's websites[2], and answers on average between 1000 - 2000 queries every day. A wide variety of extensions of BusTUC has been in development in the later years as a part of the FUIROS project, including TaleTUC (Engell, 2012), a speech recognition engine for BusTUC, and an intelligent smart phone application TaBuss (Marcussen and Moland Eliassen, 2011).

BusTUC has been anchored in Trondheim for quite a while, and it is time to move it to a new domain. Based on the discoveries made by Marius Qvam Wollamo during the final months of his master's thesis (Wollamo, 2013, pg. 1), where he found that bus routes for large parts of Norway are openly available online, it is desirable to attempt to move BusTUC out of its comfort zone and into a completely new domain with data sets from Ruter.no[3]. In addition to this, visualizations using these data sets can be created and tested on real users. An attempt to import data sets for Nordland showed an example of a geographical

[1] `http://team-trafikk.no/`
[2] `https://atb.no`
[3] `http://labs.ruter.no/archives.aspx`

expansion of BusTUC, but little more than just the data set was attempted imported. Thus the motivation for this thesis became to try to import a data set from Norway's biggest city, Oslo, and prove that it works by also integrating BusTUC with the metadata provided on Ruter's website[4]. This metadata includes bus stop names and coordinates as well as municipality information. This would allow for the searching for bus stops and scheduled routes by street addresses. In essence, the idea is to refurbish the knowledge domain of BusTUC, disregarding the known domain of Trondheim, and focusing on Oslo.

Additionally, there is also motivation to develop a mobile accessible application. In recent years the mobile market has seen an immense growth. With more and more people using smart phones and mobile technologies, the interest for creating mobile accessible applications is at an all time high. According to Cable News Network (CNN), 2013 saw the biggest decline in worldwide PC shipments since International Data Corporation (IDC) began tracking the data in 1994, with as much as -13.9% (Pepitone, 2013). A press release from IDC reports mobile phones reaching a record 1 billion units shipped in 2013, up 38.4% from the previous year (IDC, 2014). We are seeing a shift in the technology usage, and it shows tendencies towards mobile platforms being the future. Therefore, a mobile application will be developed to give BusTUC a wider exposure to users in Oslo.

Similarly to the questionnaire administered in Wollamo's thesis, a quantitative questionnaire will be administered as an evaluation of the development cycle. Respondents will be asked to compare an already existing product, Ruter.no, with OsloTUC. However, the questionnaire to be administered in this thesis will aim to be more comprehensive.

[4] `http://labs.ruter.no/how-to-use-the-api/infrastructure-flat-files.aspx`

1.3 Goals and Research Questions

Conducting research is based on structured progress, and thus, research questions and goals for these research questions were defined. Following is a list of the research questions that were constructed for this thesis, and its goals.

Research question 1 *What is required when moving a well established natural language travel system to a new city?*

Research question 2 *How do users prefer travel information to be displayed?*

Research question 3 *What is the preferred way to ask about travel information?*

Goal 1 *Modify BusTUC to enable queries for the city of Oslo to find answers for RQ1. Add new data sets from Ruter to prove that it works.*

Goal 2 *Develop a proof of concept application presenting answers from OsloTUC in a map solution using the Google Maps Directions Service API. The application should perform well on mobile devices and in web browsers. Test the application with real users to answer RQ2.*

Goal 3 *Perform a survey with real users, asking them to test the proof of concept application and comparing it to existing solutions from Ruter in Oslo. It is desirable to draw conclusions on how users ideally would like to query or ask questions to a travel schedule engine (RQ3.)*

1.4 Approach

This section will cover the general approach and how research was conducted during the time span of the work on this thesis. A general description of the progress will then follow.

The work done in this thesis can be summarized in the following phases.

1. Problem definition and research questions.
2. Initial research.
3. Getting to know BusTUC and its structure.
4. Implementation of new travel schedules for Oslo.
5. Development of prototype for OsloTUC.
6. Creation and distribution of questionnaire.
7. Evaluation of results and finalization of thesis.

As an introduction to the steps above, a short summary of each will follow.

Problem definition In the very beginning, the problem definition was created together with supervisor Björn Gambäck and co-supervisor Rune Sætre.

Initial research Research was made into the currently existing travel planning solutions in Norway. Solutions existing in other parts of the world were also explored to gain some insight into the similarities between foreign and local travel planners, and how their design and answers might differ. Some of the discovered features were presented in the questionnaire.

Learning about BusTUC The author of this thesis has no prior experience with the programming language Prolog, in which BusTUC is written, or artificially intelligent systems. Coming from an object-oriented background with languages such as Java and Python, getting to know a declarative language was a much harder task than originally estimated, and took a substantial amount of time.

Providing BusTUC with route information Data sets were imported and added to BusTUC's knowledge base. Initially, a data set for the subway system in Oslo was experimented with, but later discarded because it did not provide a satisfiable amount of stops and routes. A data set for a subset of the buses in Oslo was instead imported and became the basis of the prototype of OsloTUC.

Developing a prototype Building on the experience gained in the initial research phase, a prototype was created with some of the emerging web technologies. Further research had to be made into how these technologies work in practice. A device friendly web site was finally created, allowing users to question the new OsloTUC and have the answers displayed and the route visualized on a map.

Questionnaire Research had to be made into how a questionnaire should be administered in order to answer the research questions of the thesis. A questionnaire was then distributed to a number of respondents. A three week period was given, and answers were collected.

Evaluation and finalization The results of the questionnaire were analyzed, evaluated and discussed, and a conclusion was drawn based on the results.

1.5 Research method

The research phases of the thesis used several locations for finding credible sources and background material. Among these were the following, listed in table 1.1

Source	**Location**
Google Scholar	`http://scholar.google.com/`
BIBSYS Ask	`http://ask.bibsys.no/ask/action/smpsearch`
Springer	`http://www.springer.com/`
Library	NTNU University Library

Table 1.1: Resources used to find relevant information

To find information of relevancy and credibility, several searches had to be performed. For the initial research phase of the expansion of BusTUC and the development of what was to become OsloTUC, the main searches consisted of terms of technological nature.

Key search words used in the research phase for development

- Travel planner
- Mobile market
- Mobile applications

- Native vs Web
- Responsive design
- Google Maps

To ensure that the questionnaire was administered in a proper manner according to credible sources, it was important to perform searches on this topic as well. The key words used in this search was chosen to find articles and books which would provide a theoretical basis and clear guidelines as to how to conduct the quantitative research for this thesis.

Key search words used in the research phase for the questionnaire

- Questionnaire
- Survey
- Quantitative data
- Qualitative data

1.6 Thesis structure

1. **Introduction** introduces the thesis, and contains a description of the task, research questions and goals. It also describes the process, the method used to conduct the research, and the thesis structure.
2. **Background and Theory** contains a description of various existing travel planners and the standards they use, the history and background of BusTUC, as well as an introduction to Responsive Web Design (RWD).
3. **Modifying BusTUC** describes some of the inner workings of BusTUC, and the necessary modifications that were needed to take all of its functionality to another city.
4. **Development phase** presents the technologies used to create a Graphical User Interface (GUI) for OsloTUC, and how it works across multiple device resolutions. It also presents OsloTUC's limitations.
5. **Research method** describes what a questionnaire is, how it was created, and how it was administered and distributed to a number of respondents.

6. **Results** presents the results of the administered questionnaire, the respondents' demography and their answers.

7. **Future Work** describes possible new features, or improvements, that could be made that were discovered during the work of the thesis.

8. **Discussion and Conclusion** discusses the results of the questionnaire and answers the research questions and goals before some conclusions are drawn based on the results and the discussion.

Chapter 2

Background and Theory

This chapter describes the background theory of the thesis. Related tools and technical theory will be presented.

2.1 Related tools

In this section, travel planners will be presented, as well as a list of travel planners that are related to the work that is being done on BusTUC.

2.1.1 What is a travel planner?

Individuals or groups wanting to go from one point to another need to know, or be able to acquire knowledge about, how they are going to travel. They might also want to know by which type of transportation they are travelling, and when they need to travel in order to arrive at their destination on time. A trustworthy travel planner, or travel scheduler, is an invaluable resource in situations like these.

A travel planner is a digital search engine. Using a GUI, a user will have the means for searching for travels departing from a destination, search for a route or just search for a destination based on their current location. After searching for a location, the travel scheduler will, hopefully, suggest the most optimal route from a starting position to an end position by using advanced search algorithms that are able to find shortest paths in a network of nodes (stops) and edges (roads). If the system is advanced enough, it could even supply alternative travel route suggestions, or the ability to specify waypoints via which the user wants to go.

This way the user can pick the route most desirable.

There are several types of travel planners. The most relevant examples for background research are:

Public transportation planners A public transportation planner utilizes data sets, either openly available for anyone to use, or provided by the organization or company offering the service. These will suggest how to get from point A to point B by travelling with public transportation, such as buses, trains, trams or subways. BusTUC and OsloTUC are examples of such travel planners. This type of travel planner is the main topic of the present thesis.

Journey planners Journey planners can be used if you, for example, are going on a road trip, vacation or are venturing to another country. They will use gathered or available data to suggest a route between countries and different travel agencies or services. Such travel planners often support the most common long distance methods of travelling, like trains or airplanes, but can also support localized public transportation methods. *Traveline* and *Eurail* are such a travel planners, and will be explored in section 2.1.4

Road route travel planners Road route travel planners are typically what you would use in your car when you do not know the route to your destination. You are travelling from your home to an address in an unknown city. A typical use case scenario would be to turn on your Global Positioning System (GPS), enter the destination address into the system, letting it guide you throughout your journey. Such systems often support guidance by a voice, allowing you to keep your eyes on the road to avoid the distraction of having to look at the screen when you arrive at intersections or are otherwise unsure of where you are going. GPSs are stationary devices, often mounted on the dashboard of cars. User interfaces also exist online, such as Google Maps, or Gule Sider, which will be explored in the following sections. This kind of travel planner is also highly relevant to this thesis.

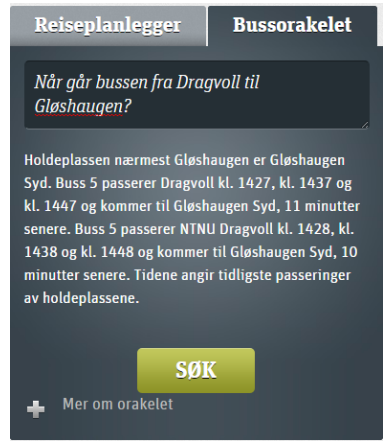
2.1.2 Travel planners utilizing BusTUC as its backend

Atb.no

AtB[1] is a company that administers the public transportation accessible in the Sør-Trøndelag County, in which Trondheim is located. AtB is owned by Sør-Trøndelag County.

[1] `https://atb.no`

(a) A til B (AtB)s boxes travel scheduler.



(b) BusTUC included on AtBs web site.

Figure 2.1: Three ways to search for travel information on atb.no

atb.no gives the user an option to input a departure point and an arrival point, as shown in figure 2.1a. These points can either be a bus stop, which, while writing, auto completes with a drop down list showing matching bus stop names, or a street address. Their system supports approximate walking time to the nearest station, and the result is displayed in a listed format, where each item is a bus departure in the near future. Clicking each item reveals the complete route, start and end time, and total travel time. The result is also displayed in Google Maps embedded into the site, marking each point of interest, but not each station. The route calculation is done by TravelMagic, which will be reviewed in section 2.1.3. AtB.no is also the only public web site by a travel agency to include BusTUC directly in their web page as an alternative to inputting your departure and arrival points in predefined boxes, as shown in figure 2.1a and 2.1b.

The buses used by AtB are tracked with GPS units, which allows for the tracking of real-time passing information. Real-time location data for each bus and routes can be retrieved via a publicly exposed Application Programming Interface (API) which is available through a web service. To use this API one needs an API-key.

AtB's web site is designed for both desktop users and mobile device users. A different[2] version of the web site will be served if you are on smaller devices.

[2] `https://m.atb.no`

This enables browser viewing on both desktop computers and mobile devices.

MapTUC

MapTUC[3] is the result of the research done by Marius Qvam Wollamo in his Master's Thesis (Wollamo, 2013). It is a modification of BusTUC[4] which enables tram support for the city of Trondheim. In addition to displaying the results in a textual form, it also displays the results in a Google Maps platform, embedded into the web page. In contrast to AtB's way of drawing the route in the map, MapTUC includes each bus and tram stop. These stops are displayed as markers on the map, each with icons representing what kind of transportation type passes that particular stop. Clicking on a marker in the map also reveals information about the bus stop, showing the other buses passing the selected station, and their real-time information. The use of maps in MapTUC inspired the work to be done in this thesis, where a similar feature for the city of Oslo will be created. MapTUC only supports a desktop version of the web site. If opened up on mobile devices, it does not work the intended way. Thus, a goal for this thesis is to create a more modern application that will work on any device, regardless of its height and width.

TaBuss

TaBuss is an Android application developed by Christoffer Marcussen, Runars Anderstuen and Lars Moland Eliassen (Marcussen and Moland Eliassen, 2011; Marcussen and Anderstuen, 2012; Marcussen et al., 2012). Since its release, it has been downloaded and installed between 100 - 500 times[5]. TaBuss is intended to be a smart travel scheduler for buses utilizing BusTUC as its backend for natural language queries. It offers real-time information from AtB and location technology which enables the user to input only a destination point as the question. The departure station is found automatically by locating where the user is. Another useful feature is the use of Case Based Reasoning (CBR), which enables the application to automatically suggest a bus route based on previous travels and events.

[3] `http://busstuc.idi.ntnu.no/maptuc/`
[4] `http://busstuc.idi.ntnu.no/`
[5] `https://play.google.com/store/apps/details?id=test.BusTUC&hl=no`

2.1.3 Other travel planners in Norway

Ruter.no

Ruter.no[6] is Ruter AS' web site where one can ask for travel information in the city of Oslo and surrounding counties. It supports subways, trains, trams, buses and boats, and can suggest routes between all of these travel types. By choosing an *advanced search* option one can specify which type of transportation is acceptable, which lines you want, or even select only direct routes without any transfers. Options for choosing the amount of time to wait for a transfer and maximum time spent walking is also present. In addition, the user can save favorites, that can be used for quickly selecting preferred routes on later returns to the website.

Asking for directions is supported via predefined text boxes for place of departure and arrival. When filling in departure or arrival, a drop down list will appear. Each item in the drop down list shows the type of transportation, and supports addresses as well. Searching for stops by placing a marker in a map is also available.

Ruter.no is also elegantly designed both for desktop computers and mobile devices. The same web site will be served for desktop computers and mobile devices, and utilizes a web technology called RWD. This technology will be discussed in section 2.4.

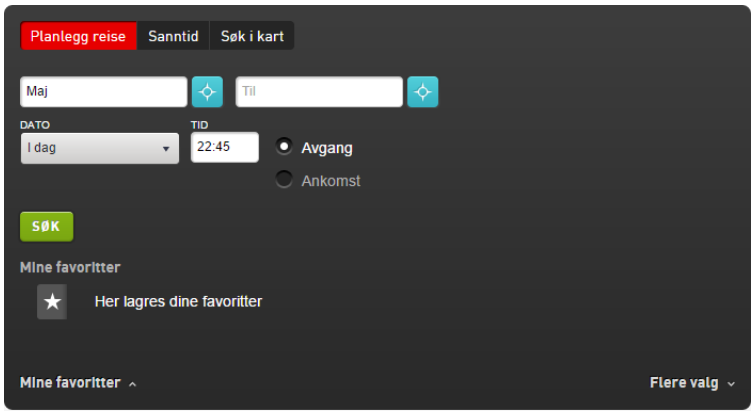


Figure 2.2: Ruter.no's travel scheduler

[6] `https://ruter.no/`

RuterReise

RuterReise is a mobile application developed for Ruter. It is available as a native application on the mobile platforms Android[7], iOS[8] and Windows Phone[9]. In the application you can search for travel routes, stops, real-time information about travel lines and it has the ability to search in or display routes in a map. The application supports storage of your favorite searches, so as to perform the same search in the future more easily. Real-time information about routes and delays are also made available.

Gule Sider

Gule Sider is best known for being the place to search for addresses, people and telephone numbers in Norway, but they also have a countrywide travel planner[10]. Their web site is elegantly designed and provides a great variety of alternatives without cluttering the design or user experience. As with AtB.no, discussed in section 2.1.2, they serve the same web site to users on mobile devices and users of desktop computers. Responsive Web Design (RWD) is used to achieve this result.

Their travel planner supports filling in predefined boxes for departure and destination, and will automatically make suggestions as you type. This gives you the option to select items in a drop down list. A unique option stems from their ability to search for personal information. When you search for a person or a phone number you have the option to pick that person's address as the destination point. Beyond this, their travel planner supports buses, trains, subways, trams, ferries, and even airplanes. Options exist to pick the fastest or shortest route, and choosing a start, end time and date for your travel. In addition to being a public transportation travel planner, Gule Sider is also a road travel planner, with information and directions given for each point in the route. Results are displayed both in the map and as a list of travel suggestions. Clicking these travel suggestions will reveal the route in detail, showing the time and transportation type for each point.

[7] `https://play.google.com/store/apps/details?id=no.ruter.reise`
[8] `https://itunes.apple.com/no/app/ruterreise/id299318111?l=nb&mt=8`
[9] `http://www.windowsphone.com/nb-no/store/app/ruterreise/5b687b57-2381-474b-aa55-f46ed79ee030`
[10] `http://www.gulesider.no/`

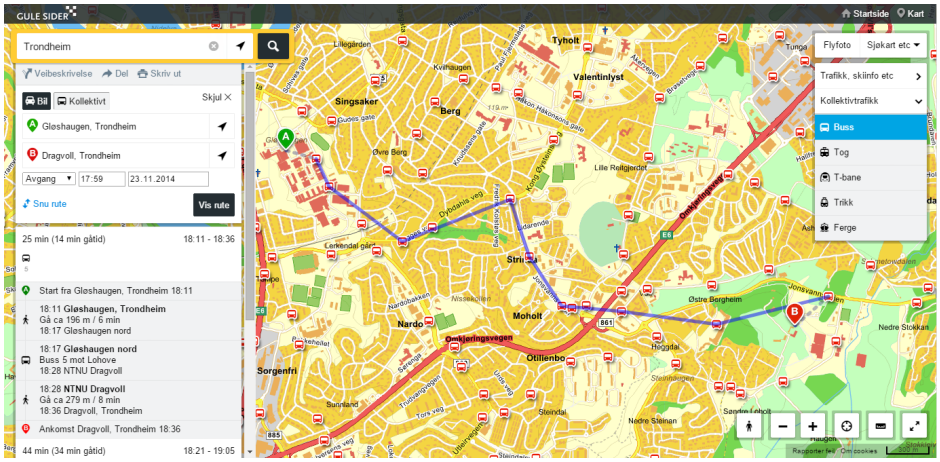


Figure 2.3: Gule Sider's travel scheduler

TravelMagic Reiseplanlegger

TravelMagic is a travel planning engine made by DataGrafikk [11]. In stark contrast to the other related tools discussed in this chapter, it is delivered as a back-end engine service to customers and operators. This means that companies like AtB do not have to create their own engine, but can rather plug it into their service. DataGrafikk are the ones responsible for the gathering of standardized traffic- and route information data sets. The engine processes these data sets and generates suggestions for routes and departure lists. They also undertake self serviced maintenance and updates of the engine around the clock. This alleviates the customers from doing a lot of the updating and maintenance in their own systems. It was launched for the first time in 1997 as a travel planner for the Haugesund area in Norway, and AtB now actively use it on their web page. The engine handles, among other things, searches between departure and arrival stops, points of interest, and the display in Google Maps. If such information is available, TravelMagic will also display real-time information for stops and vehicles, as well as information about deviation in the scheduled routes[12]. This is accomplished with a service called Service Interface for Real Time Information (SIRI)[13].

[11] http://www.datagrafikk.no/
[12] http://www.datagrafikk.no/?page_id=2896#SIRI
[13] http://user47094.vs.easily.co.uk/siri/overview.htm

2.1.4 International tools

Traveline

Traveline describe themselves as a journey planner for travels in Great Britain. It is a partnership between transport companies, local authorities and passenger groups to support travel information for buses, trains, coaches and ferries, trams, subways and walking. For the time being they are offering their services via their web site[14], and an iOS application, but they are planning to expand to Android and Windows Phone during the spring time in 2015. Their web site is served as separate sites for mobile devices and desktop computers. The design on both versions is clean and easy on the eye, and easy to navigate. Multiple journeys are displayed as results, and upon clicking a journey, a list of steps will appear. These steps describe transportation type, how long time each step takes and where to go.

Like many of the other related tools, Traveline provides searching for a journey via predefined boxes. This service, however, does not let you choose from a drop down list until after you have clicked 'search'. After selecting the specific departure and destination, the search will begin. Searching for a journey within the same city is as fast as with the other tools, but the search takes a relatively long time when attempting to find a journey between the countries of Great Britain.

Eurail

Eurail is a travel planner that lets you look up train times for a majority of the central countries in Europe, Scandinavia and certain countries in Eastern Europe. Eurail is interesting because it is a *"complete package"*. On their web site[15] customers can buy a train pass which will be valid for most train rides with companies working with Eurail. It is for all intents and purposes a journey planner, as you most likely would use the service to plan journeys between two or more countries.

Their travel planner has predefined boxes, and an option to input up to two stops to travel via. Options to select date of departure and arrival are also present, as well as the ability to select whether or not one wants to accept travelling by trains that require a reservation. The difference in this app, however, is that no map is used when displaying the result. The result consists of a list of stops with information about each train, departure time and transfer time.

[14] `http://www.traveline.info/`
[15] `http://www.eurail.com/`

In addition to their website, Eurail also offers native applications for Android users[16] and iOS users [17]. Realizing that international mobile data traffic is expensive, Eurail has made these applications offline compatible. The user can thereby find route schedules offline, and does not have to spend costly data traffic, or find free WiFi-networks, to be able to use the application.

Google Transit

Google Transit[18] was developed by Google as a showcase for the now discontinued Google Labs, and was released to the public in December, 2005. The showcase consisted of public transport route planning for the city of Portland, Oregon, in the United States. Today it features over 800 cities all around the world[19], and has support for buses, trains, trams, subways, ferries, and walking directions. In 2007, Google Transit was moved from Google Labs and was fully integrated into Google Maps as a layer on top of the already existing maps layer. The service itself calculates the optimal travel routes that are being served back as a response to a query from the user.

Google Transit, from now on referred to as Google Maps, is also the only other travel scheduler in this chapter, aside from BusTUC, which supports natural language to some degree. It can handle queries like *"from majorstuen to frogner plass"* but not more advanced queries like *"from majorstuen to frogner plass before 18:00"*. Issuing such a query will result in Google Maps asking you if you really meant something else. It seems clear that Google Maps interprets the keywords *"from"* and *"to"* and places the departure and arrival place into predefined boxes accordingly. Google Maps will show alternative route suggestions, and draw the entire route onto the map with lines between each stop. Not only does it draw the currently selected route, it also draws the alternative route suggestions so you can visually compare them.

The web site is not very original looking when viewed in a browser on a smart phone, but it shines when displayed on a desktop computer. On Android, Google Maps comes as the default maps application and is tightly integrated in the Android environment.

Google Maps makes it easy for companies to contribute. If a company provides transportation services and have static route information data feeds, they can contact the Google Transit team to sign up for a partnership. The data feed must

[16] `https://play.google.com/store/apps/details?id=de.hafas.android.eurail&hl=no`

[17] `https://itunes.apple.com/us/app/rail-planner-offline-timetable/id579547877?l=de&ls=1&mt=8`

[18] `http://maps.google.com/intl/en/landing/transit/`

[19] `http://maps.google.com/landing/transit/cities/index.html`

be delivered to Google in the form of Google's own standardized format, General Transit Feed Specification (GTFS). In 2009, Trafikanten said it was complicated delivering data to Google Maps because of economical and political reasons[20]. This changed in 2013, when Ruter stated in a press release it was now possible to use Google Maps to search for travel information in Oslo and the Akershus area[21].

2.1.5 Summary of travel planners

In table 2.1, a summary is made of the travel planners that have been studied. The focus was directed towards whether they have a travel planner on a web site, if their web site is optimized for mobile, whether they have a mobile application or not, as well as the support for natural language (NL) and auto completion of travel points.

Name	Web site	Opt. for mobile	Mobile app	NL	Auto complete
Ruter	x	x	x		x
RuterReise			x		x
Gule sider	x	x			x
Traveline	x	x	x		
Eurail	x		x		x
Google Maps	x	x	x	x	x

Table 2.1: Summary of travel planners

2.2 Prolog

In the late 1960s and early 1970s, logic programming emerged as a basic idea from work on automated deduction. The work of Alain Colmerauer and his team in Marseilles, France, in the early 1970s, later resolved in a realization that one could implement a logic programming language with an efficiency comparable to that of procedural languages, such as Pascal (Pereira and Shieber, 2002, pages 15-16). Until 1972 it had been thought that logic programming could only be used declaratively, so the idea that first order logic could be used in a procedural programming language was revolutionary (Lloyd, 1987, pages 1-3). "Marseille Prolog" was born as a result, and was the first implementation of what is today

[20] `http://labs.ruter.no/2009/2/20/vil-trafikanten-komme-paa-google-mapstransit.aspx`

[21] `https://ruter.no/no/verdt-a-vite/presse/Pressemeldinger/ruter-i-google-maps/`

known as Prolog, and its main focus was on Natural-Language Processing. Prolog simply stands for "Programming with Logic". Prolog has since been used in a wide variety of applications, particularly in the field of Artificial Intelligence (Pereira and Shieber, 2002, pages 15-16).

Prolog today, as well as many other logic programming languages, although procedural, can be thought of as largely declarative compared to languages such as C++ and Java, which are procedural and object oriented. The idea of calling a function and having results returned into variables is foreign to Prolog. Instead, relations between entities are expressed, and calls to functions can be thought of as questions to whether a certain rule holds true or not (Pereira and Shieber, 2002, page 3).

As an illustration of what Prolog looks like and what it can do, take the example of finding out if a stop 'Y' is coming up after a stop 'X'. First we define the rules for finding stops that are coming up.

Listing 2.1: An example of Prolog code

```
bus_stop_after('Vestre Aker kirke', 'Marienlyst').
bus_stop_after('Marienlyst', 'Majorstuen').
bus_stop_after('Majorstuen', 'Frogner stadion').

is_ahead(X, Y):-
    bus_stop_after(X, Y).

is_ahead(X, Y):-
    bus_stop_after(X, Z),
    is_ahead(Z, Y).
```

We can now ask if 'Frogner stadion' is a stop somewhere after 'Vestre Aker kirke':

```
?- is_ahead('Vestre Aker kirke', 'Frogner stadion').
yes
?- is_ahead('Majorstuen', 'Marienlyst').
no
```

Unlike most popular programming languages today, the programmer is more interested in the knowledge than the algorithms that exploit the knowledge. By going through the database of bus stop relationships, Prolog does the job of figuring out the answer to the question asked. (Bratko, 2001, xiii-xiv)

2.3 BusTUC

Bus - The Understanding Computer (TUC) is a natural language travel planner, written in Prolog, which has been in use by the public since it was deployed by Team Trafikk (TT), now AtB, in 1998. BusTUC is the basis for this thesis, and the goal is to expand the system to fully support the city of Oslo. People can ask BusTUC questions in natural language, the way they normally think about the information being discussed, and the travel planner's goal is to automate the function of a route information agent.

To be able to make this intelligent agent work, a lexical analysis of the question asked is performed, as well as a syntactic and semantic analysis. Reasoning is then applied, and a knowledge base of travel route data sets is consulted to provide an answer. Not only can the system interpret natural language queries - it will also provide answers itself in natural language, giving the user a more conversational feel. The system itself is bilingual, and will work equally well for English and Norwegian queries[22]. Apart from answering questions about bus schedules, BusTUC can answer questions like *"Are you intelligent?"*, *"What is the meaning of life?"* or *"Who made you?"* (Amble, 2000), (Amble, 2009).

BusTUC needs to be updated regularly with up to date travel schedules in order to deliver correct route information. Today, routes are updated manually by the administrators of the system. In order to automate this, a system could be made that fetches the latest available routes, however, this entails that AtB would have to provide a service that allows for automatic fetching.

2.4 Responsive Web Design

Fixed web site design is today no longer appropriate. When visiting web sites, users expect to have an ideal viewing experience. This is due to the fact that a wide variety of different devices with different screen resolutions have emerged the recent years (Natda, 2013).

In the past, most designers relied on JavaScript or Cascading Style Sheets (CSS) media types to create resolution-aware layouts. In 2010, Ethan Marcotte coined the term Responsive Web Design (RWD). He explained that, instead of tailoring the design of web pages to an ever-increasing number of web devices, we should create one design that would work across multiple devices with different resolutions. Rather than targeting specific browser versions, we can now

[22] And partially Swedish, which is treated as misspelled Norwegian.

create web sites with multiple ideal resolutions (Marcotte, 2011). A measure that can be used to handle both a desktop version and a mobile version of a web site is to create two separate versions of the web site. By serving the mobile version when a mobile device connects, one can achieve the effect of having a mobile friendly web site. Another option, which is encompassed in RWD is to use CSS3 Media Queries. By using Media Queries, one can easily create conditional statements based on, for example, width and height, which enables us to customize the presentation to any device's resolution using only style sheets (CSS3 will be discussed in chapter 4.3). All the major web rendering engines today has complete support for these new Media Queries (Bryant and Jones, 2012).

Some of the travel planners discussed in this chapter use RWD in their web site design.

2.5 Travel exchange standards and services

In this section, travel exchange formats will be explained, and formats important to this thesis will be covered.

2.5.1 What is a travel exchange format?

Travel exchange formats can be viewed as standards or contracts between the producer of the route data sets and the end-users or companies that wish to implement them and use them. In order to ensure that the data can be read by computer programs without any errors, standards must be created that will clearly define how such data sets *must* be read. A standard tells which fields or files are required, which is to say that if these fields or files are omitted or empty, the data set is not valid. In addition, a standard should provide clear and concise descriptions as to what each particular field or file contains, what kind of data type these are, and how they should be used by themselves or in relation to other fields or files. There are two exchange standards that are relevant for the work in this thesis, namely GTFS and REGTOPP.

2.5.2 General Transit Feed Specification (GTFS)

GTFS[23] is a travel exchange standard developed by Google, and it was first launched in 2005 as a feed specification consumed by Google Transit. It was first introduced under the name *Google Transit Feed Specification*, but a name change proposal was made in 2009[24] as a result of the growing popularity of the

[23] `https://developers.google.com/transit/gtfs/reference`
[24] `https://groups.google.com/forum/#!topic/gtfs-changes/ob_7MIOvOxU`

format. In addition to being used in Google Transit, many other companies were now using it to consume route information. The implementation of the standard itself is a collection of text files in the Comma Separated Values (CSV) format. Each of these files contain specific information to describe individual parts of a route schedule. The files are linked together by fields contained within the files, much like a relational database would work. The required files are shown in table 2.2

Filename	**Description**
agency.txt	One or more transit agencies that provide the data in this feed.
stops.txt	Individual locations where vehicles pick up or drop off passengers.
routes.txt	Transit routes. A route is a group of trips that are displayed to riders as a single service.
trips.txt	Trips for each route. A trip is a sequence of two or more stops that occurs at specific time.
stop_times.txt	Times that a vehicle arrives at and departs from individual stops for each trip.
calendar.txt	Dates for service IDs using a weekly schedule. Specify when service starts and ends, as well as days of the week where service is available.

Table 2.2: Required files in the GTFS standard.
Table descriptions are collected from
`https://developers.google.com/transit/gtfs/reference#FeedFiles`.

2.5.3 Regional Trafikk Opplysning (REGTOPP)

REGTOPP is a project that was started in the 1990s, and the REGTOPP-format is a standardized travel exchange format made by the travel industry itself. Any further references to REGTOPP in this thesis are referring to the format, not the project. The standard consists of several files, some of which are required and shown in table 2.3, with fields and values linking the files together. This is very similar to how the GTFS standard works. BusTUC uses REGTOPP as its source of route information via a REGTOPP to Prolog code converter created by co-supervisor Rune Sætre, Tore Amble and Tore Bruland in 2010. Where GTFS uses CSV as its format, REGTOPP uses its own definition. Fields are given an index start and an index end, which means that each field is bound to a specific place in each line of the file. This makes the format very prone to index errors,

but is very likely more efficient when the parsing is executed correctly. It also limits the data sets immensely by putting a restriction on the length of all fields. For instance, stop names are restricted to 30 characters in version 1.2. Should the rules for stop names be changed, a release of a new version is needed. The format has undergone several changes since its origin, from version 1.1 to 1.3.A. Table 2.3 shows the required files and a description of each file in version 1.2.

Filename	Description
turix.tix	Shows all trips that are available in the current data set.
turmstr.tms	All proceeding trip stops and trip times for all variants of all trips
hpl.hpl	Contains all trip stops with description and properties of stops.
dagkode.dko	Shows the time validity of all trips.

Table 2.3: REGTOPP v1.2 required files

Chapter 3

Modifying BusTUC

This chapter will describe the process of modifying BusTUC from being centered around Trondheim and its surrounding areas to being a near feature complete modification for the city of Oslo.

3.1 The structure of BusTUC

BusTUC was created by Tore Amble, who sadly passed away in 2012, and has since been in development by co-supervisor Rune Sætre. During the years Rune Sætre has been working on the project, he has attempted to modularize it, making sure that the knowledge base itself is one of these modules. This makes BusTUC easier to work with, and reduces the amount of work that needs to be done to extend the knowledge base. Knowledge of the Prolog programming language and how it works, as well as knowledge of how the different systems/-modules fit together, however, is still vital to an efficient process when working with this system, as it is with all other well-established systems. The process of learning how BusTUC works took much longer time than expected. A simple line count of the entire project reveals that it contains roughly 200,000 lines of Prolog code, excluding the knowledge base.

The most important files in the project related to the work done in this thesis is shown in figure 3.1, which also shows to some extend the structure and modularization of BusTUC. *busstuc.pl* is the main project starter, and is used for running the project, as well as creating hashes of all names in the currently used knowledge bases. These hashes are vital to the credibility of the system, as they allow users to, among other things, type names of stops and places with a certain degree of error. For instance, if a user asks for the place *"vikingskipne"*,

the system will change this to the correct name *"vikingskipene"*. This allows for correctly returned answers despite mistakes made by the user. In a non-natural language system, this would not be possible as these systems use direct lookup, or IDs of stops and/or places to give correct results. In contrast, as can be seen on Ruter's website[1], among others, when typing in a route stop or address the user must select a valid entry for the query to be successful.

```
bustuc
| - db
|   | - tables
|   |   |   - r0310_140319
|   |   |   | ...
|   |   |   - r0320_141005
|   |   |   | ...
|   | - places.pl
|   | - regcompstr.pl
|   | - regstr.pl
|   | - statcoord2.pl
|   | - route_period.pl
| - busstuc.pl
```

Figure 3.1: Rough outline of the important parts of BusTUC in this thesis.

The *db* folder contains the knowledge base of BusTUC. Here you will find the scheduled routes within the table folder. Each scheduled route is a collection of routes within a given time period, and the valid time periods are defined in the file *route_period.pl*. *places.pl* contains a collection of places known to BusTUC. Contained within *places.pl*, is also a collection of synonymous places, as well as additional word corrections. This file is also used to exclude certain places. For instance, in the configuration for Trondheim, most of the popular addresses located in the city of Oslo are excluded. If users attempt to ask questions about these places, an answer will be returned telling the user that the place in question is not available in the current configuration. *regstr.pl* contains all known addresses, the addresses' street numbers and the closest located route stop.

Figure 3.2 shows a few lines from the file *regstr.pl*. The uppermost line tells us that *Solheimgata* 1-12's closest bus stop is the station *Frogner plass*. The file contains one entry per known address. *regcompstr.pl* contains lexical rules which allows BusTUC to find valid stop names or addresses. Figure 3.3 gives an

[1] `http://ruter.no`

```
streetstat(solheim_street,'Solheimgata',1,12,frogner_plass).
streetstat(solkroken,'Solkroken',1,22,ulvenkrysset).
streetstat(sollerud_street,'Sollerudveien',1,54,lysaker_bru).
```

Figure 3.2: Illustration of street station for an address.

example of how words are split up internally in BusTUC. Looking at the first line we can see that *Camilla Colletts' Street* is tokenized into *camilla*, *colletts* and *street*. This allows BusTUC to quickly discard places when two words following each other are not located in one of these tokenizations.

```
composite_road(camilla,[colletts,street],camilla_colletts_street).
composite_road(cappelens,[street],cappelens_street).
composite_road(carl,[berners,plass],carl_berners_plass).
```

Figure 3.3: Example of internal tokenization of words.

3.2 Configuring for the city of Oslo

The first working version of OsloTUC was created relatively early. It was a test to see whether or not a data set provided from Ruter.no[2] could still be successfully imported into BusTUC, as shown by Wollamo in his thesis (Wollamo, 2013). A conversion tool created by Tore Amble and Tore Bruland in 2010, and later updated by co-supervisor Rune Sætre, was used to convert the REGTOPP data set to Prolog code. This is necessary because BusTUC needs Prolog code in order to understand the route schedules. The selected data set was for the subway system of Oslo, and figure 3.4 shows an answer given by OsloTUC after the import of the subway route schedules.

In figure 3.4 a question was posed to OsloTUC, asking when one has to leave from the station *Majorstuen* to arrive at *Stortinget* at a given time. One thing to notice is that these are travel lines for subways, not buses as the answer implies. This is an internal configuration in BusTUC, where each route number is assigned a vehicle name. In the case of this answer, the word "bus" should clearly have been replaced with the word "subway". However, when creating modular-

[2] `http://labs.ruter.no/archives.aspx`

```
   N: When do I have to leave from Majorstuen to be at Stortinget before 10:10pm?

Bus 6 passes by Majorstuen at 9:57 pm and arrives at Stortinget , 4 minutes later.
Bus 3 passes by Majorstuen at 10:01 pm and arrives at Stortinget , 4 minutes
     later.
Bus 1 passes by Majorstuen at 10:03 pm and arrives at Stortinget , 4 minutes
     later.
Bus 2 passes by Majorstuen at 10:03 pm and arrives at Stortinget , 1 minutes
     later.
Bus 5 passes by Majorstuen at 10:05 pm and arrives at Stortinget , 4 minutes
     later.
Bus 4 passes by Majorstuen at 10:07 pm and arrives at Stortinget , 4-10 minutes
     later.

The hours indicate the earliest passing times.
```

Figure 3.4: The first working version of OsloTUC.

ized versions of BusTUC, this work will eventually become tedious. Each route number has to be updated with its corresponding vehicle id or name. During the course of this thesis it was discovered that the vehicle type is available through the REGTOPP data sets. This could be used by the REGTOPP to Prolog converter to tell BusTUC what type of vehicle each route entry is. Although providing only an id as a numerical value for each vehicle type might seem like the correct way of doing this, it would most likely be a mistake. During updates of the REGTOPP format this field has been changed, and to circumvent having to update each version of BusTUC between versions of REGTOPP, the author proposes adding the string value of the name of the vehicle type to the converted data sets.

After having imported and proved that the subway system would work as a knowledge base for OsloTUC, it was decided that just importing this data set provided little information for a city the size of Oslo. Instead, the author decided to import data sets of buses in Oslo, which would provide coverage of routes for a larger portion of the city and its surroundings. This proved to be a more daunting challenge than initially perceived, for a number of reasons explained in the following paragraphs.

On the first attempt to import a data set for buses in Oslo, one problem immediately became clear. In a few cases, the numbering for buses in Oslo differs slightly to the numbering in Trondheim. Bus numbers containing alphabetical characters in Trondheim has the character stripped from the number when being

parsed over to Prolog code. For instance, bus number *5e* would in Trondheim's version of BusTUC become *5*. However, in Oslo, some buses had names like *N20* or *60X*. From a Prolog standpoint, the name *60X* is not a valid symbol, because it has numerical values followed by text. The name *N20* is not valid either, as variables starting with a capital letter will be interpreted as singleton variables. It was decided that the easiest and best solution was to represent all identifying values as text, rather than Prolog atoms. Modifications had to be made to the REGTOPP to Prolog converter in order to make this a permanent solution. After this discovery, and having trouble with other identifying fields, it was decided that all identifiers would be treated as unique strings. This resulted in the changes shown in figures 3.5 and 3.6.

```
route(bus_0025_0562,25,25).
route(bus_077B_0006,77B,77B).
route(bus_080E_0027,80E,80E).
route(bus_0N70_0005,N70,N70).
```

Figure 3.5: Identifiers before modification.

```
route(bus_0025_0562,'25','25').
route(bus_077B_0006,'77B','77B').
route(bus_080E_0027,'80E','80E').
route(bus_0N70_0005,'N70','N70').
```

Figure 3.6: Identifiers after modification.

During the course of the configuration phase, some of the data sets delivered by Ruter were either damaged or contained illegal fields, which slowed this phase down significantly. In addition to this, Ruter started using a newer version of REGTOPP which was not yet supported by the REGTOPP to Prolog converter. The change in data set version was not discovered at first, because it did not break the conversion program. Instead it produced a knowledge base that gave wrong passing times when asking OsloTUC a question. New modifications had to be made to the conversion program once this was discovered in order to support the newer version.

To fully support a city, street addresses for the city would have to be obtained. In BusTUC one can ask for a street address, and it will figure out what the closest bus stop is, and calculate a route from that point. Up until now,

only bus stops were working in OsloTUC. A list of all valid street addresses was found[3], but no easy way of obtaining these was discovered. It was therefore decided to write a simple web-crawler to obtain all addresses available on the Oslo Municipality web site. All addresses were parsed and stored. To find the nearest bus stop to an address, coordinates for each street address must also be obtained. As the listing on the Oslo Municipality web site contained no coordinates, it was decided to fetch these coordinates with the Google Maps API. This proved somewhat troublesome. The Google Maps API is free to use, but it has a few limitations. One Internet Protocol (IP) address can send 2,000 requests to the Google Maps API every day. After this, all requests sent that day will be discarded. In addition, one is not allowed to send more than 10 requests per second. The addresses obtained for Oslo by far surpasses 2,000, and to alleviate this situation, it was determined that addresses would be grouped together by street address numbers. It was decided that grouping these addresses would be acceptable and sufficient to provide a working prototype. This, of course, meant that some addresses would have a rather large closest bus stop error rate. For instance, grouping *'Aasta Hansteens vei 1 - 10'* would not provide any huge errors, but combining *'Ammerudveien 1 - 300'* would. When asking Google Maps for a range of addresses, an approximation will be returned. *'Ammerudveien'* has at least 7 bus stops, but only one of these will be returned if querying these addresses in OsloTUC. Taking the limitations into account, and the fact that nearly 2600 street *names* (excluding numbered addresses) were found on the Oslo Municipality web site, it would be far too time consuming to get coordinates for each individual address. All address coordinates were also stored. Once all addresses and all coordinates were successfully retrieved, a program was written to calculate which bus stop in the knowledge base was closest to each address. This resulted in the creation of new *regstr.pl* and *regcompstr.pl* files, containing street addresses and their coordinates, as well as street names with tokenizations.

The *statcoord2.pl* file contains all bus stops known to BusTUC and their coordinates. This file is essential to produce BusTUC's JavaScript Object Notation (JSON) response containing the complete route information, which would later be used to create routes in a map. A program was written to update this with the latest data collected from Ruter's web site. Ruter has made a flat file in the CSV format[4] available, which contains all stops in Norway. This file was parsed, and combined with the already existing *statcoord2.pl* file to create an updated version containing all bus stops in Norway, but most importantly the bus stops in the city of Oslo. An example query with the new knowledge base is shown in

[3] `http://www.oslo.kommune.no/om_oslo_kommune/bydelsoversikt/article86101-7808.html`

[4] `http://labs.ruter.no/how-to-use-the-api/infrastructure-flat-files.aspx`

figure 3.7.

3.3 Summary

Following is a summary of the important changes or points made in this chapter:

1. Addresses are important to the usability in BusTUC. All addresses found for Oslo was imported and mapped to their closest bus stop.

2. Using Google Maps to get coordinates for all addresses in a new city would be very time consuming. Because of this, all addresses could not be mapped to its closest bus station. A better source for finding coordinates for addresses would be preferable.

3. The Prolog to REGTOPP converter was modified to accomodate the updated version of REGTOPP.

4. Identifiers for fields in BusTUC's knowledge base were changed from numerical values to strings in order to support bus names with both numbers and letters.

5. When moving BusTUC to another city, street names for the old city must be removed. This is because different cities can have the same street names. Taken this into account, it would be preferable to isolate each city to its own knowledge base, for example, by modularizing it even further.

6. BusTUC is more error prone to users writing words with spelling mistakes than other travel planners directly mapping, for example, bus stop names to numerical identifiers.

7. An update in the REGTOPP version went by unnoticed for some time. Although the converter produced a semantically valid knowledge base, the information within this was erroneous. This made BusTUC produce wrong results when it was asked for travel information. This could imply that the Prolog to REGTOPP converter needs stricter version checking.

8. Moving from one city to another has proven to be more time consuming than first anticipated. A reason for this can be the change from one provider of data sets to another. Some of Ruter.no's data sets were damaged, or corrupt, and in one case contained illegal data according to the specification of REGTOPP.

```
N: From Trudvangveien 15 to Frogner plass?
..................................................


                            sentence
                  ____________|____________
                 /                         \
            verb_modifiers                  lit
          __________|__________              |
         /                     \             |
        pp                      pp           ?
    ____|____               ____|___
   /         \             /        \
 prep        name        prep       name
   |           |           |          |
   |           |           |          |
 from          -           to    frogner_plass
               |
               |
        trudvang_street


[modifier(A):::(frogner_plass isa station,trudvang_street-15 isa street,srel/from/
    place/(trudvang_street-15)/A,srel/to/place/frogner_plass/A,event/real/A)]


..................................................
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
{
  "transfer" : "false",
  "timeset" : "false",
  "departures" : [
    { "busstopname" : "Majorstuen", "busstopnumber" : 3010201, "busnumber" : 20,
        "time" : 1914, "duration" : 3, "destination" : "Frogner plass"
    }
  ]
}
{
  "route" :
    [
        { "name" : "Trudvangveien 15", "type" : "streetaddress", "role" : "start"},
        { "name" : "Majorstuen (i Kirkeveien)", "type" : "busstop", "nr" : 3010201,
            "role" : "enter", "xcoord" : 59.92915212410355, "ycoord" : 10.716126588209095},
        { "name" : "Frogner stadion", "type" : "busstop", "nr" : 3010222,
            "role" : "pass", "xcoord" : 59.927195770991254, "ycoord" : 10.712554284961506},
        { "name" : "Vigelandsparken", "type" : "busstop", "nr" : 3010221,
            "role" : "pass", "xcoord" : 59.92469026773165, "ycoord" : 10.708381629707821},
        { "name" : "Meldepunkt", "type" : "busstop", "nr" : 21810,
            "role" : "pass", "xcoord" : 5.996446696068045, "ycoord" : 5.023221708928743},
        { "name" : "Meldepunkt", "type" : "busstop", "nr" : 22915,
            "role" : "pass", "xcoord" : "null", "ycoord" : "null"},
        { "name" : "Frogner plass", "type" : "busstop", "nr" : 3010220,
            "role" : "exit", "xcoord" : 59.92221243214387, "ycoord" : 10.704837191037917},
        { "name" : "null", "type" : "streetaddress", "role" : "stop"}
    ]
  ,
  "busstuc" :
  "The station nearest to Trudvangveien 15 is  Majorstuen .
  Bus 20 passes by  Majorstuen at  7:14 pm , at  7:24 pm , at  7:34 pm , at  7:44 pm ,
  at  7:54 pm  and  at  8:04 pm
  and  arrives at  Frogner plass ,  3 minutes later.

  The hours indicate the earliest passing times."
}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 3.7: Showing a successful query with street address and bus stop name using the new knowledge base and a Prolog console. Output is modified for readability.

Chapter 4

Development phase

This chapter describes the development phase of the user interface of the prototype of OsloTUC and its design, as well as explaining the different strategies and choices considered throughout this phase.

4.1 Native versus web

When deciding to develop an application that can be used on multiple platforms, there are different solutions available to the problem at hand. One can choose to develop independent applications for all the platforms one wishes to support. However, this means that one would have to learn each platform's languages and APIs independently. Charland and Leroux (2011) give a great overview of the different languages one would need to know in order to create applications spanning most of the popular mobile devices today. In total, they list more than ten different programming languages with additional tools. They also discuss the differences in native code versus web code, and discuss how JavaScript is rapidly gaining popularity and performance comparable to native code because of the browser wars between Microsoft, Google, Apple, Opera and Mozilla. Native code is compiled, meaning that it runs directly on the device's Central Processing Unit (CPU), whereas JavaScript is interpreted. This means that JavaScript and a browser solution will be inherently slower than creating a native application.

When developing a native application you have complete access to the underlying system and its API. Because of security concerns, this is something you will not have access to when developing a web application. However, in recent years, and with the emergence of HTML5, browsers have been given the ability to call

some of these native APIs directly. The ability to upload photos directly from your device, get the geolocation of a device, accessing native map applications and talking to a web server in real time using sockets are among these features (Sowell, 2013).

When developing a web application, one can take advantage of Asynchronous JavaScript and XML (AJAX) programming. Whereas web pages before were static, and you had to load each page, AJAX lets you update your view by getting new information from a web server without having to refresh the page itself. Kessin (2011) and Paulson (2005) explain how JavaScript applications benefit from being able to manipulate the Document Object Model (DOM) on the fly. Combined with how AJAX applications benefit from its asynchronous nature, allowing the web application to feel more native with dynamic updates from the server, this makes for a powerful combination. AJAX applications are usually faster than regular web applications because they typically minimize traffic by requesting only what is necessary for each given task.

Web applications take advantage of the good browser support, but, as previously mentioned, have little support for native APIs. As a result, *hybrid* solutions have appeared. Their runtime environment consists of a web rendering engine which is wrapped in a native engine, allowing developers to create native apps in pure HyperText Markup Language (HTML), Javascript and CSS. This also allows for such hybrid apps to be installed natively on the device, whereas regular web applications must be opened in a browser via a Uniform Resource Locator (URL) (Heitkötter et al., 2013).

For this project, the choice fell on creating a pure web application in HTML5, JavaScript and CSS3. Learning various programming languages to develop natively for multiple platforms did not fit inside the given time frame. Asynchronous application programming with AJAX gave the flexibility of creating a web application that would fit well with multiple devices. It also removed the necessity to distribute the application to the different app stores, which are usually independent per device. In addition to supporting mobile devices, creating an application purely with HTML5, JavaScript and CSS3 will also add support for desktop computers through a web browser.

4.2 HTML5

HTML is the markup language used to structure web sites on the world wide web today. Internet browsers parse the markup language, and displays the web site in a graphical form for the end user. Before HTML5 was finalized in October

2014, it had been 15 years since the previous version, HTML 4.01, was released in 1999. With HTML5 comes a new way of thinking about the markup language itself, and what the name of each tag represents. Semantic tags are meant to represent exactly what the tag itself contains. This not only makes it easier for developers to read the markup language, but also gives search providers, like Google or Bing, a better way of indexing web sites, which improves search results. Video and audio are supported directly in HTML5 as well. (West, 2012).

4.3 CSS3

CSS3 is a style sheet language used to describe how a markup language, such as HTML5, should be rendered visually by a web browser or other markup reader. It is light weight, yet powerful, and comes with some interesting features that have been used to make OsloTUC usable for different devices. Media queries is a module which was introduced in CSS3, and is the foundation for what is known as Responsive Web Design (RWD). RWD allows a developer, or a team of developers, to create a single web site that will scale across devices with different widths and heights (Clark et al., 2012). This has been used to differentiate how elements on the OsloTUC web page are rendered when using different devices.

4.4 JavaScript

JavaScript is a dynamic programming language mostly used to allow computations to take place on the client side of a web page. Recent years have seen an explosion in the use of JavaScript, and web pages are now beginning to feel more and more like desktop applications. With JavaScript one can, for example, manipulate the DOM and create animations, or make asynchronous calls to the webserver, allowing the web page to update dynamically without the need to refresh the entire page. For OsloTUC, JavaScript was used to render the map provided by Google Maps, and to calculate and draw the routes returned when OsloTUC has answered a question. The JavaScript library jQuery[1] was used on the web page, as it simplifies writing JavaScript and making asynchronous calls to the web server.

[1] `https://jquery.com/`

4.5 Creating the web server back end

When users access a web site, they send a request from their browser to a server hosting the site, which in turn interprets the request and sends a meaningful response back to the user. This is called a web server, and there are many different considerations to take into account when choosing server application software. When deciding on a server architecture, the goal was to create a lightweight server application that could serve static files and do simple computations based on input from the user. The web server OsloTUC is serving its web page from was made from scratch in Java utilizing a Java library called NanoHTTPD[2,3]. NanoHTTPD is an open source Java web server that mainly focuses on mobility and portability. It is arguably the smallest fully functional HyperText Transfer Protocol (HTTP) server for Java. It comes with request and session handling, and leaves much of the features presented in other frameworks, such as view templating and control structures, up to the developer. One of NanoHTTPD's core features is its tiny size: the entire library is contained within one Java source code file. This small size makes NanoHTTPD a strong choice when developing a native mobile application, as it can be embedded in the application itself, but it can be used on regular server or desktop applications as well. Because of the simplicity of NanoHTTPD, a URL router had to be designed which would allow for serving the static files such as HTML, JavaScript, CSS and image files.

Figure 4.1 shows an overview of the layered nature of the architecture of OsloTUC's back end. The web application is built on top of NanoHTTPD and Java 1.7. Jasper is a bi-directional interface between programs written in Java and programs written in Prolog[4]. By calling native C libraries on the operating system, using Java Native Interface (JNI), Jasper is able to communicate between the web server and the compiled Prolog code of OsloTUC.

Figure 4.2 shows an overview of the classes found in the back end for OsloTUC. The structure of the architecture and interaction between classes was heavily inspired by the structure of Django[5] applications, a web framework for the programming language Python[6]. Each specific functionality is seen as a module. Serving the home page of OsloTUC is handled in the *index package*, asking a question to OsloTUC is handled in the *search package* and serving static files like CSS and JavaScripts are handled in the *static_files package*. The models

[2] `http://nanohttpd.com/`
[3] `https://github.com/NanoHttpd/nanohttpd`
[4] `https://sicstus.sics.se/sicstus/docs/3.7.1/html/sicstus_12.html`
[5] `https://www.djangoproject.com/`
[6] `https://www.python.org/`

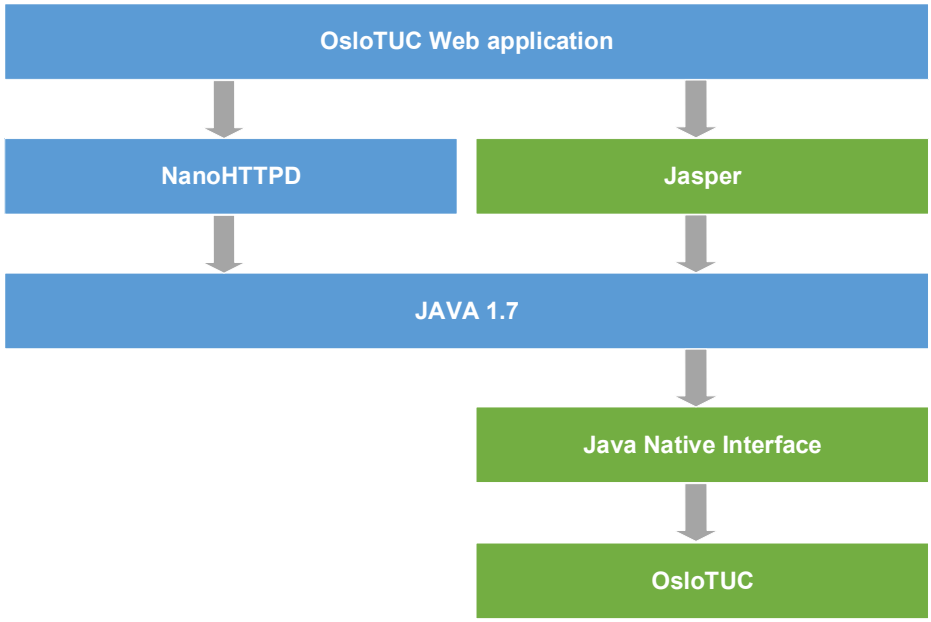


Figure 4.1: Overview of the layered nature of the back end architecture.

TUCAnswer and *TUCQuery* are used to model and format the questions and answers from OsloTUC in order to serve them back to the user via asynchronous calls from the web page. *TucToSicstus* is the class which handles a question from a user, initiating OsloTUC itself, and returns an answer.

The application utilizes the Model-View-Controller (MVC) software pattern. In MVC, the model is a representation of the data or state, the controller handles the interaction between the model and the view, and the view represents the user interface components that use the model to present data (Bass, L, P. Clements and R. Kazman, 2012).

One of the goals when writing the back end was that it should be light and flexible. The web server itself is self-contained within one file and is only 4MB in size, including dependencies. The only system dependencies are Java 1.7 and SICStus Prolog. By utilizing a different compiled version of BusTUC it should work out of the box, without requiring any modifications.

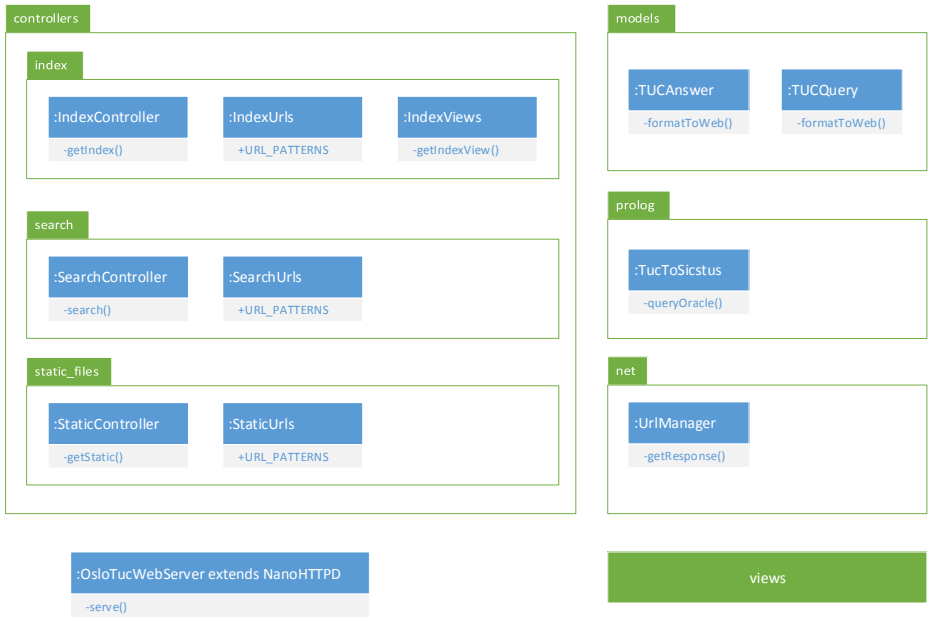


Figure 4.2: Rough overview of the structure and classes used at the back end of OsloTUC.

4.6 OsloTUC front end

Learning from the applications and web sites in the initial background research, the focus was to create a fully functional cross-platform web site with support for different screen widths and heights, as well as to cover some of the functionality that were found appealing in these. The application can be found on `http://vm-6114.idi.ntnu.no:9001/` for the time being, although this might change in the future.

The front end of OsloTUC utilizes Bootstrap v3.3.4 [7], a CSS and JavaScript framework for creating responsive web sites. Bootstrap allows easily creating grid layout which works across devices, regardless of different screen widths and heights. jQuery[8], a JavaScript framework, was used to provide a more modern

[7] `http://getbootstrap.com/`
[8] `https://jquery.com/`

approach to writing the JavaScript code, as well as making it easy to create asynchronous requests to the web server. The front end was built with the intent that users should never have to refresh the web page itself. This goal was accomplished, and all requests running against OsloTUC were made asynchronous.

Taking advantage of the stop names found in the REGTOPP data sets, a feature was made to allow users to search for stop names within the web page while writing queries to OsloTUC. By typing the character '#', users are able to search for stop names found in the knowledge base of OsloTUC. This was intended as a useful feature if users, for example, have forgotten the exact name of a bus stop, or to make writing stop names faster for more experienced users. This feature is shown in figure 4.4 for desktop computers and in figure 4.5b for mobile devices.

The map itself is made available by the use of the Google Maps API. When OsloTUC returns a response from a question, it supplies JSON data to a JavaScript function, which then processes, cleans it, and sends a request to the Google Maps API. The Google Maps API only allows a start point, an end point and 8 waypoints in between per request. Since the response from OsloTUC can contain up to 50 or more waypoints per route, a work-around had to be developed. This lead to the adaption of an algorithm[9] found online. For each 10 points in a route, a request is made to the Google Maps API. The responses of these requests are then combined to present one route. Providing this route to the Google Maps JavaScript enables it to draw routes longer than 10 points long. Each point in the route, representing a bus stop, was then given clickable markers containing metadata about each individual stops. These markers were given distinct colors, according to what type of stop they were, in an attempt to give the users a clear view of what each stop represents. The metadata contains information about what number that stop is on the route, the station name, the bus number, as well as the route's starting and end stations. The address of the station is also shown. Figure 4.3 shows a route being drawn and a marker clicked on a desktop computer, while figure 4.5a shows the route being drawn and a marker clicked on a Samsung Galaxy S4 smartphone. As can be seen in figures 4.3 and 4.5a, point 'I' is not visible, due to it being hidden behind point 'J'. These points represent points where one has to exchange buses. This happens when buses have to be exchanged on the same bus station, and the points may cover each other because they have the same coordinates. A fix for this was not finished before the development had to be finalized.

To help first time users, an 'About and Help' section was added, which can

[9] `https://lemonharpy.wordpress.com/2011/12/15/working-around-8-waypoint-limit-in-google-maps-directions-api/`

be accessed by clicking the 'About and Help' button located at the bottom of the screen. This is true for all devices. Within is located a description of the service itself, how it can be used, and who has been a part of making it.

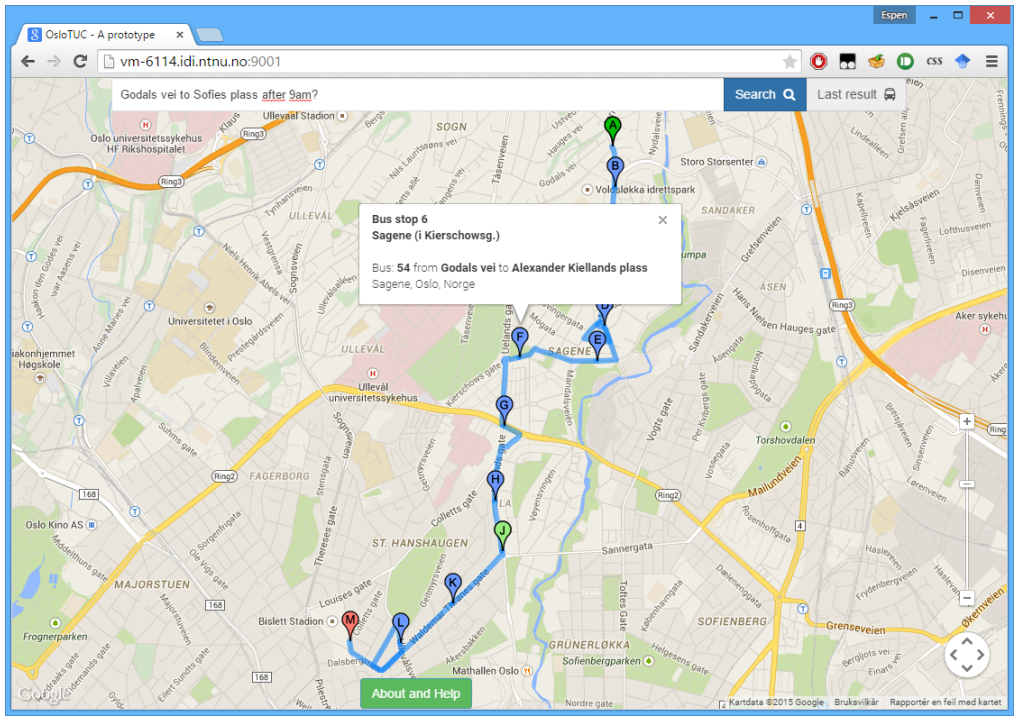


Figure 4.3: A route from 'Godals vei' to 'Sofies plass' drawn on the map on a desktop computer with marker clicked and visible.

The responsive nature of the web site can be seen by comparing the screenshots of figure 4.3 and 4.4, 4.5a and 4.5b. A view of the web page on a Samsung Galaxy Tab is also presented in figure 4.6. On the mobile view, zooming of the web page itself has been disabled, the textbox for asking questions to OsloTUC fills the top of the screen, and the buttons have had their texts removed to leave more room for the question itself. The *About and Help* button has been made smaller, and the section opened when clicking the button is also rendered in a smaller box which will fill the screen. Both the mobile and the desktop view will zoom to fit the route drawn on the map, giving an overview of the route in its entirety.

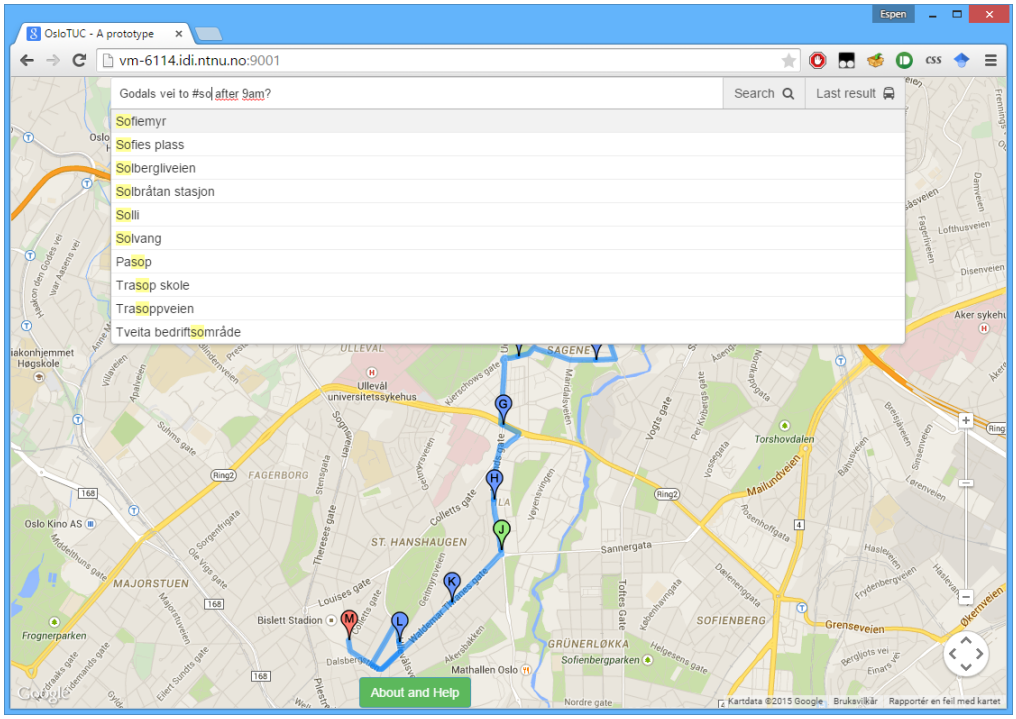


Figure 4.4: #-tag feature shown on a desktop computer providing search functionality for bus stop names.

OsloTUC was tested on some of the latest mobile devices, as well as all the major web browsers found on desktop computers, and was found to work consistently between all of these. A prerequisite for using the web page is having JavaScript enabled in the browser. The test consisted of trying different routes on various devices and browsers, seeing if there were any differences in the behaviour between them. Versions of browsers tested on a desktop computer can be seen in figure 4.1. Versions of mobile devices and browsers tested can be seen in figure 4.2.

All questions asked to OsloTUC are also logged, and in order to view these logs more easily, a separate view was made. This view allows the administrator to see all questions asked on one page, without having to open individual log files as one would in the old system.

Browser	Version
Google Chrome	42.0.2311.90
FireFox	37.0.2
Opera	29.0.1795.47
Internet Explorer	11.0.9600

Table 4.1: OsloTUC tested on different browsers on a desktop computer

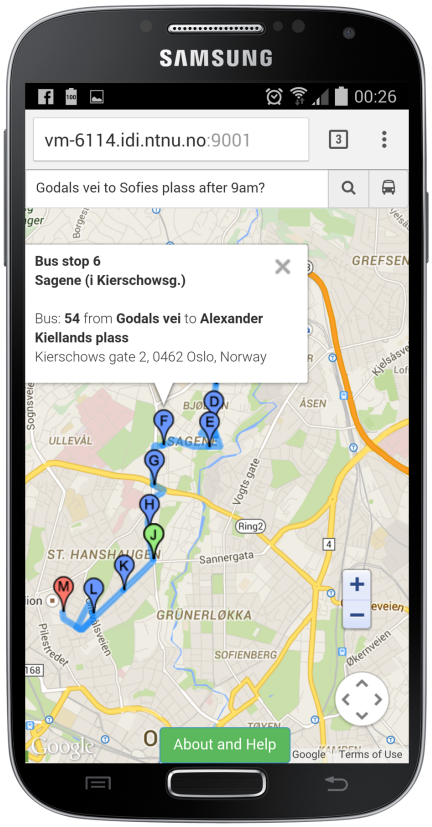
Device	System version	Browser
Samsung Galaxy S3	Android 4.4.4	Google Chrome 42.0.23
Samsung Galaxy S4	Android 4.4.2	Google Chrome 42.0.23
Samsung Galaxy S4	Android 4.4.2	FireFox 37.0.2
Samsung Galaxy S4	Android 4.4.2	Opera 29.0.1809
iPhone 5S	iOS 8.3	Safari 8.0.5
iPhone 6	iOS 8.1.1	Safari 8.0.4
Sony Experia V	Android 4.1.2	Google Chrome 41.0.22
Sony Experia z2	Android 5.0.2	Google Chrome 42.0.23

Table 4.2: OsloTUC tested on different mobile devices

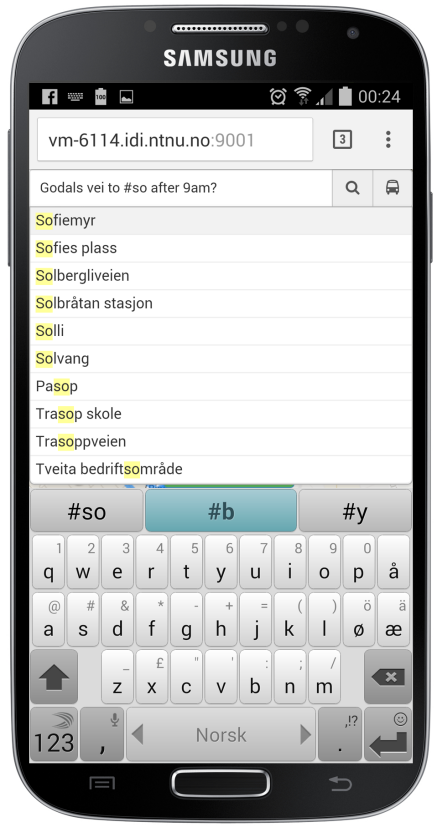
4.7 Limitations

The application uses Google Maps' API for calculating driving routes, which introduces some limitations to the presentation of these routes. First, the API presents some limitations to its use. Users of the free API, which is used in this thesis, may only send 2,500 requests over a 24 hour period, and a maximum of 2 requests per second. Also, only 8 waypoints are allowed in each request, as described in section 4.6. This may introduce errors if the application were to be used by many users at the same time. However, these limitations can be made less significant by using the Google Maps API for Work customers. This version, which is a paid service, allows up to 100,000 requests per 24 hour period, as well as 23 waypoints per request. It also supports up to 10 requests per second.

Second, the route drawn by the API introduces some errors in relation to the driving routes of the buses. The API does not necessarily know that we are asking for a bus route, and thus gives routes as if you were driving a car. Some roads are exclusively available to buses, and are therefore excluded when asking for driving routes. This can lead to very confusing routes being drawn on the map, and be a source of distrust for end users of the application. In figure 4.3 one can see such an erroneous route being drawn between point 'L' and 'M'.



(a) A route from 'Godals vei' to 'Sofies plass' drawn on the map on a Samsung Galaxy S4 with marker clicked and visible.



(b) #-tag feature shown on a Samsung Galaxy S4 smartphone providing search functionality for bus stop names.

Figure 4.5: Screenshots of OsloTUC on a Samsung Galaxy S4 smartphone.

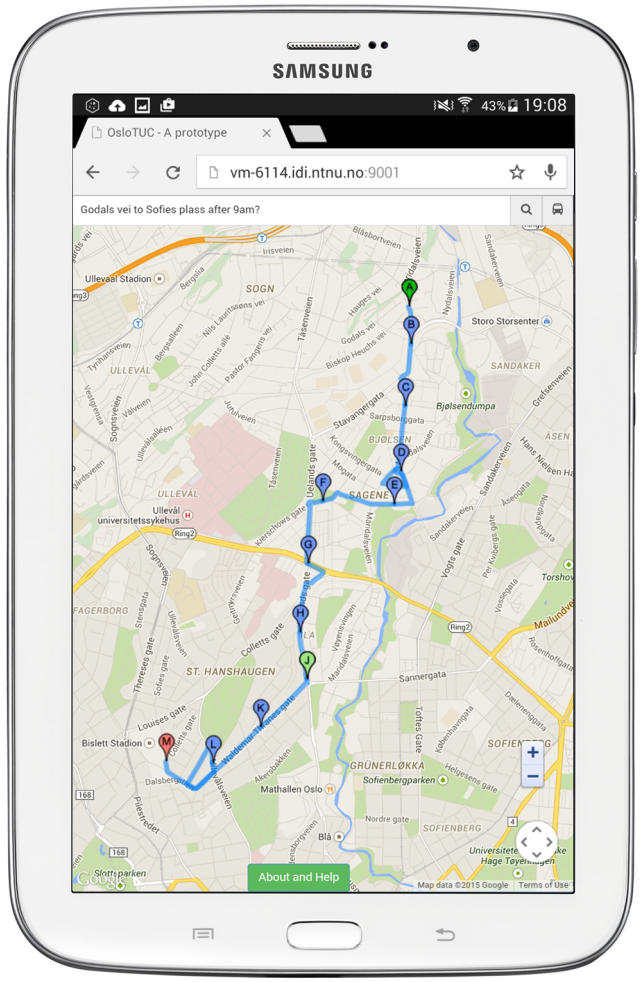


Figure 4.6: OsloTUC showed answering a question on a Samsung Galaxy Tab.

Chapter 5

Research Method

In this chapter the research method used for gathering quantitative and qualitative data will be introduced.

5.1 General method theory

> A quantitative research strategy is based on numerical data, while a qualitative research strategy is based on textual data.
>
> (Ringdal, 2007, page 22)

Research methods are typically separated into either qualitative or quantitative. As mentioned in the paraphrase above, quantitative methods deal with objective numerical data in the form of data points and spreadsheets, while qualitative methods deal with subjective opinions collected in, for example, interviews and/or observations. Where a quantitative research method requires a large number of respondents, a qualitative research method is based on thorough information obtained from a relatively small number of informants. Some researchers base their decision on choosing qualitative or quantitative methods on scientific standpoints, while others have a more pragmatic approach (Ringdal, 2007). The basis for the choice taken in this thesis will be explained later in this chapter.

For the research conducted in this Master's Thesis, the decision fell on multimethod design. This is a combination of the two aforementioned methods of research, enabling the use of both the numerical quantitative data and subjective opinions collected from the respondents. Combining quantitative and qualitative data in such a way is often called triangulation (Brewer and Hunter, 1989; Brannen and Coram, 1995, cited in Ringdal, 2007). One way to utilize this method is

to allow for one of the methods to be subordinated, or by treating both methods equally (Ringdal, 2007). In this particular thesis the qualitative method is subordinated as the quantitative data is more relevant to answer the research questions. The quantitative data is gathered through standardized questions with the option to select one answer. However, the qualitative data gathered through the use of free text input fields is still relevant due to the fact that the respondents can provide meaningful input on their experience of the prototype of OsloTUC, as well as providing new ideas for future work. The assumption is that ideas produced by the respondents in the qualitative part of the questionnaire will be a great pointer to where to go next, and what needs to be changed in order to satisfy the general public.

When conducting research, there are two types of data one can use. Primary data is data gathered by the researchers themselves. Secondary data is data that has previously been gathered in other research. One reason to choose an approach with primary data is that the researchers themselves can tailor the questions according to the research questions of their work. The cons with choosing this approach is that it can be expensive with regards to time and the amount of work needed to be done. By taking the latter approach, the amount of work put into the gathering of data can be minimized, as the data is often both free and available for use. However, by using secondary data, time must be spent on filtering out the data that is useful for the research. In this thesis, the gathering of data has been done by the researcher himself, and it is therefore considered primary data (Ringdal, 2007). The reason for choosing this approach was to tailor the questions in the questionnaire to the research questions of the thesis, as well as the fact that data on this subject has not been previously gathered in any great numbers.

5.2 Respondents

The questionnaire was not designed for a specific demography. The reason for this is that BusTUC and OsloTUC alike have no specific user group. If the demography was to be specified, it would have been as people using public transportation services. This, however, is too broad of a spectrum as nearly everyone uses public transportation from time to time. This makes the general public the demography, which will be used as respondents for the basis of this research. However, an assumption was made that it would most likely be beneficial for the research if the respondents either lived in or had lived in the city of Oslo, which is the geographic center for this thesis. Initially, this was the intended approach, but it was considered too time consuming given the time frame, as the author lives in another city.

5.3 Questionnaire

When your goal is to collect relevant data for later analysis, a data collection method is required. Using a questionnaire is a relatively inexpensive method of conducting data collection from a large number of respondents. The questionnaire is by far the most commonly used method for data collection in social sciences (Ringdal, 2007).

The questionnaire is a systematic and standardized way of gathering data from a selection of a population. A questionnaire can be conducted via telephone, in interviews or via forms that the respondents have to fill in themselves (Ringdal, 2007; Horvat, 2014). The latter approach is chosen in this thesis, as it is less time consuming and expensive to conduct the survey this way, and a greater number of people can be reached in less time. Unlike questionnaires over the telephone or in an interview, the respondents read both the question and response alternatives on their own. This can also add to the anonymity of the process. In an interview, where there is a face-to-face situation, the respondent might feel restrained when giving their answer. In a computer supported questionnaire, the respondent is absolved of any prejudice, and can answer freely without worrying about the reaction of the interviewer. It is still beyond the researchers control whether or not the respondent answers truthfully. However, evidence suggests that respondents do in fact answer more truthfully when doing web-based surveys because of the lack of an interviewer and social desirability bias, which is when the respondents adapt their answers in an attempt to satisfy the interviewer. (Kellner, 2004; Basi, 1999, cited in Brace, 2013).

A downside to questionnaires in web-based form, however, is the number of people who will choose not to answer it. There is no form of control over who has answered the questionnaire or not (Ringdal, 2007). Taking this into consideration, the questionnaire should be sent out to a large number of the population selection in order to get a reasonable amount of answers. A low number of respondents can lead to difficulties in validating the result. Another downside is the inability to clarify any questions the respondents might have if they are unsure of the meaning or the direction of questions in the questionnaire. This puts great strain on the researcher to make the questions “clear, unambiguous and engaging” (Brace, 2013).

5.3.1 Creation of the questionnaire

Google Forms[1] was used as the platform to develop the questionnaire itself. Anyone with a Google account can use the tool by accessing their Google Drive[2], and it is free of charge to use. It being free was a major contributing factor for using this particular tool, as other similar tools either cost a lot of money, or they could not offer the same type of usability and flexibility as Google Forms. After the questionnaire was made, a public URL could be sent to the respondents. The creator of the questionnaire is given a private URL which is needed to edit the document, thus securing the document from external tampering. When the respondents submit their answers, a spreadsheet linked to the questionnaire form will automatically be filled with the submitted data. Each submitted data entry creates a new data point in the spread sheet. This allows for easily handling the data later, or exporting it to a more advanced spreadsheet program, like Microsoft Office Excel[3].

5.3.2 Distribution

The questionnaire was distributed via social media because it would allow for efficiently reaching a greater number of respondents, compared to physically going into the field, approaching possible respondents and getting them to partake in the survey. Some research use traditional mail to distribute questionnaires, but that would have been very time consuming. Another reason for distributing via social media, is founded on the assumption that consumers of social media are more likely to use technological tools for finding bus schedules than potential respondents who do not use social media.

5.3.3 About the questionnaire

The initial part of the questionnaire is a general description of OsloTUC and an explanation that it is a system that allows users to ask standard questions in both Norwegian and English for obtaining travel information. The purpose and focus of the questionnaire are also explained. The focus is on how a user prefers to ask about travel information as well as how users want to view the results of their query. A justification of the limitations is given to prepare the respondent for the user cases to be performed before answering the questions of which the questionnaire comprises. An approximate time frame of 10 to 15 minutes is then given. All data is collected anonymously to protect the identity of the respon-

[1] `https://support.google.com/docs/answer/87809?hl=en`
[2] `https://www.google.com/intl/en/drive/`
[3] `https://office.microsoft.com/en-gb/excel/`

dents, and to prevent potential bias in the analysis of the results.

User tasks are then given to be performed on the web sites of OsloTUC and Ruter.no. The reason for doing the tasks on both web sites is to give the respondent a basis of comparison between natural language search and a search by filling in predefined boxes. The tasks are chosen carefully to demonstrate how the respondent can perform natural language text queries to the system as well as performing advanced queries containing date and time information. The tasks are designed to give the respondents a visualization of how route information is displayed once the query has been answered, and to give the respondents a feel for how the application is used in a general sense.

The second part of the questionnaire will be used to gather information to answer research question 2, and the third and last part will be used to answer research question 3, found in chapter 1.3.

First, the questionnaire itself asks general questions about the respondent, such as gender, age and residential city. It is of interest to know if the respondent resides in Oslo, or has knowledge of the city, as this will give a more related result, given the geographical domain of OsloTUC.

Second, questions about the presentation of the answers to a user query are presented. The main focus in this part is on the general visual interpretation of the route information and bus stops, as well as the textual answers given. The answers to the questions are given in the form of multiple choice graded options, ranging from *completely disagree* to *completely agree*. For some of the questions, the respondent has the option to provide additional information in the form of textual answers. The reason for giving the respondents an option to answer with text is because it could provide more valuable information to a question where only predefined choices are made available.

Last, the respondents are asked how they prefer asking a travel planner about travel information, and what kind of sources they use to gather such information. General questions about OsloTUC are also asked. The answers here could help figuring out what must be done in the future to make OsloTUC a viable travel planner option in Oslo.

In general this questionnaire aims to answer the research questions, stated in chapter 1.3, related to the design and user questioning process, which will hopefully provide grounds for further work on the expansion of BusTUC to other cities. The data points provided by the respondents may indicate what poten-

tial users of the application prefer when it comes to the display of the returned answers and how the users prefer to ask questions to a travel planner. This is interesting to determine, as most applications today, at least in Norway, do not use natural language processing.

Seeing how OsloTUC is a prototype of how such a system could be realized, it would be relevant to figure out whether or not users outside Trondheim like using natural language. This could be a motivator to go through with the idea of expanding BusTUC to other cities. The author hopes that the results from the questionnaire can be of value for furthering developing the system.

Chapter 6

Results

This chapter will present the results of the questionnaire. For every question given to the respondents, a summary will be made of the answers and numbers presented. The questionnaire was given to the respondents in Norwegian, but for the purpose of keeping this thesis in English, the questions have been transcribed from Norwegian to English.

6.1 Questionnaire

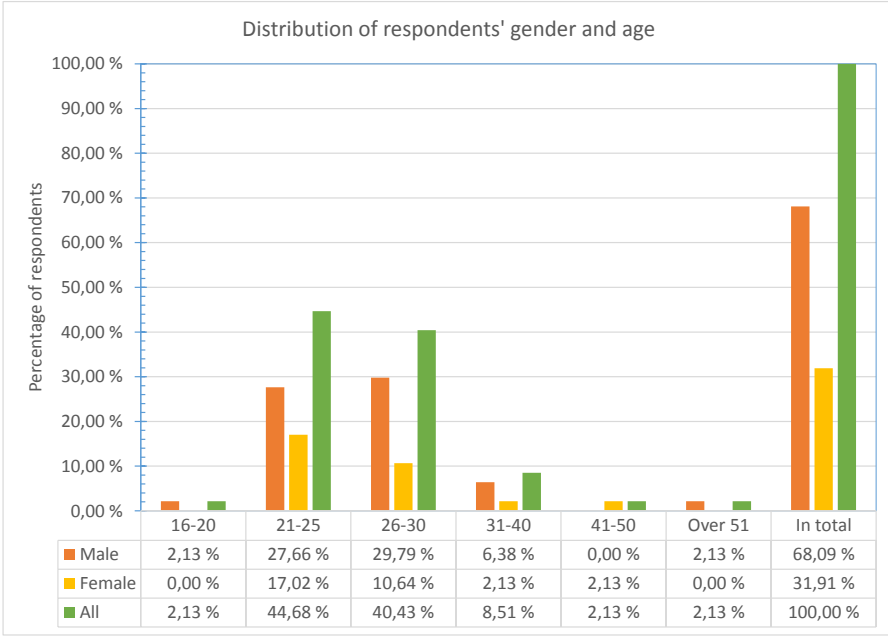
6.1.1 Distribution of respondents

In total, 47 respondents performed the questionnaire, with ages 21-25 and 26-30 being the strongest represented age group. Before further presentation of the analysis of the results of the questionnaire, the raw data points of anonymous personal information for the respondents are presented in table 6.1.

	16-20	**21-25**	**26-30**	**31-40**	**41-50**	**Over 51**	**In total**
Male	1	13	14	3	0	1	32
Female	0	8	5	1	1	0	15
All	1	21	19	4	1	1	47

Table 6.1: Distribution of age and gender in numerical data.

The number of respondents, their age and their gender have here been presented in a numerical format. Subsequent graphs will be presented with percentages, as this type of data is more informative when viewing it statistically. In figure 6.1, the data points in table 6.1 are presented as percentages of the grand total. All figures have been rounded to the nearest two decimal places.



	16-20	21-25	26-30	31-40	41-50	Over 51	In total
Male	2,13 %	27,66 %	29,79 %	6,38 %	0,00 %	2,13 %	68,09 %
Female	0,00 %	17,02 %	10,64 %	2,13 %	2,13 %	0,00 %	31,91 %
All	2,13 %	44,68 %	40,43 %	8,51 %	2,13 %	2,13 %	100,00 %

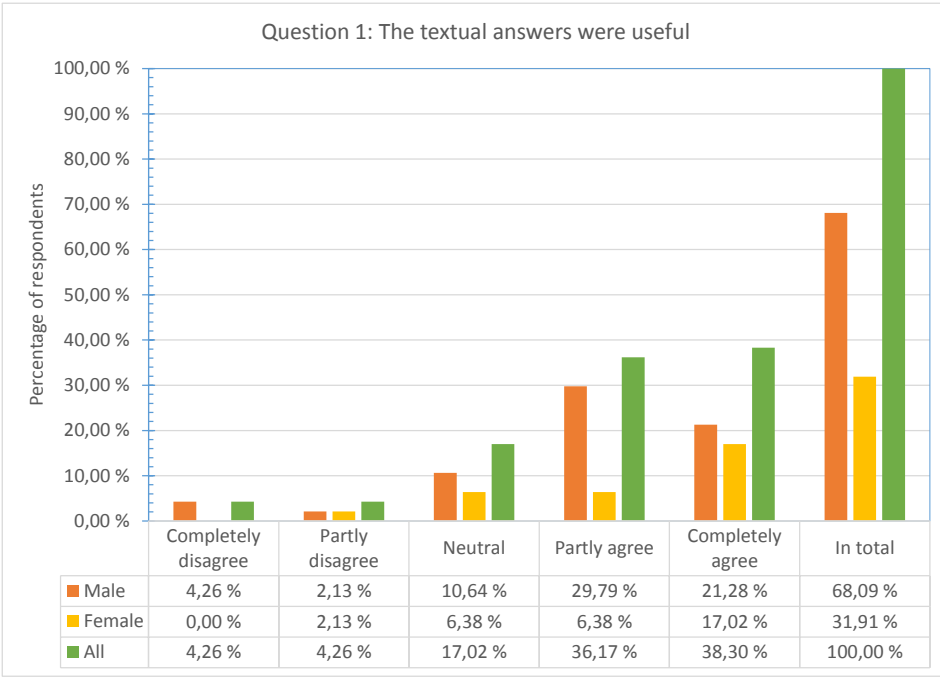
Figure 6.1: The distribution of the respondents' gender and age

It was interesting to know whether or not the respondents were situated in Oslo, if they had lived there, or if they had any experience with the public transportation there. 42.55% of the questionnaire's respondents were situated in Oslo, while 63.83% reported to be familiar with Oslo and its public transportation system. 57.45% of the respondents lived in another city, with Trondheim being the best represented city. The application is not uniquely useful for inhabitants of Oslo, as it can be used by people visiting Oslo as well. For people travelling to Oslo, such an application could be useful, if the users prefer natural language over predefined boxes.

6.1.2 The questionnaire

Question 1: *The textual answers were useful*

4 out of the 47 respondents (8.52%) answered that they completely or partially disagreed with this statement. 8 (17.02%) were neutral, while 17 respondents (36.17%) partially agreed, and 18 respondents (38.30%) completely agreed. Although some respondents did not find it useful, the majority seemed to see the practicality or advantage in having the answers presented in this manner, or that the information they got was sufficient.



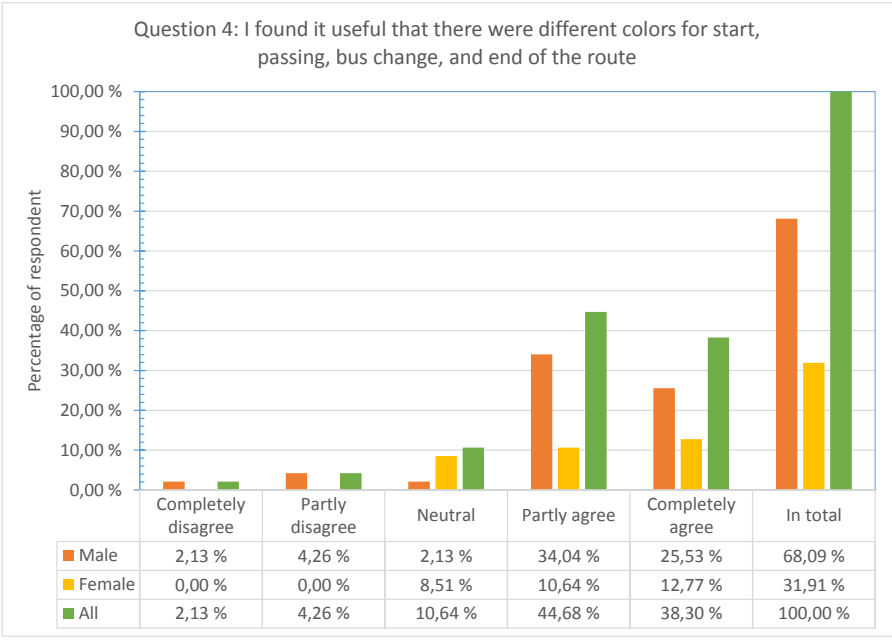
	Completely disagree	Partly disagree	Neutral	Partly agree	Completely agree	In total
Male	4,26 %	2,13 %	10,64 %	29,79 %	21,28 %	68,09 %
Female	0,00 %	2,13 %	6,38 %	6,38 %	17,02 %	31,91 %
All	4,26 %	4,26 %	17,02 %	36,17 %	38,30 %	100,00 %

Figure 6.2: Results of Question 1: The textual answers were useful

Question 2: *It was useful to have the route drawn on a map*

A total of 3 out of 47 respondents (6.39%) partially or completely disagreed with this statement, while 1 (2.13%) was neutral, and a total of 43 respondents (91.49%) partially or completely agreed that they found this to be useful. This clearly shows that the respondents prefer having this kind of information presented to them when they seek out route information.

An early hypothesis made was that users would benefit from such an application when travelling to Oslo. In question #3, which was an open question for commenting question #2, three respondents noted that such a feature would be positive because they were not familiar with the city. Displaying the information in a map would allow them to know where to go.



	Completely disagree	Partly disagree	Neutral	Partly agree	Completely agree	In total
Male	2,13 %	4,26 %	2,13 %	34,04 %	25,53 %	68,09 %
Female	0,00 %	0,00 %	8,51 %	10,64 %	12,77 %	31,91 %
All	2,13 %	4,26 %	10,64 %	44,68 %	38,30 %	100,00 %

Figure 6.3: Results of Question 4: I found it useful that there were different colors for start, passing, bus change, and end of the route

Question 4: *I found it useful that there were different colors for start, passing, bus change, and end of the route*

8 out of 47 respondents were neutral or disagreed to some degree with this statement (completely disagree: 2.13%; partially disagree: 4.26%; neutral: 10.64%), while 39 respondents (82.98%) either partially or completely agreed with it. This indicates that users would prefer to have different color codes for different parts of the journey they are planning. There were several comments about the use of colors in the map. One respondent stated that it would be better to have each part of the bus route drawn in separate colors, instead of coloring only the stops.

Another respondent stated that it was difficult to distinguish the different color codes due to color blindness.

Question 5: *It was useful to be able to see the textual answer once more after having closed it for the first time.*

0 respondents completely disagreed, and 2 out of 47 respondents (4.26%) partially disagreed with this statement. Although 7 respondents (14.89%) were neutral, 13 respondents (27.66%) partially agreed, and 25 (53.19%) completely agreed that they found this to be useful. The fact that 80.85% of the 47 respondents found this useful, may imply that they would prefer a source of information which presents the route information in such a manner.

Question 6: *The information given when clicking on a bus stop was sufficient*

0 out of 47 respondents completely disagreed, 5 (10.64%) partially disagreed, 7 (14.89%) were neutral, 19 respondents (40.43%) partially agreed, and 16 respondents (34.04%) completely agreed with this statement. In question #7, which was an open question asking the respondents to comment on question #6, some respondents gave constructive criticism. Two respondents stated that the information should contain how many minutes the bus would use from the starting point to the bus stop they clicked. Another respondent noted that it would be useful to see which other buses were departing from the stop they had clicked. Other mentions were direction of the bus, and more metadata per bus stop.

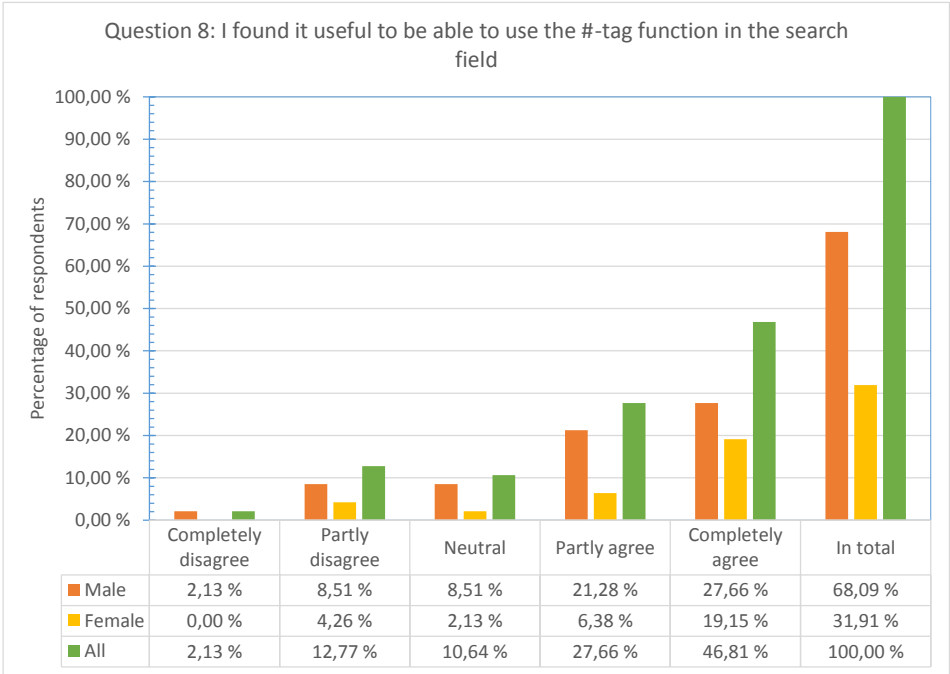
Question 8: *I found it useful to be able to use the #-tag function in the search field*

1 out of 47 respondents (2.13%) completely disagreed, 6 (12.77%) partially disagreed, 5 (10.64%) were neutral, 13 (27.66%) partially agreed, and 22 (46.81%) completely agreed that they found this to be useful. The results from the majority of the respondents show a tendency towards a preference for such a manner of auto completion in case the user does not remember the full name of the bus stop.

Question 9: *Could the answers from OsloTUC have been presented in another way?*

This question was an open question. 6 respondents had comments about the #-tag functionality. Half of them noted that they think this functionality should be automatic without the need of using the '#' symbol. Another respondent stated that the functionality was minimal, as there was too little information being displayed for each bus station in the results list. One respondent had opinions about

whether one should use the symbol '#' or '@', stating that Facebook and Twitter uses the '@' sign for identifying names, and the '#' symbol for identifying topics.



	Completely disagree	Partly disagree	Neutral	Partly agree	Completely agree	In total
Male	2,13 %	8,51 %	8,51 %	21,28 %	27,66 %	68,09 %
Female	0,00 %	4,26 %	2,13 %	6,38 %	19,15 %	31,91 %
All	2,13 %	12,77 %	10,64 %	27,66 %	46,81 %	100,00 %

Figure 6.4: Results of Question 8: I found it useful to be able to use the #-tag function in the search field

Additional graphs of the results from the questionnaire, and the questionnaire itself, can be found in the Appendices.

A short summary of the most important results

1. The respondents showed a tendency to prefer natural language over filling in predefined boxes.
2. The questionnaire showed signs that the respondents preferred having the route displayed on a map over textual answers.
3. Although more varied than some of the rest of the questions, being able to search for bus stops in real time with the #-tag feature seemed preferable.
4. Some respondents wanted more metadata for travel information than was provided by OsloTUC.

Chapter 7

Discussion and Conclusion

In this chapter, the results presented in chapter 6 will be discussed. A conclusion will then be drawn based on the results and the discussion. With the results as a foundation, the research questions will then be evaluated and answered.

7.1 Questionnaire

The questionnaire revealed some interesting findings. The demographic represented by the respondents generally suits the expected user group of an application such as OsloTUC. The age spectrum was on par with the desired results, as was the results of where the respondents were situated and how familiar they were with the public transportation system in Oslo. Nearly half of the respondents were situated in Oslo, which gives valuable information needed for a discussion and drawing a conclusion on whether to expand BusTUC to Oslo. The results also give good grounds for answering the research questions presented in section 1.3.

Looking at the questionnaire as a whole, the responses were generally positive. On the questions that gave the respondents a chance to evaluate a statement, using a ranking of *totally disagree* to *totally agree*, over 70% of the respondents gave a rank of *partially agree* or higher. This can be interpreted as being favorable for the existence of OsloTUC. These particular questions were also among the most relevant for answering the second and third research question.

BusTUC has historically only given textual answers as a basis for communication with the user. But seeing as nearly all the emerging travel planners, including the newer applications in the FUIROS project, now feature a map, one

may ask if textual representation of the answer really a necessity? One of the questions the respondents were presented with were directly linked to this case. Nearly 75% of the respondents thought that the textual answer was a useful or helpful feature of OsloTUC, which supports one of the main features of BusTUC. This is one of the features of BusTUC that separates it from the rest of the existing travel planners. Having a structured sentence returned upon asking a structured question seems to be in favor with the respondents. One could suspect that the negative rankings were given by respondents who already were familiar with travel planners in Oslo. However, judging from the data set, this was only the case for half of the ones who ranked it negatively. The vast majority of respondents also ranked being able to view the textual answer again once it was closed positively, which again underscores the usefulness of a textual answer.

As previously mentioned, most of the emerging travel planners today evolve around the functionality of a map. So why is it that users seem to prefer maps over text when using a travel planner? One respondent brings forth the seemingly obvious answer, and states in a comment that one often needs to see exactly *where* the departure or arrival point is, so that they more easily can navigate to that position. This is something you would normally not get from a textual system, which would only tell you the name of the stop, and not where to go to get there. 6.39% of the respondents did not see the practicality of having the route drawn on the map, whereas 91.49% were positive towards the way the route was drawn. The choice when drawing the route was either from *point A* to *point B*, or by using Google Maps to display the accurate driving route. The respondents seem to be satisfied with the accurate driving route, although this question might have been answered differently if they were given the option to display the route point by point. Notable here is that Google Maps will not always draw correct driving routes, due to some streets being open to buses or trams only. It still seems like it gives a satisfactory approximate route for most routes, but in some cases the calculation from *point A* to *point B* from Google Maps is somewhat off due to this aforementioned reason.

Furthermore, using a map allows for the customization of how the route should be rendered. With Google Maps one can create custom markers and custom lines on the map. The solution chosen was to distinguish the markers for point of departure, stops along the way, and point of arrival. These were distinguished by using different colors. When the respondents were asked how they perceived this feature, they were overly positive to the use of colored markers to present the points of interest. However, one respondent commented that he/she was colorblind, which made it very hard to distinguish between the departure and arrival point. This is a major concern for, not only the usability of the application, but

also the accessibility. Users with disabilities could possibly feel excluded from using the application. There are regulations according to Norwegian law (Forskrift om universell utforming av IKT-løsninger, Lovdata, 2013), that states that every publicly accessible web-site created after June 1. 2013 should meet the criteria of *Universal Design*[1], meaning that it should be accessible for everyone. It also states that web solutions should at a minimum be designed with the Web Content Accessibility Guidelines 2.0 (WCAG2.0)[2] standard on level A and AA. This should not be viewed as making it harder to create and design web-sites and applications, but more as an assurance that you are implementing quality.

Using markers on the map also gave the ability to show information about the bus stops if they were clicked on. This functionality was used to display information about bus stops. The data used for this information was data that came from OsloTUC's answer. The question about whether or not the information in these markers were sufficient was one of two that got the most varied response. Although 74.47% answered positively, 14.89% were neutral and 10.64% partially disagreed. When looking at the comments from the respondents, they not only describe why they thought so, but also gave constructive feedback as to how it could be changed, or described other features they would have wanted instead. Although the markers on the map are meant to be to the point, some respondents indicated that they wanted more information. The question then becomes: what is too much information in such a small space, and should the feature be redesigned? Allowing too much information in the marker boxes will take up considerable space, covering the map itself. Too little information, and some users will be dissatisfied. Another solution could be to have a dedicated area for such information. Among the feedback from the respondents were suggestions like the ability to view other departures from the same stop, information about how much time the bus uses to travel to that particular stop, and estimated time of arrival.

To shorten the gap between a purely textual travel planner and travel planners with defined boxes for departure and arrival, the functionality of searching for bus stop names with the #-tag feature in real time was added to OsloTUC. Users who come from a purely predefined boxed solution would perhaps be less inclined to use a purely textual service if it did not have a way of filling in the bus stop names automatically. When venturing into a market already filled with existing solutions, it seems important to try to make a product seem familiar to possible new users. The #-tag feature provides functionality for searching for

[1] `http://standard.difi.no/artikler/2010/02/krav-til-utforming-av-nettsteder-i-forhold-til-referansekatalogen#uu`

[2] `http://www.w3.org/TR/WCAG20/`

bus stop names if you are unsure of exactly what the place you were going to or from is called. The respondents seemed to agree that this was a nice feature to have in OsloTUC. 74.47% of the respondents were positive towards this feature, while 25.54% were negative or neutral. Judging from the comments about the #-tag feature, one can assume that they were not particularly against the feature itself, but how it had been implemented. Several respondents commented that the searching for bus stop names should have occurred automatically while writing the question, without the need of having to type the '#' sign in front of a station name.

Learning from the implementation of the bus stop search feature, it was later realized that OsloTUC has a greater potential for serving metadata than originally intended when implemented. By taking advantage of the REGTOPP data sets, OsloTUC as an application can do a great deal more than just serve questions when asked. The REGTOPP data sets can be mined for, for example, bus numbers, bus stops with names and coordinates, and scheduled times. This would allow the web server to serve this content statically, without going through a lexical query to OsloTUC, thus making it able to serve such metadata much more quickly. A practical example could be to load all bus stops in a given area, displaying them with markers and metadata on the map with a single request to the web server. One might argue that such features would be outside the scope of OsloTUC and BusTUC as a lexical parser, but it would provide a much richer experience for the end user, and it would more than likely alleviate the server from running as many Prolog requests.

The respondents were asked if the answers from OsloTUC could have been presented in another way. Some of the respondents suggested that the results could be displayed in a list form instead of free text. This was originally implemented in an earlier version, but it was thought to be hiding too much of the visible map area, particularly on mobile devices. It was implemented as a section that could slide in and out of the side of the screen with the click of a button, but due to mobile devices' sensitivity to outer areas on the screen, it was finally replaced by a modal dialog. Another respondent suggested changing the #-tag feature to instead use the '@' sign. The reason for this was that both Facebook and Twitter use the '@' sign for names, while the '#' sign is used for categories. This makes more sense, as the feature looks up street names and addresses. Seeing as Facebook and Twitter are two of the most widely used social media sites, it is likely that users would feel comfortable using the '@' sign. As previously mentioned, some of the respondents requested the use of more metadata per marker on the map, and one respondent suggested that times for stops should be included.

Another interesting question asked to the respondents, which was directly relevant to one of the research questions, was how they usually look up public transportation schedules. This was a multiple choice question, and the results were quite pleasing from a technological standpoint. Only 12.76% reported that they still use printed schedules, while 72.34% reported using an app on their mobile device, and 48.93% using a web site. *'Other'* was given as a choice, and was chosen by 12.76% of the respondents. This indicates that most travellers today are either on the move when looking up transportation schedules, or just want it readily available where ever they are. They also seem to use a web site if they are situated in front of a stationary electronic device. Web sites are generally inherently usable for stationary computers, while they are fitted to be usable on mobile devices. The decision to make OsloTUC a mobile friendly web site thus seems like an excellent choice. Only two of the respondents reported exclusively using printed schedules, which means that OsloTUC will cover the needs for 95.74% of the respondents participating in the questionnaire. 40.42% reported exclusively using an app on a mobile device, which again underscores the importance of mobile friendly web-sites today.

Finally, the respondents were asked if they would have used OsloTUC if it had a greater support for the variety of public transportation which Oslo offers. Here, again, we see a positive trend. Over 70% of the respondents answered that they would have used OsloTUC, even though it currently only has support for buses. This could either indicate that the respondents only travel by bus, or that they are pleased with how the system already works. However, nearly 30% answered that they would *not* have used the system in its current state. This can be interpreted as an indication that the respondents do not have a need for such a system, or that they find it lacking in features. Perhaps they are already satisfied with the existing solutions, and do not want to write whole or partial sentences in order to find travel schedules, or simply do not wish learn a new system altogether.

The last question asks whether or not the respondent would use OsloTUC if it supported trams, trains and subways. Here, only 6.38% answered that they would still not have used it. The last question was answered by 25 respondents, even though it specifically states that it should only be answered if having answered *no* to the previous question, which was done by only 14 respondents. However, the only relevant information here comes from the respondents who answered *no* to both of the questions, and those who had changed their answers between the two. Not only would OsloTUC be a better product if it had support for trains, trams and subways, but it would quite possibly attract more users, which in the long run must be seen as a goal.

7.2 Research questions and goals

In the initial stage of the development of OsloTUC, research questions and goals were formed to cover the scope of the project. The questionnaire was then meant to answer these questions.

7.2.1 Research Question 1:

What is required when moving a well established NL travel system to a new city?

A good understanding of the underlying system, as well as travel schedules and patterns in the new city is required. In addition, great insight into the different travel options and their interconnectivity must be acquired. Data sets and parsers of data sets must be robust and maintained when new versions arrive. Being able to provide support for addresses with associated coordinates is vital in order to make the application more versatile for the end user. The exterior system exposed to the end user should be modern and provide cross-platform support in order to attract a significant and lasting user base, which today are people constantly on the move.

7.2.2 Research Question 2:

How do users prefer travel information to be displayed?

The questionnaire gave an indication that users prefer both textual answers given in structural form, as well as a map providing them with an overview of points of interest and the planned travel route. In addition, they seem to require sufficient metadata for possible and alternative routes, although the differences in the need for metadata is varied. Clear indication of what the differences between the various points of interest are seems to be favorable. Not implemented, but highly requested, was having the information displayed in the form of a list, providing the users with a clearer overview of the stops.

7.2.3 Research Question 3:

What is the preferred way to ask about travel information?

The questionnaire revealed that the respondents were readily inclined to write natural language queries to a travel planner instead of filling in predefined boxes for departure and arrival points. The respondents seem to prefer using a mobile or otherwise electronic device to ask such queries, and it is apparent that they want to spend as short amount of time as possible figuring out how they need to travel, and where they need to travel from, in order to get from point A to point B. Familiar from services providing predefined search boxes is the feature for autocompletion of stop names. The respondents welcomed the feature of the #-tag search functionality, but wanted to be able to autocomplete stop names without having to write a trigger character.

7.2.4 Goal 1:

Modify BusTUC to enable queries for the city of Oslo to find answers for RQ1. Add new data sets from Ruter to prove that it works.

BusTUC was modified to support additional data, and schedule data sets provided by Ruter was parsed and embedded as the knowledge base for bus routes. Tools were created to extract street addresses and gather coordinates for these, making BusTUC fully functional for a subset of routes in the city of Oslo. This gave the foundation for creating a web server and GUI for BusTUC and the system was then named OsloTUC. The REGTOPP parser was updated to support newer versions of REGTOPP.

7.2.5 Goal 2:

Develop a proof of concept application presenting answers from OsloTUC in a map solution using the Google Maps Directions Service API. The application should perform well on mobile devices and in web browsers. Test the application with real users to answer RQ2.

A web server and a GUI was created with up-to-date technology, supporting all mobile devices as well as stationary desktop computers. Google Maps API was used to support map functionality and to draw routes on the map. Markers with route metadata was made clickable, and provided additional information

for the interface. Real time search functionality for bus stop names was implemented statically from the web server without having to ask a query to OsloTUC.

7.2.6 Goal 3:

Perform a survey with real users, asking them to test the proof of concept application and comparing it to existing solutions from Ruter in Oslo. It is desirable to draw conclusions on how users ideally like to query or ask questions to a travel schedule engine (RQ3.)

A questionnaire was administered, and completed by a total of 47 respondents. The respondents compared how asking questions differed on Ruter's web site and in the proof of concept application created for OsloTUC. The respondents were instructed to ask the same questions on Ruter's web site and in OsloTUC to give sufficient grounds for comparison between the two travel planners.

7.3 Conclusion

Travel planners exist for all major cities in Norway today. BusTUC brings an edge with its capabilities to understand natural language queries. So far, BusTUC has only been deployed on a wide scale in Trondheim, but this project has shown that it is possible to create a modern application for daily use in other cities as well. The ability to have one solution work on multiple platforms, both stationary and mobile devices, is a feature we see increasingly more in products emerging today. As seen in section 2.1, many of the newer tools provide multi platform support. Some provide one solution for multiple devices, whilst others provide multiple solutions for multiple devices. The trending topic here is to cover a broad spectrum of the population by supporting multiple platforms.

The process could have been handled differently. Focus should have been directed towards making full support for schedules and addresses in Oslo at an earlier stage. As this was not the case, it resulted in the questionnaire being performed at a late stage in the process. To fully back the data collected in the questionnaire, a second questionnaire should have been performed on a second group of respondents. This would have provided more concrete evidence to support the claims made in the thesis.

Overall, OsloTUC seems to provide the same functionality in Oslo as BusTUC does in Trondheim, which was the intention. Instead of only having a textual interface, users in Oslo can now visualize the routes in a map across all devices. However, if there had been more time for the development of the GUI, it would have been positive to add additional features, making OsloTUC stand out from the masses.

The results of the questionnaire gave first hand input on what worked nicely, and what could have been done differently. They gave good a indication of the needs and expectations of users, and this could be used to tailor the experience further if OsloTUC, or another variant of it, is to be refined and further developed.

To conclude, the questionnaire gave satisfying results, and all of the goals were fulfilled. It is the author's wish that the contributions of this project might prove to be useful to the FUIROS project, and that it might help future implementations of BusTUC in Oslo or other Norwegian cities - or even other parts of the world. Additionally, the results from the questionnaire can hopefully be used as a reference to create a clear and user friendly interface in the future.

Chapter 8

Future Work

This chapter will present further features that could be implemented in OsloTUC. Suggestions for how BusTUC could be structured to better be suited for modularization to one specific city, as this thesis has covered, will be presented.

8.1 Universal design

It was a little saddening to see some of the results of the questionnaire. One of the respondents did not get to experience OsloTUC in its intended state, and commented on having problems seeing the different colors used in highlighting the departure, passing and arrival stops, due to color blindness. This lead to the conclusion that OsloTUC should be made accessible to everyone. Sadly, this could not be implemented before the thesis came to an end. Since it is required by Norwegian law (Forskrift om universell utforming av IKT-løsninger Lovdata, 2013), OsloTUC should be developed in accordance to the WCAG2.0 standard. This will ensure the accessibility for users both with or without disabilities. Direktoratet for Forvaltning og IKT, The Norwegian Agency for Public Management and eGovernment (Difi)[1] is responsible for making sure that the requirements of universal design are met, and they have excellent guidelines on how to meet with the standard on their web sites[2].

[1] `http://difi.no`
[2] `http://uu.difi.no/`

8.2 Statically served data

OsloTUC has a greater potential than just serving answers to questions asked to its engine. Even though this is primarily a natural language system, many things can be simplified for the end users. Serving answers from OsloTUC is done dynamically as each question arrives, but there are other kinds of data that could be served statically. To give an example, all bus stops in a bounded area can be served with coordinates, stop names and various other metadata. This could be done in one single request to the web server without having to parse an entire sentence, thus making the response much faster. This response could in turn be used to plot all available stops on the map when the user loads the application or navigates to different portions of the map. Furthermore, the next scheduled route could be fetched just by clicking on a bus stop if the REGTOPP data sets were also parsed and made available to the web server itself. This would likely bring down the server load as one would not have to perform a prolog query for quick and simple questions that instead can be performed by the click of a marker.

8.3 Real time support from Ruter

The trending topic in todays big cities, at least in Norway, is the emerging use of GPS tracking of transportation, and the ability to produce real time data information by using these. When using public transportation in Oslo, the widespread usage of real time systems is immediately apparent. Both on buses, trains, the tram and on subways you can see exactly when the transport will be arriving at its destination. OsloTUC can benefit from this by taking advantage of the publicly available real time API exposed by Ruter. This could be implemented by the web server and, upon a user asking a question, be embedded into the returned result. By combining this real time feature with the statically served data suggested in the section above (8.2), additional features could be presented. Implemented this way, a user could click on a marker representing a station and get immediate results for real time information for that particular station.

8.4 Adapt BusTUC to support routes with overlapping dates

There were some problems during the phase where BusTUC was modified, that, in the end, made OsloTUC lack support for trams, trains and subways. Initially, this was thought to be a simple task, on par with embedding the knowledge base for buses. It proved to be difficult to include scheduled routes that had

overlapping periods of time. It was attempted to import two data sets, containing subway and bus schedules, respectively. BusTUC would only parse answers from one of these, namely the one that was included first. This resulted in the choice of picking either the bus schedule or the subway schedule as a basis for OsloTUC. The bus schedules were chosen on the grounds that they provide more complexibility to the application itself. Oslo has a complex public transportation system. Many of the routes rely on different types of transportation when travelling from point A to point B. These routes can be dependent on the exchange from, for example, bus to tram, or train to subway. In order to fully support this complex public transportation system, this issue must be addressed.

8.5 Supporting different transportation types

During the course of this thesis, the REGTOPP data sets provided by Ruter changed versions from 1.2 to 1.3.A. This change results in changes to existing solutions of importing the data sets to Prolog code. A modification of the REGTOPP to Prolog converter was necessary. During the modification process it was discovered that the REGTOPP format contains a field for which type of transportation the current route represents. This field is not utilized by BusTUC today, and could be the source of vast improvements and simplification of the extension of other types of transportation. Instead of implementing specific fixes for each type of transportation, this field can be used to let the user know by which transportation type he or she will be travelling.

For future references: In REGTOPP version 1.1D* the field is known as *Trafikkart* in the file *TURIX.TIX*. In version 1.2 the field is known as *Trafikktype* in the file *TURIX.TIX*. In version 1.3.A the field is known as *Transport mode* in the files *TURIX.TIX* and *Connection Protection* (known as *Samtrafikk* in the older formats).

Further documentation of the REGTOPP format can be found on trafikanten.no's website[3].

[3] `http://labs.trafikanten.no/2011/4/13/dokumentasjon-av-regtopp-formatet.aspx`

8.6 Modularizing the knowledge base

Today, BusTUC has *one* knowledge base for buses. In order to move the travel planner to Oslo, the existing knowledge base had to be modified. Synonyms for street names found in both Trondheim and Oslo had to be removed, but there are still fragments left, possibly making queries to OsloTUC erroneous. The author suggests modularizing the knowledge base itself, moving it to its own directory based on a geographical name. This way, a totally new knowledge base could be generated without the need to worry for, for example, overlapping addresses or bus stop names.

To be able to create OsloTUC, a fork had to be made from the code base of BusTUC. This has resulted in the two code bases diverging from each other. By using one code base, with modularized knowledge bases, the underlying system would not need to be changed if support for a new city was to be added.

Bibliography

Amble, T. (2000). BusTUC: a natural language bus route oracle. In *Proceedings of the sixth conference on Applied natural language processing*, pages 1–6. Association for Computational Linguistics.

Amble, T. (2009). BusTUC - a savant level intelligent bus oracle. *Norwegian Artificial Intelligens Symposium (NAIS)*.

Basi, R. K. (1999). WWW response rates to socio-demographic items. *Journal of the Market Research Society*, 41(4):397–401.

Bass, L, P. Clements and R. Kazman (2012). *Software Architecture in Practice*. SEI Series in Software Engineering. "Addison Wesley", 3rd ed edition.

Brace, I. (2013). *Questionnaire design: how to plan, structure and write survey material for effective market research*. Market research in practice series. London : Kogan Page, 3rd ed edition.

Brannen, J. and Coram, T. (1995). *Mixing methods: Qualitative and quantitative research*. Aldershot, Hants : Avebury.

Bratko, I. (2001). *Prolog programming for artificial intelligence*. Pearson education.

Brewer, J. and Hunter, A. (1989). *Multimethod research: A synthesis of styles*. Sage Publications, Inc.

Bryant, J. and Jones, M. (2012). Responsive web design. In *Pro HTML5 Performance*, pages 37–49. Apress.

Charland, A. and Leroux, B. (2011). Mobile application development: web vs. native. *Communications of the ACM*, 54(5):49–53.

Clark, R., Studholme, O., Murphy, C., and Manian, D. (2012). A layout for every occasion. In *Beginning HTML5 and CSS3*, pages 311–395. Apress.

Engell, T. B. (2012). TaleTUC: Text-to-Speech and Other Enhancements to Existing Bus Route Information Systems. Master's thesis, Norwegian University of Science and Technology.

Heitkötter, H., Hanschke, S., and Majchrzak, T. (2013). Evaluating cross-platform development approaches for mobile applications. In Cordeiro, J. and Krempels, K.-H., editors, *Web Information Systems and Technologies*, volume 140 of *Lecture Notes in Business Information Processing*, pages 120–138. Springer Berlin Heidelberg.

Horvat, J. (2014). Questionnaire. In Lovric, M., editor, *International Encyclopedia of Statistical Science*, pages 1154–1156. Springer Berlin Heidelberg.

IDC (2014). Press release: Worldwide smartphone shipments top one billion units for the first time. [Online; posted 27-January-2014; `http://www.idc.com/getdoc.jsp?containerId=prUS24645514`].

Kellner, P. (2004). Can online polls produce accurate findings? *International Journal of Market Research*, pages 3–21.

Kessin, Z. (2011). *Programming HTML5 applications: building powerful cross-platform environments in JavaScript.* " O'Reilly Media, Inc.".

Lloyd, J. (1987). *Foundations of Logic Programming.* Berlin: Springer-Verlag.

Lovdata (2013). Forskrift om universell utforming av IKT-løsninger. *Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger* [Available online; `https://lovdata.no/dokument/SF/forskrift/2013-06-21-732`].

Marcotte, E. (2011). *Responsive web design.* Jeffrey Zeldman.

Marcussen, C. J. and Anderstuen, R. (2012). *TaleTUC: Automatic Speech Recognition for a Bus Route Information System.* MSc Thesis, Department of Computer and Information Science, NTNU.

Marcussen, C. J. and Moland Eliassen, L. (2011). *TABuss: An Intelligent Smart-Phone Application.* Specialization Project, Department of Computer and Information Science, NTNU.

Marcussen, C. J., Moland Eliassen, L., Sætre, R., and Gambäck, B. (2012). Context-Awareness and Real-Time Information in an Intelligent Smartphone Application. In *Proceedings of the 12th International Conference on Innovative Internet Community Systems*, number P-204 in Lecture Notes in Informatics, pages 48–58. Bonn, Germany, June. Gesellschaft für Informatik, Köllen Verlag.

Natda, K. V. (2013). Responsive web design. *Eduvantage*, 1(1).

Paulson, L. D. (2005). Building rich web applications with AJAX. *Computer*, 38(10):14–17.

Pepitone, J. (2013). Worst PC sales drop in history. [Online; posted 10-April-2013; `http://money.cnn.com/2013/04/10/technology/pc-sales/?iid=EL`].

Pereira, F. C. and Shieber, S. M. (2002). *Prolog and natural-language analysis*. Microtome Publishing.

Ringdal, K. (2007). *Enhet og Mangfold: samfunnsvitenskapelig forskning og kvantitativ metode*. Fagbokforlaget Vigmostad & Bjørke AS.

Sowell, E. (2013). Native APIs, HTML5, and CSS3 on mobile today. In *Mobile ASP.NET MVC 5*, pages 157–179. Apress.

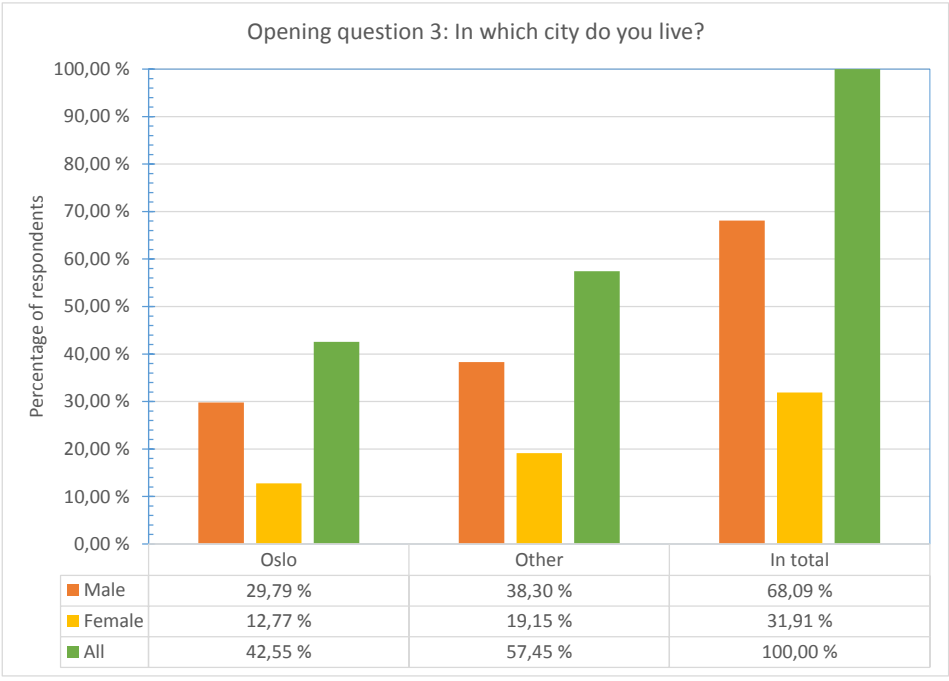
West, A. (2012). Moving to HTML5. In *Practical HTML5 Projects*, pages 1–37. Apress.

Wollamo, M. Q. (2013). Improving an existing natural language bus route information system. Master's thesis, Norwegian University of Science and Technology.

Appendices

A.1 Graphs

Opening question 3: In which city do you live?



	Oslo	Other	In total
Male	29,79 %	38,30 %	68,09 %
Female	12,77 %	19,15 %	31,91 %
All	42,55 %	57,45 %	100,00 %

Figure 1: Results of Opening question 3: In which city do you live?

Opening question 4: Are you familiar with the city of Oslo and its public transportation?

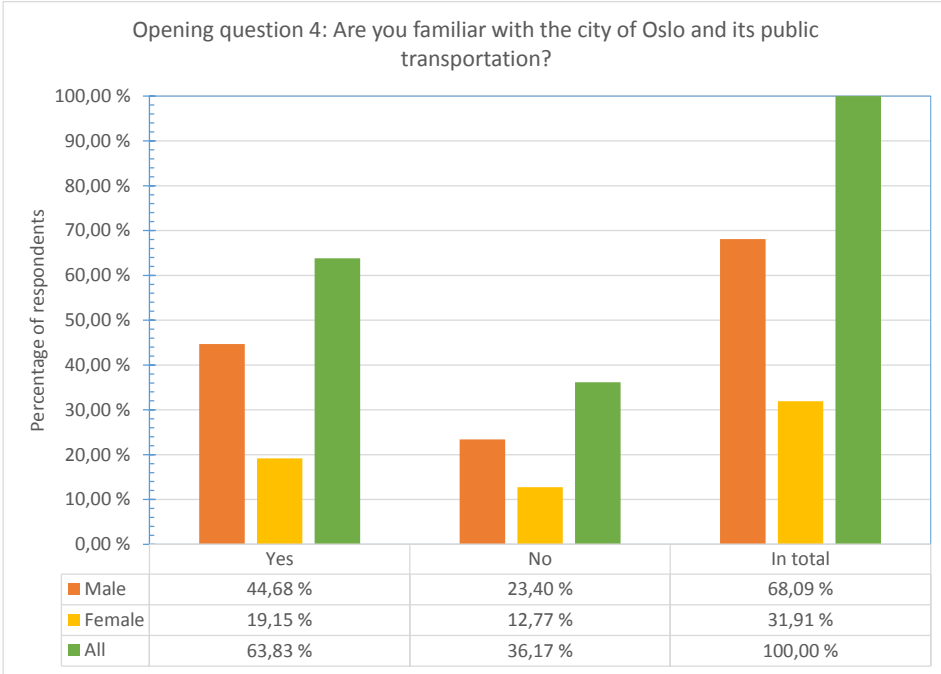
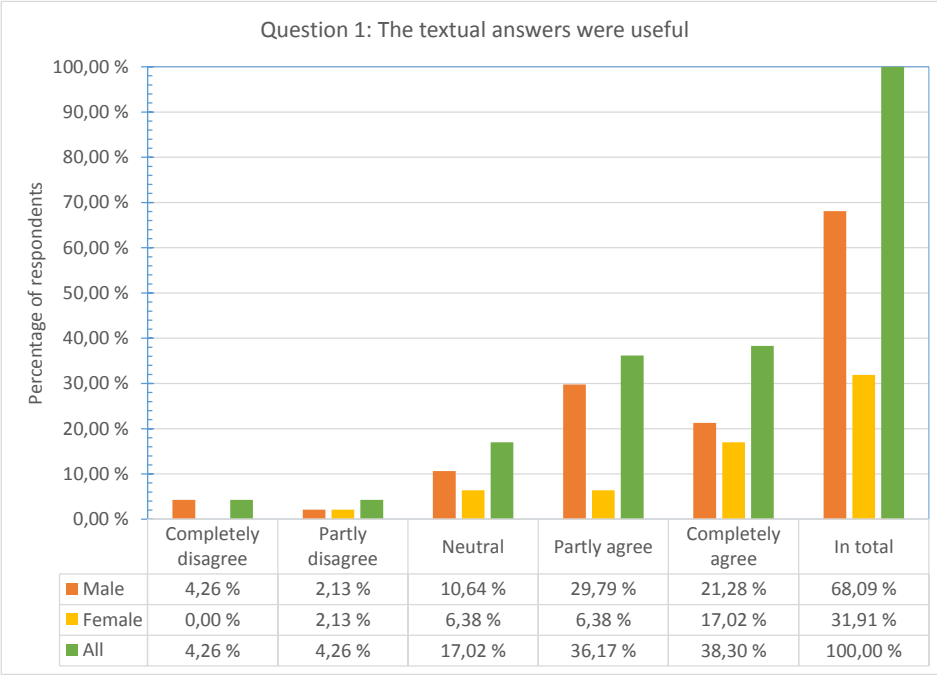


Figure 2: Results of Opening question 4: Are you familiar with the city of Oslo and its public transportation?

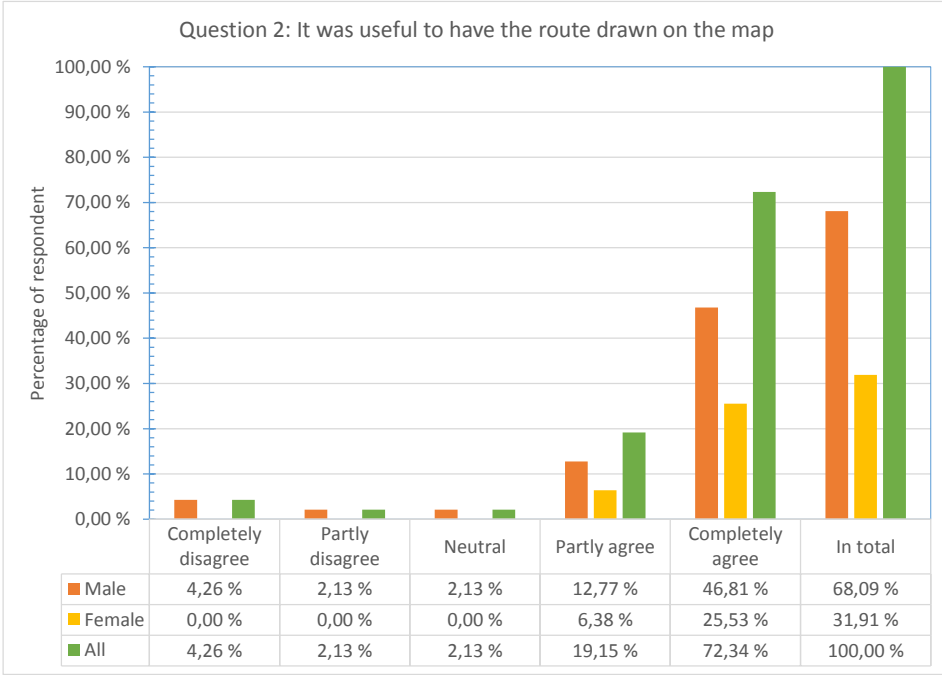
Question 1: The textual answers were useful



	Completely disagree	Partly disagree	Neutral	Partly agree	Completely agree	In total
Male	4,26 %	2,13 %	10,64 %	29,79 %	21,28 %	68,09 %
Female	0,00 %	2,13 %	6,38 %	6,38 %	17,02 %	31,91 %
All	4,26 %	4,26 %	17,02 %	36,17 %	38,30 %	100,00 %

Figure 3: Results of Question 1: The textual answers were useful

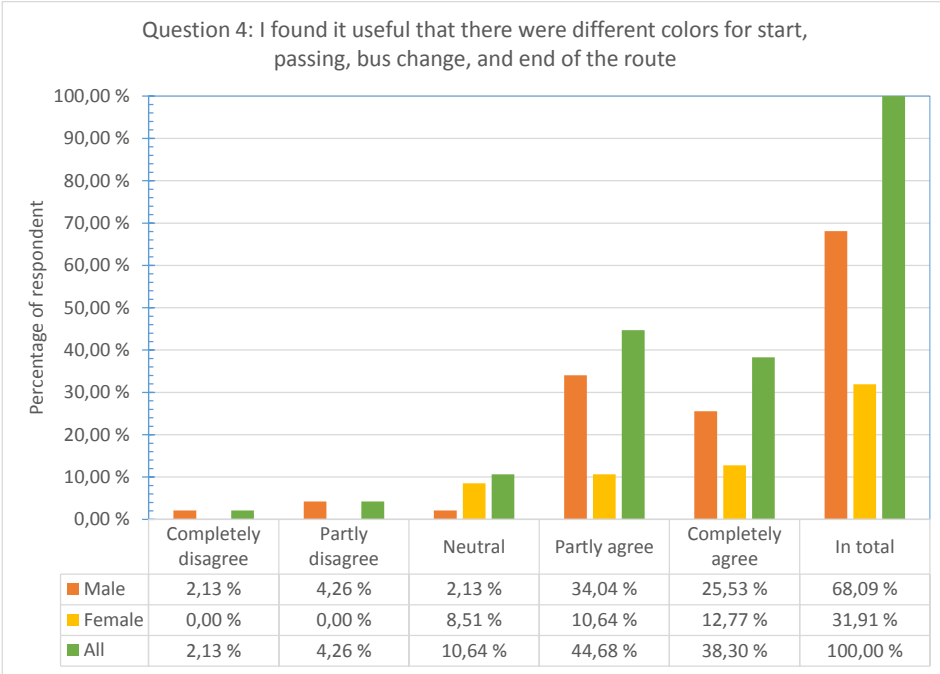
Question 2: It was useful to have the route drawn on a map



	Completely disagree	Partly disagree	Neutral	Partly agree	Completely agree	In total
Male	4,26 %	2,13 %	2,13 %	12,77 %	46,81 %	68,09 %
Female	0,00 %	0,00 %	0,00 %	6,38 %	25,53 %	31,91 %
All	4,26 %	2,13 %	2,13 %	19,15 %	72,34 %	100,00 %

Figure 4: Results of Question 2: It was useful to have the route drawn on a map

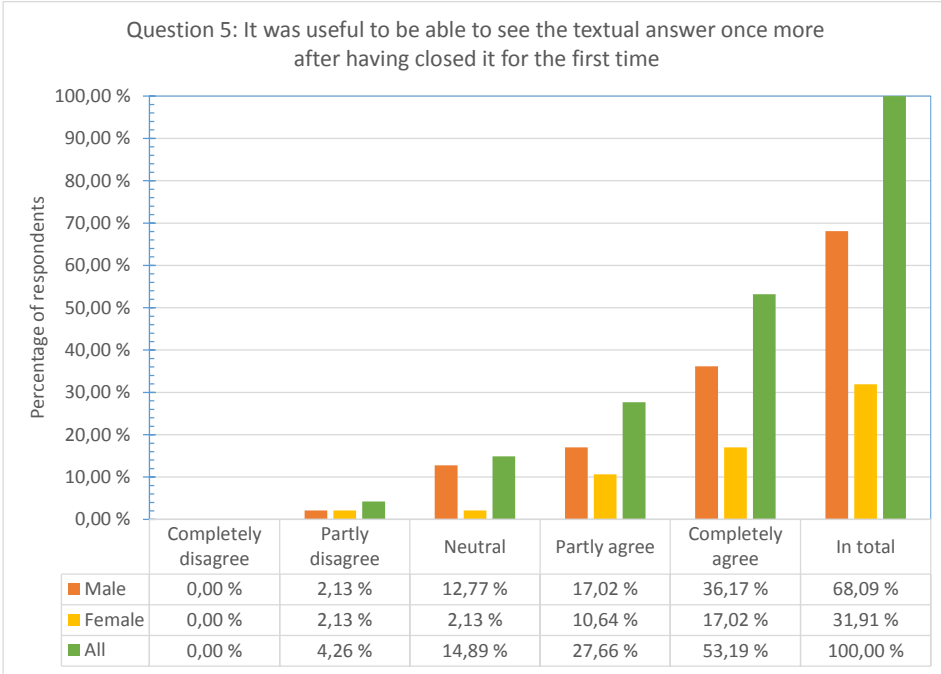
Question 4: I found it useful that there were different colors for start, passing, bus change, and end of the route



	Completely disagree	Partly disagree	Neutral	Partly agree	Completely agree	In total
Male	2,13 %	4,26 %	2,13 %	34,04 %	25,53 %	68,09 %
Female	0,00 %	0,00 %	8,51 %	10,64 %	12,77 %	31,91 %
All	2,13 %	4,26 %	10,64 %	44,68 %	38,30 %	100,00 %

Figure 5: Results of Question 4: I found it useful that there were different colors for start, passing, bus change, and end of the route

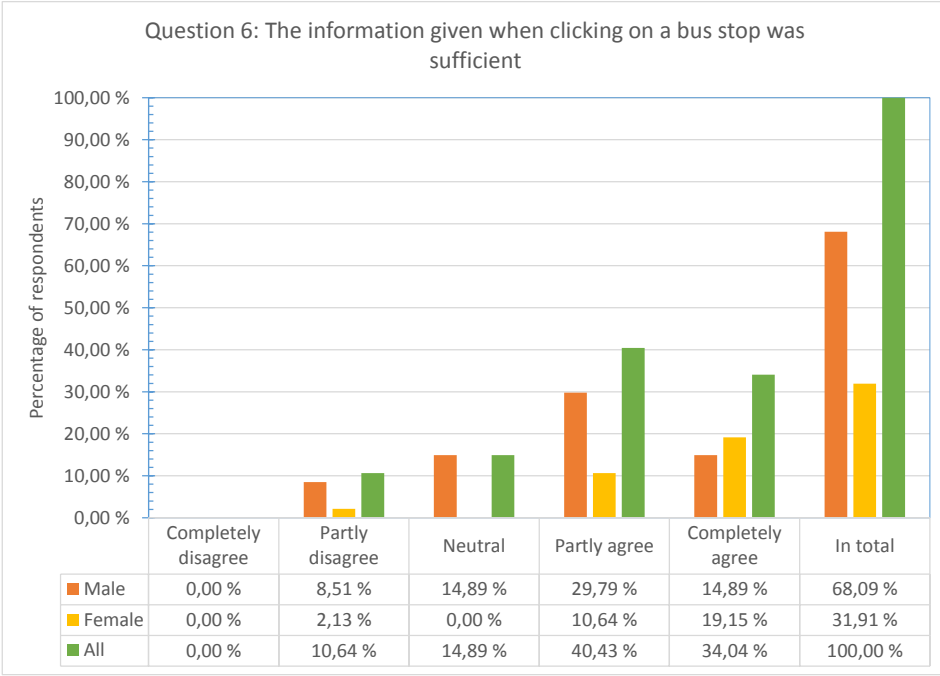
Question 5: It was useful to be able to see the textual answer once more after having closed it for the first time



	Completely disagree	Partly disagree	Neutral	Partly agree	Completely agree	In total
Male	0,00 %	2,13 %	12,77 %	17,02 %	36,17 %	68,09 %
Female	0,00 %	2,13 %	2,13 %	10,64 %	17,02 %	31,91 %
All	0,00 %	4,26 %	14,89 %	27,66 %	53,19 %	100,00 %

Figure 6: Results of Question 5: It was useful to be able to see the textual answer once more after having closed it for the first time.

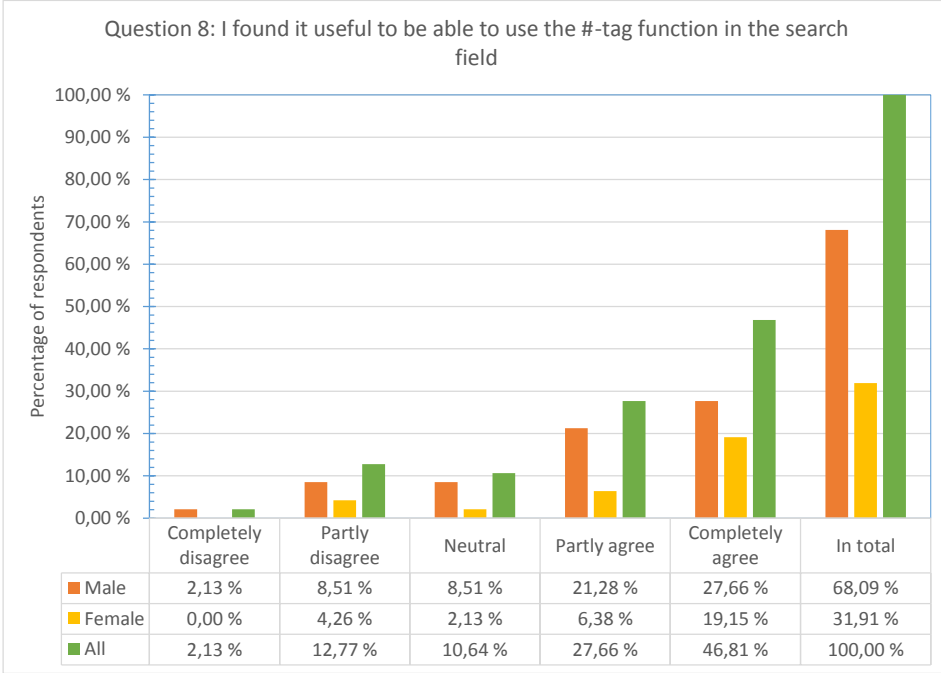
Question 6: The information given when clicking on a bus stop was sufficient



	Completely disagree	Partly disagree	Neutral	Partly agree	Completely agree	In total
Male	0,00 %	8,51 %	14,89 %	29,79 %	14,89 %	68,09 %
Female	0,00 %	2,13 %	0,00 %	10,64 %	19,15 %	31,91 %
All	0,00 %	10,64 %	14,89 %	40,43 %	34,04 %	100,00 %

Figure 7: Results of Question 6: The information given when clicking on a bus stop was sufficient

Question 8: I found it useful to be able to use the #-tag function in the search field



	Completely disagree	Partly disagree	Neutral	Partly agree	Completely agree	In total
Male	2,13 %	8,51 %	8,51 %	21,28 %	27,66 %	68,09 %
Female	0,00 %	4,26 %	2,13 %	6,38 %	19,15 %	31,91 %
All	2,13 %	12,77 %	10,64 %	27,66 %	46,81 %	100,00 %

Figure 8: Results of Question 8: I found it useful to be able to use the #-tag function in the search field

Question 10: How do you normally find travel schedules when you are going to travel by public transportation?

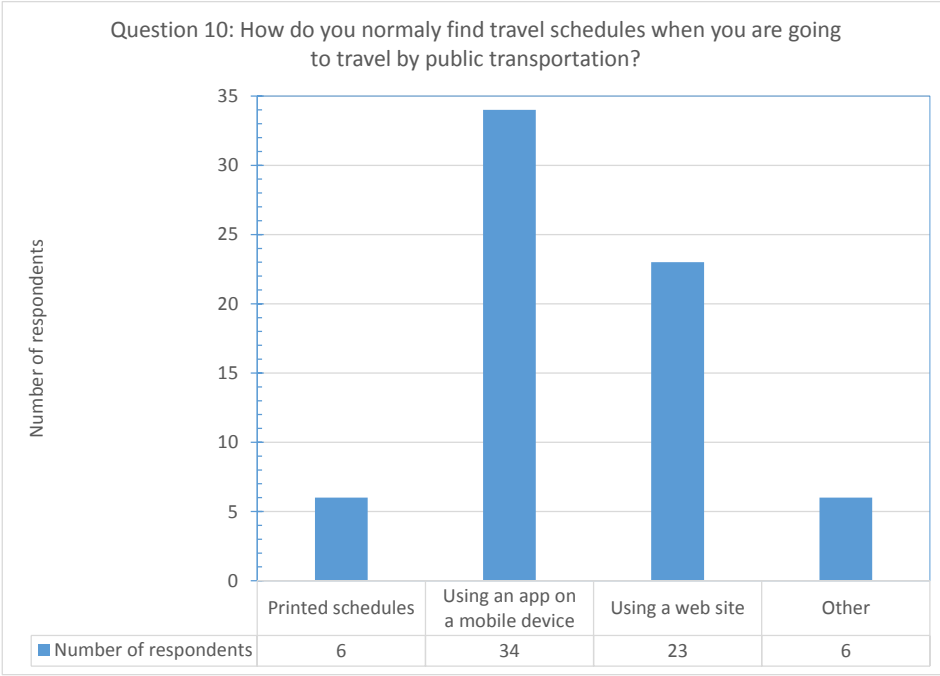
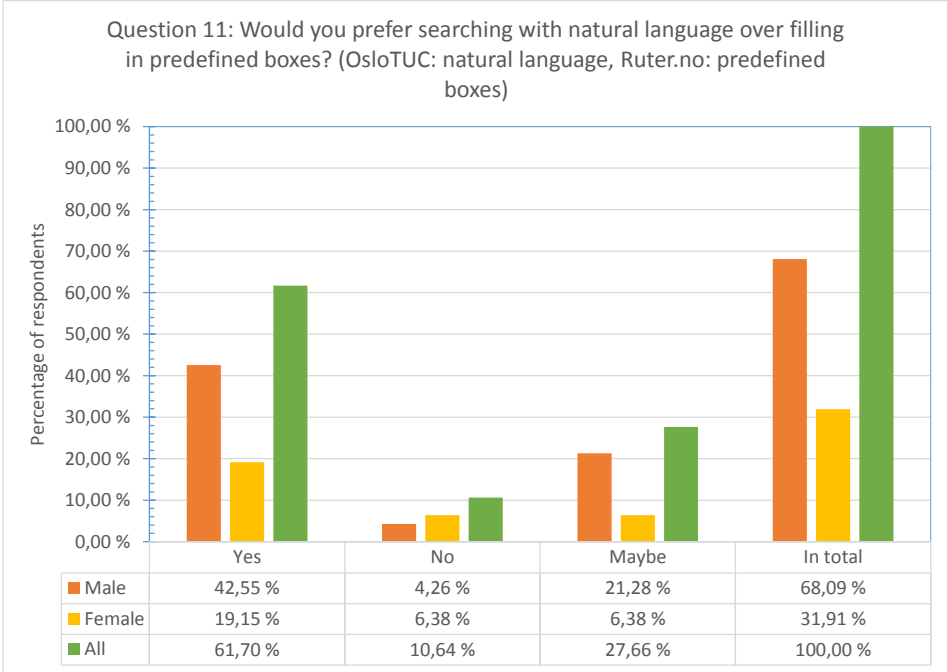


Figure 9: Results of Question 10: How do you normally find travel schedules when you are going to travel by public transportation?

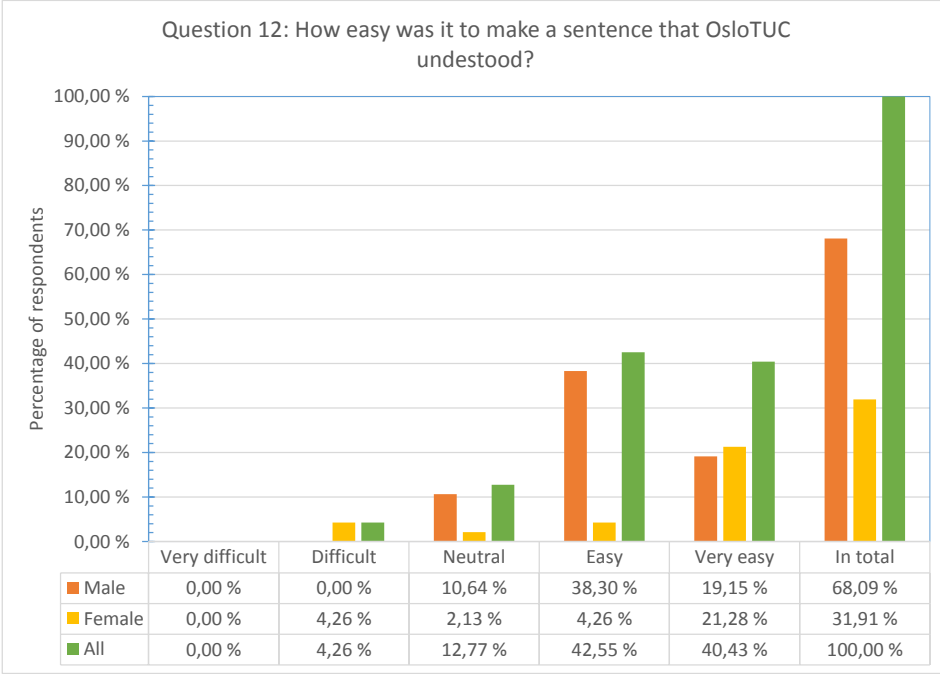
Question 11: Would you prefer searching with natural language over filling in predefined boxes? (OsloTUC: natural language, Ruter.no: predefined boxes)



	Yes	No	Maybe	In total
Male	42,55 %	4,26 %	21,28 %	68,09 %
Female	19,15 %	6,38 %	6,38 %	31,91 %
All	61,70 %	10,64 %	27,66 %	100,00 %

Figure 10: Results of Question 11: Would you prefer searching with natural language over filling in predefined boxes? (OsloTUC: natural language, Ruter.no: predefined boxes)

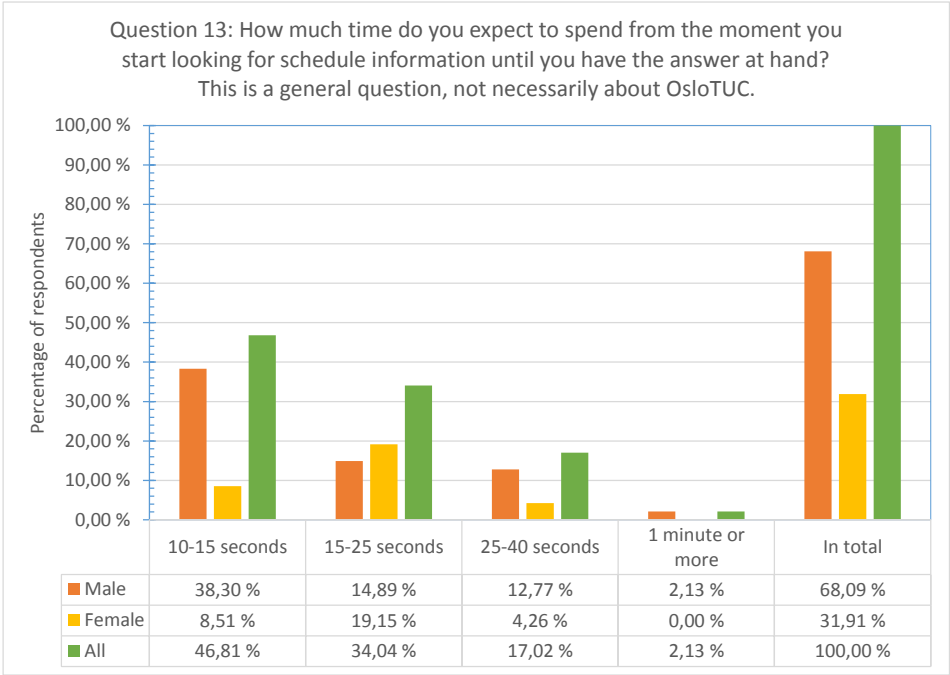
Question 12: How easy was it to make a sentence that OsloTUC understood?



	Very difficult	Difficult	Neutral	Easy	Very easy	In total
Male	0,00 %	0,00 %	10,64 %	38,30 %	19,15 %	68,09 %
Female	0,00 %	4,26 %	2,13 %	4,26 %	21,28 %	31,91 %
All	0,00 %	4,26 %	12,77 %	42,55 %	40,43 %	100,00 %

Figure 11: Results of Question 12: How easy was it to make a sentence that OsloTUC understood?

Question 13: How much time do you expect to spend from the moment you start looking for schedule information until you have the answer at hand? This is a general question, not necessarily about OsloTUC.



	10-15 seconds	15-25 seconds	25-40 seconds	1 minute or more	In total
Male	38,30 %	14,89 %	12,77 %	2,13 %	68,09 %
Female	8,51 %	19,15 %	4,26 %	0,00 %	31,91 %
All	46,81 %	34,04 %	17,02 %	2,13 %	100,00 %

Figure 12: Results of Question 13: How much time do you expect to spend from the moment you start looking for schedule information until you have the answer at hand? This is a general question, not necessarily about OsloTUC.

Question 14: I would have used OsloTUC to find travel information in the city of Oslo

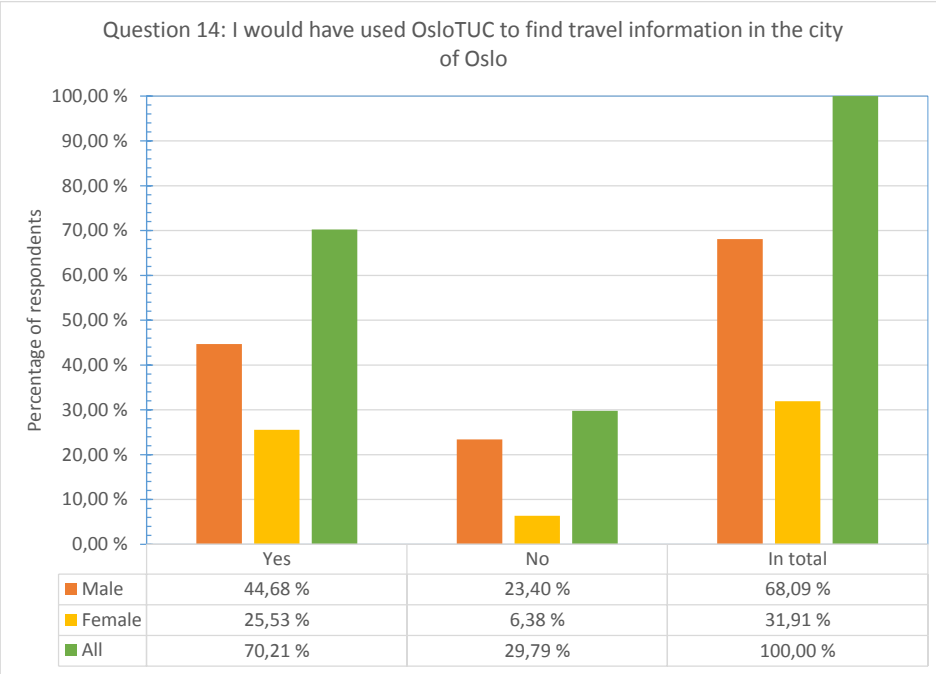


Figure 13: Results of Question 14: I would have used OsloTUC to find travel information in the city of Oslo

Question 15: If 'no' to the question above, would you have used OsloTUC to find travel information if it had support for trams, trains and subways?

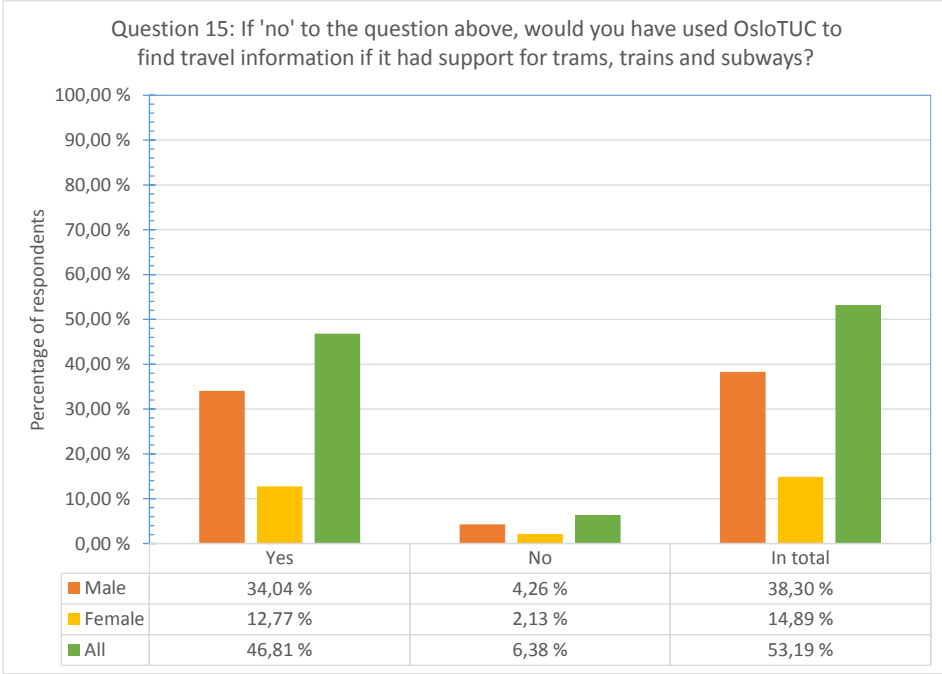


Figure 14: Results of Question 15: If 'no' to the question above, would you have used OsloTUC to find travel information if it had support for trams, trains and subways?

A.2 Questionnaire

Edit this form

Spørreundersøkelse for OsloTUC

OsloTUC er et fritekst reiseinformasjonssystem som fungerer for enkelte bussruter i Oslo. Du kan stille programmet et spørsmål om ruteinformasjon i form av en helt vanlig norsk eller engelsk setning. I denne brukerundersøkelsen vil du bli spurt om å utføre noen få oppgaver på OsloTUC, samt det allerede eksisterende systemet for Ruter. Undersøkelsen fokuserer på hvordan en bruker foretrekker å spørre om ruteopplysninger, og hvordan en bruker foretrekker å få resultatet vist.

OsloTUC har kun støtte for busser fra Unibus i Oslo, som dekker store deler av Oslo-området. Støtte for trikk, t-bane og tog er for øyeblikket ikke mulig. Dette vil resultere i noe ineffektive ruter sammenlignet med Ruter sine tjenester, da det ikke er mulig å benytte seg av overganger til noe annet enn busser. For holdeplasser som ikke ligger i rutene til Unibuss vil det derfor ikke dukke opp resultater i det hele tatt. Vær vennlig å ha dette i bakhodet når du tester OsloTUC.

Oppgavene skal gjennomføres både på OsloTUC og Ruter.no for å få et inntrykk av forskjellen på de to, før svarene i spørreundersøkelsen besvares. Svarene dreier seg i hovedsak om opplevelsen med OsloTUC.

Undersøkelsen vil ta mellom 10-15 minutter.

Oppgave 1: Åpne OsloTUC sin nettsideog trykk på "About and Help"-knappen nederst på siden, og les og forstå innholdet. (Bare på nettsiden til OsloTUC.)

Oppgave 2: Finn en forbindelse mellom 'Majorstuen' og 'Frogner plass' med OsloTUC og Ruter.no.

Oppgave 3: Klikk knappen "See last" for å se teksten til siste resultat med OsloTUC. Klikk på en holdeplass for å se informasjon om holdeplassen.

Oppgave 4: Finn en forbindelse mellom 'Majorstuen' og adressen 'Tåsen alle 12C' med OsloTUC og Ruter.no

Oppgave 5: Finn en forbindelse mellom 'Kampheimveien' og 'Holmlia Stasjon' før klokken 20:00 neste dag med OsloTUC og Ruter.no

Oppgave 6: Bruk #-tag-funksjonen beskrevet i "About and Help"-seksjonen for å finne holdeplassene 'Bislett' og 'Majorstuen' og gjør et søk mellom disse to holdeplassene.

Nettsider:

OsloTUC: http://vm-6114.idi.ntnu.no:9001
Ruter.no: http://ruter.no

Her kommer en rekke påstander om OsloTUC som du svarer på etter å ha gjort oppgavene:

*Required

Kjønn *

- ○ Kvinne
- ○ Mann
- ○ Other:

Alder *

- ○ Under 16
- ○ 16-20
- ○ 21-25
- ○ 26-30
- ○ 31-40
- ○ 41-50
- ○ Over 51

I hvilken by bor du? *

- ○ Oslo
- ○ Other:

Er du godt kjent med byen Oslo og dens kollektivtrafikk? *

- ○ Ja
- ○ Nei

Det var nyttig med et tekstuelt svar *

- ○ Helt uenig
- ○ Delvis uenig
- ○ Nøytral
- ○ Delvis enig
- ○ Helt enig

Det var nyttig å få ruten tegnet opp på kartet: *

- ○ Helt uenig
- ○ Delvis uenig
- ○ Nøytral
- ○ Delvis enig
- ○ Helt enig

Kommentar til ovenstående spørsmål

Jeg synes det var nyttig med forskjellige farger for start, passering, bytte av buss og avsluttet reise *

- ○ Helt uenig
- ○ Delvis uenig

- ○ Nøytral
- ○ Delvis enig
- ○ Helt enig

Det var nyttig å kunne se det tekstuelle svaret på nytt etter å ha lukket det første gang *

- ○ Helt uenig
- ○ Delvis uenig
- ○ Nøytral
- ○ Delvis enig
- ○ Helg enig

Informasjonen som kom opp når man klikket på en holdeplass var tilstrekkelig *

- ○ Helt uenig
- ○ Delvis uenig
- ○ Nøytral
- ○ Delvis enig
- ○ Helt enig

Kommentar til ovenstående spørsmål

Jeg synes det var nyttig å kunne søke etter bussholdeplasser med #-tag-funksjonen i søkefeltet *

- ○ Helt uenig
- ○ Delvis uenig
- ○ Nøytral
- ○ Delvis enig
- ○ Helt enig

Kunne svarene fra OsloTUC vært presentert på en annen måte?

Hvordan finner du vanligvis reiseopplysninger når du skal ta kollektivtransport *

- ☐ Rutetabeller
- ☐ Bruker en app på telefonen
- ☐ Bruker en nettside
- ☐ Other:

Ville du foretrukket fritekstsøk framfor å fylle inn i definerte bokser? (OsloTUC: fritekstsøk, Ruter.no: fylle inn i bokser) *

- ○ Nei
- ○ Kanskje
- ○ Ja

Hvor enkelt var det å formulere en setning som OsloTUC forsto? *

- ○ Svært vanskelig
- ○ Vanskelig
- ○ Nøytral
- ○ Enkelt
- ○ Svært enkelt

Hvor lang tid forventer du å bruke fra du begynner å lete etter ruteinformasjon til du har svaret du trenger? Dette er et generelt spørsmål, ikke nødvendigvis om OsloTUC. *

- ○ 10-15 sekunder
- ○ 15-25 sekunder
- ○ 25-40 sekunder
- ○ 1 minutt eller mer

Jeg ville ha brukt OsloTUC for å finne ruteopplysninger i Oslo *

- ○ Ja
- ○ Nei

Hvis nei på ovenstående, ville du brukt OsloTUC for å finne ruteopplysninger dersom den hadde støtte for trikk, tog og t-bane?

- ○ Ja
- ○ Nei

Hvis nei på ovenstående, hva kunne vært gjort anderledes som ville fått deg til å bruke den?

Submit

Never submit passwords through Google Forms.