Approaches to natural language processing in app-development

CAMRAN DJOWEINI
HENRIETTA HELLBERG
Acknowledgements

Firstly, we would like to thank our advisors at KTH, the Royal Institute of Technology, Anders Sjögren and Fadil Galjic for their patience, continuous support and much appreciated guidance throughout the course of this project.

Secondly, we would like to thank William Dalheim for his incredible enthusiasm and for the introduction to an exciting, and for us new, field.
Abstract

Natural language processing is an on-going field that is not yet fully established. A high demand for natural language processing in applications creates a need for good development-tools and different implementation approaches developed to suit the engineers behind the applications. This project approaches the field from an engineering point of view to research approaches, tools, and techniques that are readily available today for development of natural language processing support.

The sub-area of information retrieval of natural language processing was examined through a case study, where prototypes were developed to get a deeper understanding of the tools and techniques used for such tasks from an engineering point of view.

We found that there are two major approaches to developing natural language processing support for applications, high-level and low-level approaches. A categorization of tools and frameworks belonging to the two approaches as well as the source code, documentation and, evaluations, of two prototypes developed as part of the research are presented.

The choice of approach, tools and techniques should be based on the specifications and requirements of the final product and both levels have their own pros and cons. The results of the report are, to a large extent, generalizable as many different natural language processing tasks can be solved using similar solutions even if their goals vary.

Keywords
Natural language processing, information retrieval, voice-control, implementation approaches, NLP.
Sammanfattning

Datalingvistik (engelska natural language processing) är ett område inom datavetenskap som ännu inte är fullt etablerat. En hög efterfrågan av stöd för naturligt språk i applikationer skapar ett behov av tillvägagångssätt och verktyg anpassade för ingenjörer.

Detta projekt närmar sig området från en ingenjörs synvinkel för att undersöka de tillvägagångssätt, verktyg och tekniker som finns tillgängliga att arbeta med för utveckling av stöd för naturligt språk i applikationer i dagsläget.

Delområdet 'information retrieval' undersöktes genom en fallstudie, där prototyper utvecklades för att skapa en djupare förståelse av verktygen och teknikerna som används inom området.

Vi kom fram till att det går att kategorisera verktyg och tekniker i två olika grupper, beroende på hur distanserad utvecklaren är från den underliggande bearbetningen av språket. Kategorisering av verktyg och tekniker samt källkod, dokumentering och utvärdering av prototyperna presenteras som resultat.

Valen av tillvägagångssätt, tekniker och verktyg bör baseras på krav och specifikationer för den färdiga produkten. Resultaten av studien är till stor del generaliserbara eftersom lösningar till många problem inom området är likartade även om de slutgiltiga målen skiljer sig åt.

Nyckelord

Natural language processing, informationsinhämtning, röststyrning, implementerings tillvägagångssätt, NLP.
Table of contents

1 Introduction ........................................... 12
   1.1 An overview........................................... 12
   1.2 Problem.................................................. 14
   1.3 Aim........................................................ 15
   1.4 Limitations.............................................. 15

2 Background theory ................................... 16
   2.1 Natural language processing............................... 16
   2.2 Estimating probability.................................... 17
   2.3 Artificial intelligence and natural language processing........... 18
   2.4 String pre-processing, to prepare data for machine learning........... 19
   2.5 Parsing.................................................... 19
   2.6 Sequence tagging, labeling useful information......................... 20
   2.7 Neural networks and natural language processing....................... 20
   2.8 Natural language processing algorithms............................... 22
   2.9 A model for development of natural language processing............... 26

3 Method ...................................................... 28
   3.1 Research methodology..................................... 28
   3.2 Sub-questions of the research question........................... 30
   3.3 Research method.......................................... 32
   3.4 Project method........................................... 36
   3.5 Documentation and modelling method............................ 38

4 Techniques and tools for natural language processing ............... 40
   4.1 Structualizing development.................................... 40
   4.2 The underlying principles of natural language processing............ 41
   4.3 Tools for developing natural language processing support................ 42
   4.4 Bridging the theory and the prototype development..................... 48

5 Development of natural language processing support ................ 50
   5.1 Choice of techniques and tools................................ 50
   5.2 The prototypes............................................ 52
   5.3 Prototype one............................................. 54
   5.4 Prototype two............................................. 60
   5.5 Alternative solutions...................................... 66
6 Discussion and conclusion

6.1 Implementing natural language processing support .......................... 68
6.2 Validity and reliability of the methodology ................................. 70
6.3 The result .................................................................................. 71
6.4 Sustainability, economic and ethical aspects .............................. 72
6.5 Concluding remarks .................................................................. 73

References .................................................................................... 76
Author concepts and abbreviations

NLP  Natural Language Processing  
NLU  Natural Language Understanding  
CL  Computational Linguistics  
ML  Machine learning  
PS  Probability and Statistics  
AI  Artificial Intelligence  
NER  Named Entity Recognition  
POS  Part Of Speech  
CRF  Conditional random fields  
HMM  Hidden Markov Model  
ANN  Artificial Neural Network  
RNN  Recursive Neural Network  
LSTM  Long Short Term Memory implementation of a neural network

High-level techniques - High-level tools are an easily scalable package deal that distance the programmer, to different extents, from the underlying principles of natural language processing.

Low-level tools- Tools categorised as ‘low-level tools’ in this report are toolkits and libraries used for natural language processing, where the developer is allowed more control of the implementation-specific details of the natural language processing techniques. They are, however, not termed low-low level tools based on the availability of working with adapting algorithms and mathematical formulas.
1 Introduction

This chapter introduces the reader to the field of natural language processing, through an introductory section, and describes the context of the project.

1.1 An overview

Interest in how computers understand and process language, natural language processing and machine learning, more generally, began in the late 1940’s, and around the time when pioneering computer scientist, Alan Turing, published an article on ‘Computing Machinery and Intelligence’, where he asked the question: “can machines think?”[1]. The question itself is deceptively simple, but it gets to the root of more complicated philosophical questions of the meaning of ‘thinking’ in the computer context and the social implications if computers were to become truly intelligent. Computer intelligence is more relevant today than it ever has been before, with the machine learning field having matured from a theoretical stage into something that already has started to improve the everyday life of people across the world, through intelligent software applications.

Natural language processing, or NLP, is a field that is made up of several of the sub-areas of computer science, including machine learning and artificial intelligence. Natural language processing concerns itself with understanding and modelling of how computers can understand human language. The difference between the ease with which a human and a computer can learn and understand a language represents an interesting area of computer science that lies at the intersection of the human-computer interface. Interesting questions include: why is it so easy for a human to understand language and so hard for a computer to understand the context of a language whose “rules” it already knows? To us, this is a very interesting question that digs into the details of our understanding of how computers and machine learning works.

The term “natural language” refers to human languages, and is the opposite of “artificial languages”, which have been created by humans for interacting with computers. “Natural language processing” lies at the intersection of natural and artificial language and is about how computers process, and understand, natural language input. As a summary, the field of natural language processing is concerned with the question: how can we make computers understand language in the same way we understand other people?
Since Alan Turing wrote the article in the 1950’s, the field of machine learning has shown significant progress, much of which can be applied to natural language processing. Yet, the field is still in its infancy, there is not one correct or best solution to a natural language problem [2]. In more recent years, natural language processing has experienced a big increase in interest as part of the mobile revolution [3]. Humans have become increasingly dependant and reliant on software and its applications as it has become an integral and integrated part of society and our everyday lives. For example, interest in language processing has increased because of its promise to increase the efficiency by which users can complete diverse tasks, since humans have a natural inclination to prefer the natural language on which their own understanding of the world is based. The recent surge in interest for devices and applications like Amazon’s Alexa home assistant, and Google Home, is testament to this, since they allow the domestic user to interact with software as if it was another person [4].

However, problems remain in how computers process language, and there is active interest in trying to optimise existing solutions, while also researching new ones. The key areas of research and development are focussed on the individual components of natural language processing, such as for example entity recognition, required by engineers to develop efficient and accurate natural language processing engines.

Natural language processing is particularly interesting for engineers, since the fields allows for a different approach to problems both old and new, by allowing the computer to handle the execution of commands requiring several components as well as input in the form of voice rather than only manual text input. Natural language processing extends the possibilities of functionality and accessibility of systems as well as adds a new dimension to problem solving.

Next-generation natural language processing that allows human natural language use to control software outputs has the potential to revolutionize an increasingly connected and mobile society, and give rise to a new host of applications and engineering problems.
The natural language processing field covers many different aspects of natural language handling and includes several subfields, however the focus of this report lies in the subfield of identifying and processing relevant content in a set of data [5][6].

In this project different techniques for implementing natural language processing support are researched to better understand different approaches of implementation.

1.2 Problem

A Great Thing AB is a software developing company that develops apps for mobile platforms. A key feature of mobile platforms and the connected economy is ease of use, where the user of the application can communicate with the app and its platform and the service provider in an as easy way as possible. Natural language is an intuitive way for this communication to be performed, allowing the software user an easy way to communicate with the software. With regard to A Great Thing AB, the company is looking to improve a feature that allows the app user to use natural language to make demands and provide instructions. The company is looking into integrating customized natural language processing support and to develop a customised natural language processing solution that will be usable for both the current and future projects.

Natural language processing is an on-going field that is not yet fully established[6] and there is room for further research. A high demand for natural language processing in applications creates a need for good development-tools and different implementation approaches developed to suit the engineers behind the applications. Next-generation NPL would require the software to reliably detect not just the content of language, but its context as well as to deal with problems of ambiguity. This is an on-going focus of current development and research in the field and the question of what techniques and tools can be used for development of state of the art natural language processing support is tightly connected.

For our project, we have made an attempt to understand the current state of the field from an engineering point of view and based on this make directions for future development of natural language processing support for applications. The purpose of this thesis is to answer the question:

“How can natural language-support be implemented in an application?”
1.3 Aim

The aim of the project is to explore how natural language processing can be implemented in order to extend the functionality of an application as well as to create an overview of how to approach development from an engineering point of view. The project also aims to determine the value of the different approaches with respect to the complexity of the solutions and based on the company's wish to develop their own natural language processing engine. The project will benefit engineers by providing an introduction to how to approach development of natural language processing support, both in a general sense and in the context of information retrieval.

1.4 Limitations

The project touches on many parts of natural language processing but only focuses on the natural language processing techniques required for voice control based on natural language understanding and information retrieval. The project is limited by not going into greater detail about the initial phase of our natural language processing task, voice recognition. Voice recognition covers how voice is converted from analogue sound to digital sound and it is not covered in detail despite of its relevance to limit the scope of the project. In addition, the project is limited by not going into greater detail of the underlying principles of natural language processing. Only the subsections most relevant for our specific task, referring to the prototype development phase, are covered in detail. Beyond this is the fact that no one really holds the answer to how natural language processing should be implemented is a limitation in itself. The advancements in natural language processing are incremental and there is no single correct solution to a problem.
2 Background theory

This chapter contains an introductory section about the foundations of natural language processing as well as sections describing the theory behind the areas of natural language processing relating to this project.

2.1 Natural language processing

Natural language processing is made up of a combination of subsections of computer science where computational linguistics, probability/statistics as well as artificial intelligence and machine learning are all of high importance [7][8]. Computational linguistics is the area of natural language processing concerned with the “comprehension” of human language and is important because it is essential for the processing of language, to find meaning, as well as for the production of natural language by machines. Computational linguistics includes the task of handling basic language data and analysing it [9]. The relations between the different areas connected to natural language processing can be seen in figure 1.

![Figure 1](image)

**Figure 1.** Overlapping areas related to natural language processing: Artificial Intelligence, Machine Learning, Computational Linguistics, Probability and Statistics and Deep Learning.
2.2 Estimating probability

Probability and statistics, from a natural language processing standpoint, help in estimating the meaning of language. A statistical approach to natural language processing has been the mainstream for research for the last decades. It builds on the concept of language models, LMs for short, that are produced from the processing of training data by machine-learning algorithms [3].

When using a spell checker or a language tool like autocorrect, the word suggested to the user is determined from the input based on the probability of the word being the word the user is trying to input. The probability being based on the similarity of the word from a dictionary and the likelihood of the word following the previous word, or words, in that context. It is, for example, more likely that a sentence says “I am going home” rather than “I going home am”, in the same way as it is more likely that a sentence is meant to say “I live over there” rather than “I lime over ear”. The previous example highlights the value of probability estimations in natural language processing and hints at a number of different applications (e.g. in machine translation or for discerning what was said in a noisy environment, where “hear” might have been registered as something more similar to “ear”).

One type of word prediction algorithms, used for building models to estimate meaning, is called “N-gram”. In the N-gram, the “N” stands for ‘a number of words’ and, for example, a three-word N-gram is a three-gram (a three-word-long sequence of words). N-gram-models are one of the simplest kind of models used to assign probability to words in sentences to make an educated guess of the identity of the last word of the N-gram (like in the case of the 3-gram what the 4th word is likeliest to be) [10]. In effect, probability theory makes these suggestions educated guesses [11]. Probability estimations are very useful in estimating likelihood of meaning. However, different languages have different sets of expressions and grammatical rules, which makes a one-size-fits-all approach to natural language processing unlikely [12].

Statistical parsing also relates to the subsection of natural language processing that deals with probability and statistics. A statistical parser is built on probabilistic models of syntactic knowledge, meaning it uses statistical knowledge about syntax to improve parsing accuracy. The value of statistical approaches to parsing is most apparent when trying to solve problems of ambiguity.
2.3 Artificial intelligence and natural language processing

Artificial Intelligence, AI, is intelligence displayed by machines. Artificial intelligence is to (biological/natural) intelligence what artificial language is to natural languages. Artificial intelligence can be divided into two main categories with high relevance to the natural language processing field. These two categories can be referred to as the top-down and bottom-up methods and there are essentially different arguments for them being the most suitable method for developing artificial intelligence applications. The first, top-down method, aims to create AI from scratch (i.e. outside of the context of biological intelligence), and the other, bottom-up, aims to do the opposite, essentially creating digital neural networks that have been inspired by biological ones [13].

Machine learning is a central part of artificial intelligence and has a strong connection to natural language processing, as many of the techniques used today are built on a foundation of machine learning. Machine learning creates systems that can grow with a task and benefit from large sets of input data to better estimate probability and improve NLU capabilities. It is essentially the area of computer science that looks into how computers can learn using algorithms to find patterns that are of interest [3].

The algorithms used for machine learning can be divided up into smaller categories, depending on how the learning process is laid out. The main categories that are usually referred to in this context, are supervised, semi-supervised, unsupervised and reinforced learning.

The majority of machine learning is done by supervised learning [14]. Supervised learning is the method of having algorithms learn by training data. The data is made up of training examples, where each example have input variables, as well as an output variables. A supervised learning algorithm should analyze data and map the function, from the input variables, to the output variables, so that the function can be used to predict the output variable for new input data that it is given. Supervised learning can be conducted, for example, using algorithms such as Naive Bayes, k-nearest neighbour as well as various neural network approaches [14].

Unsupervised learning is used to find patterns in datasets without any information about what kind of samples are supplied. In this kind of training the computer doesn’t know what it’s trying to learn, but the aim is to discover patterns in the dataset.
2.4 String pre-processing, to prepare data for machine learning

To prepare data for machine learning tasks, string pre-processing also referred to as normalization, can be done to increase the efficiency of the machine learning. For example, instead of annotating all separate conjugations of words with different weights, referring to the importance of the word in a context (e.g. how often it appears), stemming and lemmatization can be used to find the most basic version of a word. The goal of both stemming and lemmatization is to find the stem from which a word ‘originates’. For example, ‘is’ and ‘are’ come from ‘to be’. A word can appear as different versions of itself in natural language. One example of this is the word ‘organize’. In natural language one might use the meaning of organize by writing ‘organized’, ‘organizes’ or ‘organizing’, to help discern the meaning of a word in its many different forms stemming is used to find the word stem. In the example of stemming the stem is ‘organiz’ and using lemmatization the stem is ‘organize’. Lemmatization makes a morphological analysis to decide on what a word stem is, while stemming only makes a cut where the last letter is the same in all representations [15].

Another string pre-processing technique is tokenization. Tokenization refers to the process of splitting up sequences of words into smaller parts. This often means splitting a sentence into its “word components”. Generally a token is a sequence of characters that in some way belong together to create semantic meaning. Tokenization is done as part of natural language processing to find useful semantic units and often includes the removal of punctuation as it doesn’t carry any deeper real world information [16].

Ex: Input: Hello there neighbour!
After tokenization: “Hello” “there” “neighbour”

2.5 Parsing

Syntactic parsing is an approach for handling natural language input and interpreting its syntactic structure. Syntactic parsing of a sentence is what gives the individual words meaning in relation to each other and in that way attempts to discern the meaning of the entire sentence. One of the biggest challenges in syntactic parsing is how to solve ambiguity problems [17].

Many of the state of the art solutions to dependency parsing belong to a group of parsers referred to as Transition-based dependency parsers. The
Stanford parser [18] belongs to this group together with the state-of-the-art parsers of ClearNLP, spaCy and syntaxNet [19] [20].

2.6 Sequence tagging, labeling useful information

So far, we have been discussing the role of machine learning and statistical models in natural language processing. The process of “sequence tagging” or “sequence labeling” is what deals with pattern recognition in machine learning, and it is commonly used in natural language processing in the form of part-of-speech tagging or Named Entity Recognition.

Named Entity Recognition or NER is a data extraction task that aims to find keywords in text that are of high value to the current processing task. The entities could be people, locations, time, values or other key elements in sentences that are important, depending on the task they are being used for. In relation to this project one such entity could be a location, valuable information needed for further processing of voice input [21].

A large number of the available sequence tagging models used today are linear statistical models and include Hidden Markov Models, Maximum Entropy Markov Models and Conditional Random Fields [22]. Out of these, conditional random fields is of particular relevance to the work described in this report.

Conditional random fields, or CRF, is an unbiased (i.e it uses unsupervised learning) method used for classification problems by which the language model learns how to classify different types of inputs into input classes and to derive from these classes, the classification of the original input [23]. In the context of this project, we use CRF for named entity recognition through the Stanford Named Entity Recognizer to identify locations [23].

A more recent approach to sequence tagging that has been yielding promising results is a non-linear process that uses recurrent neural networks to solve common sequence tagging problems [24][25].

2.7 Neural networks and natural language processing

Artificial neural networks, or ANN for short, are networks of nodes that have been developed to work in a similar way to that of mammalian brains, where each node in a network of nodes is referred to as a “neuron” [26]. A specific kind of neural network that has proven itself valuable in natural language processing is deep neural networks.

In deep learning, algorithms are used to try to mimic the process of “thinking” by finding abstractions in the data the networks are deployed on. Deep learning is made up of layers of algorithms, where each layer is
generally quite simple and only uses one algorithm or function, and the data that is being processed passes through each layer through an input-output connection. The ‘outer-most’ layer is referred to as the input layer and the last layer is referred to as the output layer. The input and output layers are connected to each other through ‘hidden layers’, the layers that lay in between them. Deep learning techniques have become more powerful with time as they benefit from large quantities of data, now more readily available than a decade or two ago, and faster processing time in computers, corresponding to faster GPU/CPUs.

Deep learning has great value in the context of natural language processing. Deep learning started to outperform other machine learning techniques back in 2006 and despite the fact that deep learning has been focused towards computer vision, up until recently, the first breakthrough results of ‘deep learning’ on large datasets happened in speech recognition [27].

Recurrent neural networks are a type of artificial neural networks closely related to deep neural networks. They share the basic structure of many layers (deep), but recurrent neural networks implement memory in each layer and each layer accepts input and can produce output (in comparison to ‘regular deep networks’ that only have one input layer and one output layer)[28]. Recurrent neural networks have demonstrated state of the art or near state of the art results in several areas related to natural language processing. Recurrent networks are for example very valuable when working with sequence tagging in general [29], part-of-speech tagging [30] and dependency parsing [31][32].

Long Short-Term Memory, LSTM, are a specific type of recurrent neural networks that have shown promising results for natural language processing. LSTMs are very effective and accurate, but are also more complicated to train and configure [33][34]. LSTMs and bi-LSTMs, a bi-directional implementation of LSTMs, also perform very well over several different languages. LSTMs outperform Hidden Markov Model and CRF approaches and have their strength in their proportional advantage growth depending on the size of the training corpora [35]. Readers interested in LSTMs, RNNs and ANNs in general can get a good primer in the paper “A primer on Neural Network models for natural language processing” by Yoav Goldberg [36].
2.8 Natural language processing algorithms

In this section we bring up, or explain in greater detail, some algorithms that readers will find interesting and relevant to the natural language processing field that were not covered in the background or that were not covered in detail great enough.

Hidden Markov Model

Hidden Markov Models, or HMMs, are used in natural language processing to compute the probability distribution of different components and to decide what is the best sequence for those components. Sequence labeling is an important part of natural language processing and tasks related to this phase include, but are not limited to, speech recognition, part-of-speech-tagging and named entity recognition [38].

HMM is a statistical model used for predicting probabilities based on certain observables. The model consists of two probabilistic pieces, the transition model and the observation model. The transition model tells us the transition from one state to the next over time, while the observation model tells us in a given state how likely we are to see different observations. If we have a set of states, which we define as \( \{S_1, S_2, ..., S_n\} \), the probability of the next state is dependant on the previous state. This is defined with formula 1.

\[
P(s_{ik} | s_{i1}, s_{i2}, ..., s_{ik-1}) = P(s_{ik} | s_{ik-1})
\]

**Formula 1.** Probability of the next state.

Calculating the probability of a sequence of states, can be done with formula 2 [39].

\[
P(s_{i1}, s_{i2}, ..., s_{ik}) = P(s_{ik} | s_{i1}, s_{i2}, ..., s_{ik-1})P(s_{i1}, s_{i2}, ..., s_{ik-1})
= P(s_{ik} | s_{ik-1})P(s_{i1}, s_{i2}, ..., s_{ik-2}) = ...
= P(s_{ik} | s_{ik-1})P(s_{ik-1} | s_{ik-2})...P(s_{i2} | s_{i1})P(s_{i1})
\]

**Formula 2.** The probability of a sequence of states.
When it comes to Named Entity Recognition, one of the main differences between HMM and, for example, CRF is that HMM assumes features to be independent, while CRF does not.

**Viterbi algorithm**

The Viterbi algorithm is used for finding the most probable sequence of the hidden states. This sequence is called the Viterbi path, resulting in a sequence of observed events. This algorithm is commonly used for speech recognition, whereas the algorithm receives an acoustic signal, which it treats as the observed sequence of events. A string of text is assumed to be the cause of the signal, and the algorithm finds the most probably string of text considering the signal. The algorithm looks at the series of previous states and the current received state to figure out what the most likely value of the current state is. The observation that the algorithm does is that for any state, at a given time T, there is one most likely path to that state. This means that if several paths meet at a certain stage, at the given time T, instead of calculating all of the paths from this state to states at time T+1, the less likely paths can be ignored. When this is applied to each time step, the number of calculations required are greatly reduced from $N^T$ to $T*N^2$ [41].

**Conditional Random Fields**

Conditional random fields, or CRF, is a statistical modeling method. It is intended to deal with task-specific predictions, where there exists a set of input variables, and a set of target variables. As an example, for text processing, the words in the sentence are the input variables, while the target variables are labels of words such as person or location. To increase the accuracy of labels, CRF uses the labels of the previous targets (i.e feature dependencies are taken into account). Every Feature function is a function that takes in the following as input:

- A sentence S
- A position I of a word in the sentence
- The Label L[I] of the current word
- The Label L[I-1] of the previous word
In the end the output of the feature functions transform to a probability [42].

CRF is often used for Named Entity Recognition, which is also the case in this project.

**Naive Bayes**

The Naive Bayes algorithm is a machine learning algorithm that is commonly used for classification (e.g. for text classification, such as spam filtration, and classification of news articles).

For instance, if we want to classify a book review, and we have two classes, positive and negative. Our book review will contain certain negative words (e.g. hate, boring), and certain positive ones (e.g. love, hilarious, funny). The number of times positive and negative words appear in the review will affect if it is classified as a negative or positive review. The algorithm is called ‘Naive’ because it naively assumes features to be independent and ‘Bayes’, because it is based on Bayes’ theorem. Bayes theorem’ describes the probability of ‘event A’ occurring given if ‘event B’ has occurred. The formula for Bayes theorem’ can be seen in formula 3 [43].

\[
P(A | B) = \frac{P(B | A)P(A)}{P(B)}
\]

**Formula 3.** Bayes theorem, formula for calculating probability.

In formula 3, P(A|B) is the probability of ‘event A’ occurring given if ‘event B’ has occurred. P(A) and P(B) are the probabilities of the occurrence of ‘event A’ and ‘B’ respectively. P(B|A) is the probability of ‘event B’ occurring given if ‘event A’ has occurred.

Naive Bayes has many applications in natural language processing. It is used in Bayesian classifiers to assign classes to content, content being what is fed as input to the algorithm (e.g. sentences or whole documents). The class assignment is done to tag the input with what the algorithm finds is the most suitable class for the content.

We use Naive Bayes to estimate the probability of a sentence referring to a specific user intent. Expressing something using natural language can be done in a number of ways, while the core meaning of the sentence remains the same. For example, if a user expresses a wish to buy a train ticket, this wish could be expressed as, “A ticket to Vienna, please”, “Can I buy a ticket
for the 9:51 service to Stockholm” or “I would like to buy an off-peak return to Edinburgh”. The algorithm does not take into account how or if any of the words depend on each other and the presence of one word does not affect the “estimation-value” of another. The algorithm is used to create a classifier by using annotated input data to train it.

Input for the algorithm is a document and a fixed set of classes \( C = \{c_1,c_2,c_3,c_4\} \) as well as a training dataset of \( M \) entries. The classes used in this project are a set of intents. The output is a classifier trained for the specific input data supplied. Naive Bayes sees a document as a “bag of words”, where each word has a different value for each probability of it belonging to a specific class and the probability of each word belonging to a specific class is calculated depending on the input dataset. If the word “ticket” appears more commonly in sentences referring to when someone wants to go somewhere by train than, for example, when someone is trying to say goodbye and close an application, the probability of the sentence belonging to the train intent class increases as the word train will get a higher probability value for the train-class than the goodbye-class.

The probabilities of the full sentences belonging to each of the classes \( c_1, c_2, c_3 \) and \( c_4 \) are all calculated by multiplying the individual ‘word-belongs-to-class’ probability values together, then by multiplying them with the probability of the class itself and finally by dividing them by the evidence (the probability of encountering a specific pattern independent from the classes) to normalize the result. The result of the probability of the sentence belonging to the classes is then used to compare them to each other to see which class it most probably belongs to [44][45].

A figure, formula 4, describing the calculation, where the X represents what is in our case words) can be seen below.

\[
\text{Estimated probability} = \frac{(P(X|C) \cdot P(C))}{P(X)}
\]

**Formula 4.** The formula for calculating probability in a naive bayes classifier.
2.9 A model for development of natural language processing

AI, machine learning and computational linguistics are all crucial for natural language processing. There are many different applications and customization needs for natural language processing, and therefore it helps to further subdivide the task of implementation into smaller components. Ranjan et al [2], describes a way of dividing up natural language processing into three components as illustrated in figure 2. The first component, language modeling, is where the probable meaning of the input is statistically evaluated without any respect to the actual meaning. For example the likelihood of a sequence of words having been spoken in that order.

The second component, part of speech tagging is where the grammar of the input is identified and tagged [2]. Part-of-speech, or POS, refers to what syntactic relation a word has to a sentence. When using POS-tagging the words in a sentence are classified as belonging to a certain group. One of the main groups are nouns. Part-of-speech tagging is important as the group a word belongs to carries a lot of information about the meaning of the word and affects the meaning of a sentence. It is also very valuable in named entity recognition [37].

In the third component, parsing, the context of the input is evaluated. See figure 2.

![Figure 2](image.png)

Figure 2. Components of natural language processing. Ranjan et al.’s three components of natural language processing. The first component of their model is ‘language modeling’, where the probability of the input is statistically evaluated. The second is the ‘part of speech tagging’ where the individual words of the sentence are tagged with their grammar and the third component is where the dependencies of the words are evaluated through parsing.
3 Method

This chapter describes the methodology and method used in the project. The chapter includes a section about the research method, a section about the project method as well as sections describing the choice of techniques, sub-questions of the research question and a section about the documents of the project.

3.1 Research methodology

This section gives an overview of the theory behind the research method. A case study, a qualitative study and the technological scientific method described by Bunge are described.

3.1.1 Case study

This report examines the question: ‘How can natural language support be implemented in an application?’ To do so, we use a case study to research how to develop NLU for a specific case in the form of a natural language voice-command. Prototypes are developed to explore the natural language processing techniques and to incrementally get closer to exploring techniques that can be used for developing a customized engine for natural language processing. The problem is approached by dividing the wide context of the topic into smaller sub-questions.

3.1.2 Qualitative method

The work presented in this report has been performed qualitatively, revolving around the notion that the understanding of a problem is based on acquiring an overview of it. To this purpose, the report starts by exploring an area of computer science that is previously unknown to the authors and to then explore the field through a case study and the development of prototypes.
3.1.3 Bunge

The research method used in this project is based on a general outline for scientific research methods in the form of the technological scientific method described by Bunge [46]. The ten bullet points described by Bunge that were used as the foundation of the research method developed for this project are described below:

1. How can the problem be solved?

2. How can a technique or product be developed to solve the problem in an efficient manner?

3. What data is available for developing the technique or product?

4. Develop the technique or product based on the data. If the technique of the product is satisfactory, go to step 6.

5. Try a different technique or product.

6. Create a model or simulation of the suggested technique or product.

7. What are the consequences of the model or simulation?

8. Test the implementation of the model or simulation. If the result is not satisfactory go to step 9, otherwise go to step 10.

9. Identify and correct possible flaws in the model or simulation.

10. Evaluate the result together with previous knowledge and praxis and identify new problem areas for future research.
3.2 Sub-questions of the research question

The main research question of the project is described and linked to the more specific sub-questions based on the case study approach to the research question, according to the paragraphs below:

Main research question:
“How can natural language support be implemented in an application?”

Sub questions:

- Research if it is possible to implement voice support in an application given the scope of the project and the author's background.

  ‘Is it possible, based on the project members level of knowledge at the start of the project, to develop NLP support for an application?’

- Research the different approaches to developing voice support in an application given what tools are readily available today.

  ‘What approaches are used to implement NLP? How can they be categorized?’

  ‘What frameworks and tools are available for developing NLP applications today?’

- Research natural language processing in the context of information retrieval on a conceptual level.

  ‘How can meaning be extracted from text?’
Research how different techniques work to gain a better understanding and lay the foundation for approaching the problem from a more complex level, both during the project and for future projects.

- Research important concepts
- Research what algorithms are useful in the development of natural language understanding applications.

'What are the underlying principles of the tools and frameworks used?'
3.3 Research method

The research method is based on a general outline for scientific research methods in the form of the technological scientific method described by Bunge. Details about this method are covered in the methodology section earlier in this chapter. The research method in full is described in figure 3 below.

![Figure 3: Research method](image)

**Figure 3** Research method

1. **Understand the problem**

   This step corresponds to the first and second steps of Bunge’s scientific technological method, “How can the problem be solved? How can a technique or product be developed to solve the problem in an efficient manner?”, and is where knowledge is collected to create an overview of the field and gain a better understanding of the problem and what solving the problem entails. This is done by conducting a literature study. Our project method is based on the method described by Eklund (see project method section in this chapter). Ekelund's method highlights the importance of focusing on the **effect goals** rather than the **result goals** to answer the question “What is the problem to be solved?” [47]. The effect goal, in our context, is concerned with improving/extending the functionality of an application. The **result goal** is some form of documentation and evaluation of tools that aid in developing natural language processing support as well as prototypes that demonstrate the use of the tools and techniques.
As a summary, this phase of the project is dedicated to understanding how natural language processing, in the form of voice-control can be implemented in an application.

2. **Find development techniques**

   This step corresponds to the third step in Bunge's method, “*What data is available for developing the technique or product?*”, and aims to map different approaches to implementing natural language processing support. This is done by researching what techniques and tools are available for developing natural language processing support.

3. **Evaluate techniques and tools**

   Tools and techniques found through the literature study are evaluated to determine their value in the context of the project. The tool or technique is evaluated with respect to the company's specification as well as its relevance to information extraction tasks. Tools are evaluated and categorized to differentiate the complexity levels of utilizing them in the development of natural language processing support.

4. **Development of prototypes**

   When techniques have been categorized, prototypes are developed. This step corresponds to the phase of the research method illustrated in figure 4.

![Figure 4. The prototype phase](image-url)
The techniques and tools, identified in the second and categorized in the third step of the research method, are explored through the iterative and incremental development of prototypes.

The prototypes should be able to process a specific request made by the user starting with the request being, optionally, spoken out loud. The spoken language should be transcribed into text and passed on to NLU components for interpretation. The information retrieved should be used to trigger an appropriate action based on the request.

4.1 Development approach

The first set of tools and techniques used in the development of the prototypes should be based on the most easily approachable category of tools. A successful implementation based on this category serves as an initial proof of concept by demonstrating that implementation of natural language processing support is possible, see the section 'sub-questions' earlier in this chapter.

As the aim is to gain a deeper understanding of the different types of tools and techniques used for natural language processing development should ideally be attempted based on all identified categories that fall within the scope of the project. All following prototypes serve as proof of concept for their respective level. The tools and techniques explored through development form an integral part of the foundation for the discussion of the sub-questions.

5. Evaluate prototype:

This step reflects steps 7 and 8 in Bunge’s method. “What are the consequences of the model or simulation? Test the implementation of the model or simulation“. When a prototype passes the evaluation, a new set of development tools are selected from the same, or the following, category and another prototype is developed. If the prototype does not fulfil the requirements, flaws are identified and corrected.
5.1 *Evaluation method*

Meeting the requirements of the prototypes corresponds to the successful implementation of all functionality. The resulting prototype and the techniques used are evaluated by assessing how well they fulfill the specification. The tools and techniques are also evaluated based on their value in the context of developing a customized natural language processing engine, and depending on the evaluation of the prototypes, one prototype is possibly integrated into the company’s system.

**Specification**

*The prototype should be able to process a specific request made by the user starting with the request being spoken out loud by the user. The spoken language should be transcribed into text and passed on to the natural language understanding components. The final step is to complete the action found in the request.*
3.4 Project method

For projects of this magnitude, project methods are of great importance in managing resources. Project methods are methods and tools that can be used to help make sense of the workload ahead when embarking on a new project. The first phase of this project was the planning phase, where a Gantt chart was created. Our chart illustrates the schedule of the project, and helped create an overview of the work ahead. The chart covers set deadlines and goals for all steps required in the project and was an easy, yet efficient, method of keeping track of the progress made within the project.

3.4.1 The MoSCoW method

The MoSCoW method is a prioritization method used in projects to highlight the level of importance of different requirements with, in our case, regard to the financial, functional and time limitations of the project. The method is used to guide the resources of the project into achieving a ‘good’ or an ‘even better’ result. In our case this corresponds to how well we can answer the research question based on the result. The MoSCoW model can be linked to the iterations of the prototype phase in the research method. The resources of the project control the number of iterations that fall within the budget of the project.

The MoSCoW method is used in combination with the triangular method described by Eklund, represented in figure 5. The resources of the project are evaluated based on the three cornerstones of a successful project, time, function and cost.

![Figure 5. Our adaptation of Eklund’s triangle balanced around the three cornerstones of a successful project.](image)
The combination of these methods, having taken into regard the circumstances of the project, leaves us with the prioritization hierarchy described below and on the next page.

**Must have**

- **Achieve the course goals**
  
  → *Done by displaying the usage and understanding of the relevant scientific research methods as well the knowledge of relevant mathematical and technical methods needed to complete the project.*
  
  → Answering the research question.
    - Identify techniques and tools of interest to developers.
    - The creation of a first prototype, a proof of concept.

**Should have**

- A second iteration of the prototype phase, development of a prototype using another set of techniques and tools on either from a second category or the same, depending on the outcome of the previous prototype.

**Could have**

- Implement learning. Including training of models.

- Begin the process of implementing the prototype to the company system.

**Won’t have**

- Attempt more approaches.
3.5 Documentation and modelling method

The documentation is based on templates provided by KTH, but have been adapted to fit the general format of the project.

UML was used as the main modeling method and standard. It was used mainly for modeling the interaction diagrams of the prototypes, but also for developing general models of system overviews.

A GitHub repository was created to share the source code of the prototypes in hope that it might help developers wanting to implement natural language processing support, as well as to help future projects start of where this project ends with suggestions for future work. The github repository contains the project files of prototype one and two (github.com/NLPproject2017).
4 Techniques and tools for natural language processing

This chapter is part of the result and contains sections describing; natural language processing in the context of our project on a conceptual level; a categorization of tools, how to structuralize development as well as a section about the underlying principles of natural language processing.

4.1 Structuralizing development

We felt that it was helpful, to address how to structuralize the natural language processing of our prototypes, to divide natural language processing task into smaller components. Ranjan et al's model, described in the background chapter, gives a good overview of a natural language processing task. However, it is quite specific and does not include individual components for some components needed for the incremental development in this project. Although the development basically follows the same flow as Ranjan et al's model we decided to make; speech recognition, the creation of training datasets, string pre-processing, training of models and sequence tagging (in the context of this project) into separate components. Where, although speech recognition is technically related to both modelling and sequence tagging, it has its own component. This division is based on speech recognition not being a focus in this project and separate from the processing of the resulting string. Figure 6 describes Ranjan et al's model as well as our model.

![Figure 6. Components of natural language processing.](image)
A. Ranjan et al.’s three components of natural language processing. The first component of their model is ‘language modeling’, where the probability of the input is statistically evaluated. The second is the ‘part of speech tagging’ where the individual words of the sentence are tagged with their grammar and the third component is where the dependencies of the words are evaluated through parsing.

B. Our model adapted from Ranjan et al.’s. The first component is ‘speech recognition’ and is where speech is captured and transcribed into text. The second component is ‘creation of training datasets’ and is where datasets containing data specifically useful to our task are created. Ideally there needs to be both datasets for training as well as for testing. The third component is ‘string pre-processing’ and corresponds to the preparation of the data before the training of models. The fourth component is ‘training of models’, the fifth is ‘sequence tagging’.

In our model the components were also categorized as belonging to either of two component blocks. One for speech recognition and one for the remaining components.

4.2 The underlying principles of natural language processing

Through the literature research we learned about concepts and techniques relevant for developing natural language processing support and for information retrieval. Some of the most prominent general concepts identified were: Intent identification, intents, entity recognition, entities, classification and models. Techniques have connections to a number of different concepts important for the use of the technique, although too many to create a list. A few examples, linked to string preprocessing and described in the theory chapter, are: tokenization, lemmatization and stemming

4.2.1 Extracting meaning from text

‘How can meaning be extracted from text?’

There is no single ‘correct’ approach for how information should be retrieved
from text but there are a number of techniques that, when used either singularly or in combination, can help in identifying interesting pieces of information.

Techniques useful for extracting information from text, described in the theory and explored through the development of the prototypes are primarily; string preprocessing, training of models, classification, parsing, intent identification and sequence tagging.

A chain of techniques that could be utilized for retrieval of information from a string was identified through the literature study. The chain, described in the model in the previous section, works by combining features of different techniques to extract information and to then combine this information to interpret meaning. The chain is made up of: ‘creation of training datasets’, ‘string pre-processing’, ‘training of models’ and ‘sequence tagging’.

String preprocessing is especially valuable when creating datasets or when preparing datasets for training as it helps in handling large amounts of data and improves accuracy of models.

Models are trained using machine learning algorithms and are useful for finding patterns. Models can be used for identifying things like entities (e.g. locations) or intent (e.g. commands) in text using classifiers. There are different types of models and which ones should be used depend on the purpose of the natural language processing task. It is generally a good idea to use multiple models in combination.

The output from our chain of events is entities and intents. These ‘string components’ basically carry the meaning of the input and can be used to trigger an action based on a command.

### 4.3 Tools for developing natural language processing support

‘What frameworks and tools are available for developing NLP applications today? How can they be categorized?’

This section describes a selection of the tools that we came across during the literature research, and a categorization, that are useful for developing natural language processing support for applications.

To begin our research we wanted to understand what natural language processing tools are already available, to guide our own development. To this effect we performed an Internet search trying to identify a wide ranging collection of tools based on the previously identified techniques (e.g. string preprocessing, sequence tagging, training of models, and parsing) that could be used to extract meaning from natural language. The search was performed
using the Google search engine (ironically also a NLP application of a sort).

Many of the tools we encountered in the literature, we felt fit quite well into one of two categories of utility for natural language support development. Therefore, we decided to organise these categories based on the level of knowledge about natural language processing required for working with the tool itself. We thought that cloud-based solutions accessed via APIs, where the natural language processing input is handled as a service, were best placed into a category we termed ‘\textit{high-level tools’}, whereas frameworks and toolkits that do not require the use of cloud services were placed in a category termed ‘\textit{low-level tools’}. These categories are described in more detail in their corresponding sections.

The tools identified in tables 6 and 7, \textit{high-level} and \textit{low-level} tools, only represent a subset of the large collection of tools that are already available to developers. Notably, most of the tools that we identified do not have a full combination of features developers might desire in an natural language processing tool. Most of the low level tools support features focused around only one of the commonly used natural language processing techniques. For example, some of the tools have features focused around sequence tagging (\textit{e.g.} Stanford NER), some string pre-processing (\textit{e.g.} Stanford Core, NTLK) and some are focused around machine learning (\textit{e.g.} Weka).

Low-level tools could be categorized further based on the primary technique-focus of the tool. For example, it would be suiting to place Stanford Core into the string pre-processing category. Another observation is that there are several tools that work as overlays to other tools (\textit{e.g.} tflearn, SyntaxNet) and these tools could be placed into separate subcategories linked by their dependencies. This categorization is not displayed in the tables of the report based on the fact that the tables only contain a selection of the tools available. Some tools also cover several techniques and attempting to display several layers of categorization on paper might obscure the original purpose of creating an overview of tools.

\textit{4.3.1 High level tools: natural language processing as a service}

The high-level tools include tools for developing natural language processing support that, to different extents, do not reveal the underlying principles of natural language processing to the developer. These tools are easily approachable as a developer as they do not require any deeper understanding of natural language processing. These tools have their foundation in state of the art solutions and can be accessed through web interfaces and API calls.
Key concepts for working with the high-level techniques are: intent, agent, and entities. Table 6 describes a selection of high level tools.

Table 6. High level NLP tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Google Dialogflow</strong></td>
<td>Dialogflow (formerly known as API.ai) is a company that develops technologies for human-computer interaction based on natural language conversations. This company has a voice-enabling engine, allowing VUI (Voice-user interface) to applications on different operating systems, such as Android and IOS. This interface works by using machine learning algorithms to be able to match the request of the user to specific intents and uses entities to be able to extract the relevant data from the request. Dialogflow will use the language model together with the examples that you provide the agent to create an algorithm that is unique to your agent [47].</td>
</tr>
<tr>
<td><strong>IBM Watson</strong></td>
<td>Watson is a machine learning computer system developed by IBM that is trained by data, not rules, and relies on statistical machine learning to answer questions asked in natural language. Watson uses the DeepQA software and Apache UIMA (Unstructured Information Management Architecture) framework, both of which were developed by IBM. What makes Watson special is not a specific algorithm, but that Watson can simultaneously execute hundreds of algorithms [48][49].</td>
</tr>
<tr>
<td><strong>Facebook Wit.ai</strong></td>
<td>Wit.ai is a cloud service API, used for speech recognition and NLP. Wit.ai uses machine learning algorithms to understand the sentence. It also extracts the meaning of the sentence in the form of entities and intens [50].</td>
</tr>
<tr>
<td><strong>Microsoft LUIS</strong></td>
<td>Microsoft’s LUIS (Language Understanding Intelligent Service) is another system used for NLP. LUIS offers its users the possibility to either use a pre-built domain model, to build your own, or to use the best of both of the options. LUIS uses machine learning so developers can build their own natural language understanding applications. The significance of LUIS lies in that it uses</td>
</tr>
</tbody>
</table>
active learning to improve itself. What this means is of all the words/statements that the user speaks to the system, LUIS identifies the ones it is unsure of, and asks the developer to label them, thus leading to enhanced performance [51][52].

| **Amazon Lex** | Amazon Lex is a service for creating interfaces into applications using natural language. Amazon Lex combines the advanced deep learning functionalities of automatic speech recognition (speech to text conversion), with NLU to comprehend the intents of the text. Amazon Lex uses the same deep learning algorithms as Alexa for creating neural networks [53]. |
| **Web Speech API** | The Web Speech API uses a controller interface for handling input from the user and delegates the speech recognition to the default speech recognition service on the device/system. In our case, Windows 10 and Cortana and in the case of an Android app, Android speech. This makes the Web Speech API very versatile [67][68]. |
| Readers might also be interested in | IBM BlueMix (IBM Open Cloud), Google Cloud Natural Language API, Samsung Viv |
4.3.2 Low-level tools: working with techniques for natural language processing

Tools categorised as ‘low-level tools’ in this report are toolkits and libraries used for natural language processing, where the developer is allowed more control of the implementation-specific details of the natural language processing techniques. They are, however, not termed low-low level tools based on the availability of working with adapting algorithms and mathematical formulas. Developers interested in customizing natural language processing solutions will have more freedom using these low-level tools than the high-level tools listed in the previous section. Features offered by these tools include, for example: string preprocessing, training of models, classification, parsing, intent identification and sequence tagging. Table 7 describes a selection of low level tools.

Table 7. Low level NLP tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SyntaxNet</strong></td>
<td>SyntaxNet is a syntactic parser neural network NLP framework for TensorFlow. SyntaxNet handles uses part-of-speech tagging to tag each word with it’s syntactic meaning and attempts to discern the combined meaning of the words in a sentence. Parsey McParseface is a syntaxNet parsing model for English that can reach a 94% accuracy, current state-of-the-art, in finding the dependencies between the words of a sentence in English in well structured documents [54].</td>
</tr>
<tr>
<td><strong>Natural Language Toolkit (NLTK)</strong></td>
<td>Natural Language Toolkit was originally developed at the University of Pennsylvania and is an open source framework, collection of libraries in Python, that is commonly used for natural language processing tasks. NLTK uses the text mined from voice input to then make sense of the input by splitting it into words and finding semantic meaning. The main features include string processing, part-of-speech tagging, classification, chunking and parsing [55][56][57].</td>
</tr>
<tr>
<td><strong>SpaCy</strong></td>
<td>SpaCy is an open source Python/Cython library made primarily for the production of natural language</td>
</tr>
</tbody>
</table>
processing software. The developers behind SpaCy focus on staying up to date and use an object oriented approach to NLP handling. SpaCy is currently the fastest syntactic parser available and supports statistical neural network models for several different languages as well as deep learning and statistical models trained by machine learning libraries like TensorFlow [58].

| **Stanford Core NLP & Stanford NER** | Stanford’s natural language processing library is an annotation-focused NLP tool, available in most major modern programming languages and comes in two forms, the Core NLP and the simple Core NLP version. It aims to be easy to use and supports several languages. The development relies on Stanford's natural language processing research [59][60]. Stanford NER is a classifier used for entity recognition that also builds on Stanford research [40]. |
| **Apache Open NLP** | OpenNLP is a library for natural language processing that is based on machine learning. It supports many of the tasks most commonly used for natural language processing [61]. |
| **MALLET** | Machine Learning for Language Toolkit or MALLET is a library for applying machine learning tasks to text. It uses notable algorithms, for example HMM, MEMM and CRF, to solve common NLP tasks such as named entity recognition[62]. |
| **GATE** | General Architecture for Text Engineering, is a toolkit for NLP originally developed at the University of Sheffield, but is now used worldwide both commercially and educationally [63]. |
| **Alchemy API** | Alchemy is owned by IBM and part of IBM Watson. It uses machine learning in the form of deep learning to solve NLP tasks. The main focuses are semantic text analysis and face recognition [64]. |
### 4.4 Bridging the theory and the prototype development

The categorization of different techniques introduces another dimension to answering the research question by highlighting that there are many approaches to implementing natural language processing support, where the choice of technique should be based on the specifications and requirements of the final product.

To guide our own learning and understanding of techniques used in natural language processing, and linked to a specific approach for information extraction in this chapter, we chose a subset of tools from the categories described in the section about development tools and their categorization. The tools selected from the high-level category for the development of prototype one were; Web Speech API, for handling speech to text recognition; and Dialogflow, for handling of NLU. The subset of tools selected from the low-level category for the development of prototype two were: Weka for identifying intent using machine learning based on the techniques of sequence tagging, string pre-processing and training of models, as well as Stanford NER, for entity recognition based on the technique of sequence tagging.
5 Development of natural language processing support

This chapter is part of the result and contains sections describing; motivation behind the choice of techniques and tools, an overview of the prototype development as well as two sections describing each of the prototypes in more detail.

5.1 Choice of techniques and tools

The techniques and tools used in the project are selected based on how well the features offered correspond to what is needed for the development of the prototypes as well as with respect to the quality of the documentation of the tool or library.

5.1.1 High level prototype

The techniques used for the development of prototype one were selected from the ‘high-level tools’ section in the previous chapter. Table 4 describes the IDE and programming languages used for development of the first prototype.

<table>
<thead>
<tr>
<th>Description</th>
<th>Tool overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDE used for development</td>
<td>Netbeans IDE</td>
</tr>
<tr>
<td>Programming languages used</td>
<td>Javascript HTML</td>
</tr>
<tr>
<td>NLP tools</td>
<td>Dialogflow</td>
</tr>
<tr>
<td></td>
<td>Web speech</td>
</tr>
</tbody>
</table>

Table 4. IDE, NLP tools and programming languages used for the development of prototype one.

Prototype one is a demonstration of implementation of natural language processing support using tools selected from the high level category. Prototype one uses Dialogflow for extracting entities and intent and the choice of technique is based on the development approach described in the method chapter. The first prototype should be developed using tools and techniques from the most easily approachable category of tools. Dialogflow is a tool directed towards users of various backgrounds and natural language processing experience and offers features that cover all of the system
requirement specifications of the first prototype.

Google is responsible for great research breakthroughs in the natural language processing field and keeps its solutions up to date with the use of the latest state of the art techniques [47]. Google also offers many related services that could be utilized to extend functionality by being easily integratable with other solutions from their own cloud library.

5.1.2 Low level prototype

The techniques used for the development of prototype two were selected from the ’low-level tools’ section in the background. Table 5 describes the IDE and programming languages used for development.

<table>
<thead>
<tr>
<th>Description</th>
<th>Tool overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDE used for development</td>
<td>Netbeans IDE</td>
</tr>
<tr>
<td>Programming languages used</td>
<td>Java</td>
</tr>
<tr>
<td>NLP tools</td>
<td>Weka&lt;br&gt;Stanford NER&lt;br&gt;CMUSphinx(Sphinx4)</td>
</tr>
</tbody>
</table>

Table 5. IDE, NLP tools and programming languages used for the development of prototype 2.

The choice of technique for prototype two was influenced by our own interest in researching the possibilities of developing natural language processing support using Java as the main programming language. All choices were primarily based on the quality of the documentation (in addition to offered features) as this is integral to successful development.

CMUSphinx and more specifically the Sphinx4 library was chosen for working with speech recognition. CMUSphinx was chosen as it is a popular library for working with speech recognition and based on it being a Java library. It handles the voice capture as well as the transcription of speech to text.

Weka was chosen as it is a popular tool for working with machine learning and data mining using Java. Weka can be used both through a user-interface and directly through the framework. Prototype two uses the framework for both the string pre-processing, the training of the probability model and the classification of input using the model.
Stanford NER was chosen to identify entities in the input. The choice was based on Stanford NER being a Java implementation as well as for the quality of the documentation and its strong connection to state of the art techniques. Tools, like the Stanford NER, developed by universities that conduct research on natural language processing, give developers the chance to work with the latest state of the art techniques. We found this very intriguing. The choice of using the Stanford NER was based on the opportunity of gaining a wider perspective of tools and techniques.

5.2 The prototypes

Through the development of the prototypes we learned that the task of implementing natural language processing support stretches, like the field, across many levels of difficulty. Through the development of prototype one we learned how little code and configuration is required for implementing a simple natural language processing solution using cloud services. We also learned, primarily through the development of prototype two, more about the practical aspect of the underlying principles of natural language processing.

By working first hand with some commonly used natural language processing techniques we learned about how they can be utilized in combination to solve more advanced problems than what is possible when using them individually. For example, we learned about what data, we as developers, have to work with and what data is required for the different components in processing of language. The value of investing time and effort in training of customized models rather than using pre-trained models. How probability estimations and algorithms can be used for extracting meaning in text in the forms of entities and intent as well as about some of the limitations of using the specified tools. For example the high resource demands of in-real-time speech recognition.

5.2.1 Prototype functionality

Both prototypes boast the same basic functionality, the difference being that prototype one relies on cloud services and prototype two does not. The prototypes should be able to, based on the specifications stated in the method chapter, accept input in the form of spoken natural language and to then retrieve the information needed from the input to trigger actions.
5.2.2 Component blocks

Component blocks, introduced in chapter 4, in section 4.1, refer to blocks of functionally related components. The components from the model we adapted from Ranjan et al’s model are divided between the component blocks. This division was made to simplify implementation and to get a better overview of the dependencies between components. Block one contains the components most closely related to speech recognition and block two, the NLU components that handle, intent and entity recognition. Our requirements on the blocks can be summarized in the following way, as seen below.

Requirements on the component blocks:

**Component block 1**
Implementation of speech to text

**Component block 2**
Text input, without the use of component block 1, can be processed and relevant content is identified in the string. An action is triggered.

**Component blocks 1+2**
Component block 1 and 2 work together and it is not necessary to use text input to trigger an action.

5.2.3 Programming languages

Prototype one was designed to work first and foremost as a proof of concept but also to work as a demonstration of a simple implementation of natural language processing. Prototype one is a simple web page programmed using HTML and Javascript. Using Javascript it was easy to utilize the Web Speech API to record audio and to connect to the /query endpoint of Dialogflow, one of many options.

Prototype two was developed using Java as a programming language. Java was chosen, both because it is the language we, as developers, have the most experience working with, but also, more importantly, because we found it interesting to research the possibilities of developing natural language processing using Java. Many tutorials on natural language processing implementation as well as the available tools
are mainly directed towards programming natural language processing functionality using Python.

**5.2.4 Programming responsibilities**

While developing the prototypes maintaining low coupling and high cohesion was done to make it easy to adapt the design to work as part of a bigger system. Low coupling was achieved by keeping the dependencies between the different components low. Each component has its own specific task and variables within each component are kept local/private to not create unnecessary dependencies. Low coupling means the components can be easily detached and switched for other components with the same functionality. For example, the Web Speech API could be switched out to use another API for handling of speech to text transcription.

In both prototypes, high cohesion is achieved through the separation of functionality into ‘specialist’ classes/modules. Every class/module has its own purpose and unnecessary dependencies between classes are avoided.

In prototype one, the main module is divided into two main components; speech recognition handler, and dialog flow handler. The speech recognition handler handles the interaction with the Web Speech API and captures and transcribes the users voice into a text string that can then be handled by the Dialogflow handler and passed to Dialogflow as a query.

Both prototypes were, in addition, developed using the MVC pattern to maintain a low coupling and high cohesion. The MVC pattern helps in keeping the functionality of the different components modular.

**5.3 Prototype one**

*Is it possible, based on the project members level of knowledge at the start of the project, to develop NLP support for an application?*

Prototype one was developed based on the high level category of tools in section 4.3, where natural language processing is handled as a cloud service. The specific voice-command used for prototype one is: ‘create a note’.
5.2.1 Component blocks

The first component block in prototype one is Web speech. The Web Speech API offers both speech recognition and speech synthesis services, but prototype one only uses the Web Speech API for voice recognition. The Web Speech API receives the audio recorded by the users microphone and then computes the input based on probability estimations. When the audio has been transcribed into text the result is returned.

Component block two is Dialogflow. Dialogflow was configured through the web interface and is used to handle the NLU component of prototype one. Prototype one includes a feature called “small talk” which adds conversational features to the Dialogflow configuration of an application as an extra, simply because of the ease of implementation. The correct configuration is linked to the prototype, using an access token in the code. Dialogflow can be queried by sending a POST requests to a base URL. The query endpoint is used to process text input and the result of the processing, based on the Dialogflow configuration made, is returned in JSON format and contains the result of the query and other data relevant to the query. The component blocks of prototype one are illustrated in figure 7 below.

Figure 7. The component blocks of prototype one.
5.2.2 System abstraction

A system abstraction, and overview, of the functional components of prototype one is described in figure 7. The boxes represent the cloud services used for voice to text and natural language understanding as well as the application itself.

![Image](image.png)

**Figure 8.** System abstraction of prototype one

5.2.3 Package structure

An overview of the system package structure can be seen in figure 8. In prototype one the main module carries the responsibility of being both the controller and the model and is located in the ‘Controller/Model’ package. The main module is divided into two parts with separate functionality and responsibilities. The first part is responsible for the communication with the Web speech API, while the second is responsible for the communication and packaging of messages to be sent to Dialogflow. The ‘view’ package contains the two html pages used in the application. ‘NoteWindow’ is a window for taking notes that opens up on request and index is the main page where speech input is captured.

![Image](image.png)
5.2.4 Sequence diagram

The sequence diagram in figure 9 describes the interactions between the different component blocks of prototype one as well as the interaction with the cloud services.

When a keyboard button, when in the index page, is clicked the voice recognition using the Web Speech API is triggered. The recording stops when the button is let go and soon after a transcription of the speech received by the Web Speech API is returned. When the send button in index.html is clicked, the string is inserted into the AJAX object that is sent to Dialogflow for processing. A JSON object containing the processed string and entities and intents found is returned. This information is then used to trigger an action.

5.2.5 Implementation

Front end
The web page visible to the user is shown in figure 10. When a button on is pressed, the Web Speech API is called and speech recognition is started. The
captured speech is then inserted into the box above the send button. The send button initiates a chain of events that processes the input text and passes it on to Dialogflow for interpretation.

![Image](image.png)

Figure 11. Page layout of the website for prototype one

Functionality

Handle a spoken request

The spoken request is converted to text through the use of the Web Speech API. A webkitSpeechRecognition object is created to handle the recognition, but also to set event handlers.

The spoken request is transcribed and returned through speech.onResult(), meaning if there is a result after speech.start() and speech.stop() it is obtained. The result is then placed in the input box for the user to see before choosing to press send.

Interpret the meaning of the text

The text input is sent to Dialogflow using jquery through an AJAX request containing a JSON object. The meaning of the string arriving at Dialogflow is then interpreted through the use of agents, intents and entities configured in the Dialogflow web UI. The main intent used for prototype one is called ‘open_new’ and corresponds to the user intention of when a user wants to open, create, make or start something. All the words listed work as synonyms for the intent and signify what the user wants to do. The main entity used is called ‘note’ and represents what the user wants to create, make, start or open.
Trigger an action

The data returned from Dialogflow is handled and printed to show the intents and entities identified. See figure 12.

```
Action triggered: open
Entities found: note
```

**Figure 12.** Prototype one prints the Identified entities and the intent and executes an action based on these.

The triggering of the action is based on the identified intent and is executed by opening a new window for taking notes.

5.2.6 Evaluation

The evaluation is based on the functional requirements of the component blocks described in the introduction to the prototypes in detail. The gray box below gives an overview.

<table>
<thead>
<tr>
<th>Functional requirements</th>
</tr>
</thead>
</table>
| ➔ Component block 1: Handle a spoken request and convert speech to text  
  Spoken request: “create a note”.  |
| ➔ Component block 2: Extract relevant information from the text and trigger an appropriate action.  |

Component block 1, speech to text is handled and works as intended. The speech recognition had a very low error rate even if the native language of the speaker was not English. However a reliable error rate estimation would have to be based on a larger number of phrases, and testing of speech recognition accuracy is not part of the scope of the project.

Component block 2, the natural language understanding, consists of extracting the intent as well as the entity/entities of the input sentence. The intent is identified correctly provided the input is based on the request. Intent
can be detected based on several triggers.

The action, a window for taking notes is opened, is triggered if the intent ‘open_new’ as well as the entity ‘note’ are identified.

Testing of component blocks one and two together was done by connecting the input of the second component block to the output of the first component block. The string returned by the Web Speech API was inserted into the AJAX object and passed to Dialogflow for evaluation. Everything worked as intended.

5.4 Prototype two

The purpose of prototype two was to gain a deeper understanding about the underlying principles of natural language processing and to based on this gain an understanding of the complexity and value of developing a customized engine. Prototype two was developed based on the low-level category of tools in section 4.3. The specific voice-commands used for prototype two are different ways of expressing the wish to purchase a train ticket, ordering a taxi or initiating or ending a conversation (using greeting or ‘goodbye’ phrases).

5.4.1 Component blocks

Component block one is made up of the speech recognition component. In prototype two voice capture and speech recognition is handled through the use of the Sphinx4 library. Component block two consists of the string pre-processing component as well as the NLU the two sequence tagging components. The string pre-processing is handled through the use of the Weka library and the sequence tagging is handled by both the Stanford NER and the Weka library. Weka is used for classification of the intent and the Stanford NER is used for entity recognition.

The Weka classifier for intent is trained using the Naive Bayes algorithm, which would ‘classify’ it as Naive Bayes classifier. Naive Bayes was selected based on its good performance in small datasets. To train the classifier, a dataset containing examples of possible input along with corresponding labels is needed. Our dataset is a small collection of sentences where the sentences belong to the classes ‘greeting’, ‘taxi’, ‘train’ and, ‘goodbye’. The Naive Bayes classifier is described in detail, using our implementation as an example, in the background chapter in section 2.8.
The Stanford NER is pre-trained and is primarily used for extracting locations, but can also be used to identify several other entity types, including people and time. Figure 13 describes the component blocks of prototype two.

![Figure 13. The components and component blocks of prototype two.](image)

### 5.4.2 Package structure

An overview of the system package structure can be seen in figure 14. Prototype two, like prototype one, was developed with the MVC pattern in mind. The aim was to simplify the process of exchanging individual components if the need should arise as well as to simplify the extraction of a component for use in another system.

The ‘view package’ contains the main class and is responsible for input as well as to relay the input information to the controller in the ‘controller package’. The controller handles to calls to the different components of the model and is responsible for the communication with the view. The ‘model package’ contains classes for handling voice recognition, training of the Naive Bayes classifier, classification of intents and recognition of entities.

![Figure 14. An overview of the system package structure.](image)
5.4.3 Sequence diagram

The sequence diagram in figure 15 is a simplified sequence diagram for prototype two. The calls to methods that do not add clarity to the main sequence of actions have been omitted.

![Sequence Diagram for Prototype Two](image)

**Figure 15.** A sequence diagram of prototype two.

5.4.4 Implementation

Prototype two was developed using techniques from the low level techniques, where the processing of input is handled locally. The Sphinx4 library was used for the implementation of component block 1 and the Weka and Stanford NER libraries were used for the implementation of component block 2. The input and output of prototype two is handled through the console and it is possible to run prototype two using both speech and text as input. Figure 16 is an example of a typical output from prototype two and figure 17 is a collection of charts displaying the probability of the intent based on our model.
Figure 16. Two examples of output from prototype two. The first row is the input sentence, the second is the sentence annotated using the Stanford NER, the third is the intent identified using our own classifier, the following four rows are the classified probabilities of the intent belonging to either of the four classes (train, taxi, greeting, goodbye) and the last row is the print out from the action that is triggered by the information retrieved from the sentence.

Figure 17. The charts describe the probability of each intent belonging to each of the classes (3: goodbye, 2: greeting, 1: train, 0: taxi) on the x-axis including +/- deviation. The coloured bars show the correct guess based on the input sentences and the gray bars show the remaining alternatives, incorrect guesses. Probability output from prototype two using five different sentences belonging to each class, were used to calculate the average probability of each individual output.
Functionality

Handle a spoken request

The Sphinx4 library, part of the CMUSphinx toolkit, is used to handle voice recognition (the dictionary, the acoustic model and the language model were not trained by us and can be downloaded from the CMUSphinx website. The pre-trained models were used as speech recognition is not the prime focus of this project).

Voice recognition starts automatically when the application is run, after the initial loading phase. The application supports both input through sound files and through microphone to ease experimentation. A hypothesis of each word is fetched and put together into a sentence, before being sent on to the second component block for handling.

Interpret the meaning of the text

After the input is received from the speech recognition component or input is provided in the form of text, a sentence object is constructed. The sentence carries information retrieved about the input and stores it for later use (this approach was used to simplify the process of running several types of information retrieval tasks as well as several different ones on the same sentences, while also making it easier to process documents one sentence at a time and organizing the results in an orderly fashion).

The first information retrieved from the input is the intent. The process of identifying the intent (as a greeting, a wish to purchase a train ticket, book a taxi or say goodbye) as well as the training of the classifier is done using the Weka library.

Through the Weka library, we use the Naive Bayes algorithm to estimate the probability of a sentence referring to a specific user intent. The algorithm is used to create a classifier by using our annotated input data to train it (the dataset can be found in the GitHub repository of prototype two).

The classes used in this project are a set of intents (train, taxi, greeting and goodbye). The output is a classifier model trained with our specific dataset.

The probabilities of the full sentences belonging to each of the classes $c_1$, $c_2$, $c_3$ and $c_4$ are all calculated by multiplying the individual ‘word-belongs-to-class’ probability values together, then by multiplying them with the probability of the class itself and finally by dividing them by the evidence (the probability of encountering a specific pattern independent from the classes) to normalize the result. The result of the probability of the
sentence belonging to the classes is then used to compare them to each other to see which class it most probably belongs to. In figure 16, the result is the ‘greeting-class’.

**Trigger an action**

The actions triggered are based on the intent identified, a greeting triggers a greeting, a request to buy a train ticket or book a taxi triggers the simulation of an appropriate app opening and the goodbye intent triggers a goodbye.

### 5.4.5 Evaluation

The evaluation is based on the functional requirements of the component blocks described in the introduction to the prototypes in detail. The gray box below gives an overview.

<table>
<thead>
<tr>
<th>Functional requirements</th>
</tr>
</thead>
</table>
| **Component block 1**: Handle a spoken request and convert speech to text  
Spoken requests concerning the intents, taxi, train, greeting and goodbye |
| **Component block 2**: Extract relevant information from the text and trigger the appropriate action. |

Component block one, Voice recognition works by recognizing and transcribing spoken language, however no time was put into fine tuning the acoustic model or training of a customized one, which is needed for achieving good results. Hence the error rate is noticeable enough to be an annoyance when running the system. However Sphinx4 uses state of the art algorithms and it is possible to achieve very good results given more time is invested. The error rate is also dependant on the process of voice recognition being very sensitive to the input format. Different microphones deliver different sound formats and the prototype does not support any type of input- formatting. Voice recognition on its own is also a very demanding task and the prototype only uses one thread to separate the process from the main thread to prevent the application from hanging, thus the processing takes a
few seconds to complete. To improve and customize the voice recognition one more step of statistical evaluation could be inserted after collection of the hypothesised words to improve the ‘correctness’ of input.

Component block 2, the intent recognition, works very well and in almost all of the cases the correct intent was identified. The only cases when this was not the case was when the sentences diverged too much from the type of sentences in the training set. The entity recognition, although being much slower than our custom intent recognition, always identified the correct entities if they were locations and people but not if they belonged to the ‘time’ category. The appropriate action was triggered based on the intent being correctly identified.

Testing of component blocks one and two together was done by connecting the input of the second component block to the output of the first component block. The option to allow speech recognition through inputting a sound file instead of using only the microphone was added to achieve better results. The option to exclude speech recognition was also added to be able to test the component blocks separately. Everything worked as intended, but the models used for voice recognition as well as the dictionary need further work to achieve a satisfactory level of accuracy.

5.5 Alternative solutions

For our first prototype, we chose to use Dialogflow. The reason for our choice is motivated under 3.5. Alternative high level techniques that could have been used are listed under 4.1, such as IBM’s Watson, Microsoft’s LUIS etc. Considering also that prototype 1’s front end is a simple website, the coding language that seemed like the most obvious choice for us was Javascript, but as most of the high level solutions support many programming languages, any could have been chosen. There is specifically very good support throughout the natural language processing community for Python.

For prototype 2, we chose to do the coding in Java. The prototype could have been developed in different programming languages too, such as Python and #C. For entity recognition, we chose to go with Stanford NER. A full list of low level techniques can be found under 4.1. Alternative low level techniques that we could use for entity recognition could be spaCy or Apache OpenNLP, both of whom feature statistical NER.

As a summary, any of the tools from the two categories of techniques can be used to develop the same solution. Though, when choosing techniques it is important to keep in mind that not all tools described offer the same features. Tools that offer the individual features, like string pre-processing and sequence tagging, are needed no matter the combination of tools chosen.
for development of natural language processing support. Techniques from both categories can also be combined in the cases where this would achieve the best results.
6 Discussion and conclusion

This chapter contains the discussion of the research questions and the motivation for validity and reliability of the methods, and the result, as well as a discussion of the concerns that were brought up during the course of the project about natural language processing from an ethics, sustainability and economical view point. The conclusion is included in this chapter under the heading concluding remarks.

6.1 Implementing natural language processing support

How can natural language support be implemented in an application?

There are many ways to approach a natural language processing task, depending on its purpose and the amount of control and customisation needed. There are good recommendations for how natural language understanding can be utilized to retrieve meaning from text however, the natural language processing field is still in its infancy and the state of the art is not a static domain. Natural language processing is a very big area where the divisions within the field are not always clear. Many different tasks can be solved using similar solutions, even if their goals vary. We learned through the literature study that there are many smaller subdivisions of the field. One of these subdivisions has to do with the retrieval of information from text, while others deal with topics such as translation of documents between languages. The goal of each area is not necessarily closely related, but the connection is made through the use of similar techniques to solve a problem. The different subdivisions within the natural language processing field are bound together by their reliance on machine learning. It is a popular approach to describe language using grammatical rules, however, statistical approaches, as demonstrated in prototype two, are surprisingly effective at finding meaning in language.

In the result chapter a division between different types of tools was made. We found that there are two major approaches to developing natural language processing support for applications. The approach with the least amount of barriers between a developer and a working implementation is the high-level approach. These solutions are an easily scalable package deal that distance the programmer, to different extents, from the inner workings of natural language processing. This also sometimes interferes with the programming of fully customizable content. Using high level-tools implies relying ‘third party natural language processing engines’ that will, in a sense,
always be outside of the developers control. The positive aspect of relying on third party solutions is that the maintenance and updating of the natural language processing components is handled by the third party company, often, for a very affordable price. The value of using high-level tools is therefore dependent on how much the natural language processing functionality of the application relies on being fast, efficient and up to date to be competitive and whether the application has to be truly competitive to be ‘competitive’. For example does a natural language processing component need to be 94% accurate (approximate accuracy of state of the art parsing techniques) or is it sufficient if it is only 92% accurate? Does the processing of language need to happen in real time or is there a margin for some delay?

When considering using high level tools and approaches, it should also be taken into consideration that it is possible to customize these solutions to a great extent, and that it is possible to utilize only part of these solutions in your own implementation.

A natural language processing engine with clear limitations to a specific area of the field, such as information retrieval, and limited application scope will benefit from the customizability that can be achieved from using the low level tools, and through them the techniques. Low level techniques can be used to develop fully customizable solutions, where developers can add support for whatever features they find interesting in language. These solutions do not come between developers and state of the art solutions, they only require more from the developer to be ‘competitive’. There are also cases where high level tools would fit the needs for natural language processing in an application, but it might not be the wisest decision from an economical standpoint to use them. For handling of a simple natural language processing task, that exceeds some limit of usage, a well grounded decision would be to use an open source library.

There are many tools that can be used in different combinations to aid in developing natural language processing support for applications. A lot of the best tools available for natural language processing are centered around the Python programming language and there are many alternatives in every category, from string pre-processing to sequence tagging to machine learning libraries. The amount of tools available to work with using Java as a programming language are fewer, but the tools available are still on par with the corresponding Python tools, in terms of the effectiveness. Many tools can be used with more than one language and support for more languages is often developed based on demand.
6.2 Validity and reliability of the methodology

6.2.1 The research method

Our research method is based on Bungen's scientific technological research method. The main criteria for evaluating the research method is whether or not we were able to answer the research question. To successfully answer the research question ‘How can natural language-support be implemented in an application?’, we had to create some system/prototype to try implementing natural language-support into.

The first natural step of our research method, was to understand the problem. Considering that we were new to the field, this first step was a good start to get familiar with the task at hand. Before we move on to the next step, we have to make sure that we have sufficient information regarding natural language processing and that we understand the problem, know how to tackle it and know what tools/techniques we have at our disposal. The next step would be to develop a prototype. As mentioned previously, this step is necessary in order for us to answer our research question. We need some sort of system/prototype to implement natural language-support in.

Validity refers to whether the research method successfully measures what it was intended to measure. Meaning that the collected information can be used to answer the research question. Taking the project question into consideration, ‘How can natural language-support be implemented in an application?’, the first natural step for someone like us would be to research what exactly natural language processing entails. But also, to answer the ‘How’, we need to research the different approaches, techniques and tools available for implementing natural language processing in an application. Each step also requires evaluating the achieved result, this helps us answer the question and mainly ‘How’ it can be done, therefore our research method is a valid one.

Reliability refers to the trustworthiness of a measurement and its methods. Meaning that if someone else were to conduct the same study/experiment, under the same circumstances, the result will be the same. Considering that we conduct more of a qualitative research rather than a quantitative one, both validity and reliability are harder to measure in a qualitative research than they are in a quantitative one. Some in the scientific community even argue that since reliability issues concerns quantitative measurements, it is irrelevant in qualitative research. Others argue that to ensure reliability in qualitative research, reliability lies specifically in the trustworthiness, meaning that examination of the trustworthiness is
essential. [67] To ensure reliability and trustworthiness in our research method, the literature study gives us a foundation of understanding based on previous achievements and discoveries within the field. Next step for us was to research and work with different methods, algorithms and tools that can be used to implement natural language -support. Even of algorithms and tools that we have not used in any of our prototypes, but by doing so we have the ability to make a better judgement of which algorithms and tools that suits our project the best.

6.2.2 Alternative methods

Alternative methods for gathering information could be for instance interviews. Interviews would serve a similar purpose as a literature study. Although interviews could be a useful way of getting insight and information about NLP, one problem is that, as previously mentioned, the field of NLP is relatively new and not fully established yet. This may result in it being more difficult to find the ‘right’ people to interview.

Another alternative method for gathering information could be surveys. Surveys would also have to be aimed at the properly qualified persons, meaning people with work experience or research experience within the field of NLP. For both interviews and surveys making sure that the target group is qualified is vital, if not this will have a negative impact both on the methods validity as well as reliability.

6.3 The result

The result is based on findings from the literature study, the categorization of techniques and the tools, as well as the prototypes that were developed. When researching questions with strong ties to the implementation or development of systems, as an engineer the most logical way to try to approach the problem is to take an engineering approach, to develop something and gain deeper insight about the problem along the way. Reading about natural language processing development techniques introduced us to the concepts, the development of the prototypes gave us first hand experience and proof of concept. The prototypes demonstrate two working examples of how natural language processing support can be implemented and hence give a good foundation for the discussion.

In retrospect, regarding the probability graphs of prototype two, it might have been fairer to classify correct or wrong answers as yes or no, but we wanted to show the variability in the predictions. to give the reader a better idea of the underlying principles of our intent identification. For
example, prototype two’s uncertainty in determining if the intent was train or taxi is made clear even if the correct intent was selected.

### 6.4 Sustainability, economic and ethical aspects

An important aspect of this thesis and natural language processing is information extraction. For example, we use named entity recognition and intent recognition in our prototypes to extract information, key words, such as locations and user intent such as a wish to book a ticket for a train. Information extraction can be utilized in many different areas, such as for the stock market. Algorithmic trading, which is a sort of trading method that is completely controlled by technology, has been increasing in popularity for the past year. Many of these financial decisions are impacted by news and journalism, which the majority of is in English. What natural language processing then can do is extract the relevant data out of the articles to then take into account into the trades the algorithms do. The relevant data could be information about a merger happening, the prices etc [64]. Natural language processing also has similar potential applications in the healthcare industry. For instance, when it comes to the treatment and detection of Cancer, in 2014, 140,000 academic articles were released [65]. This is too much information for anyone to read and determine what kind of treatment is best for any specific case. This is also where information extraction comes in handy, where extracting data based on certain keywords can provide better and more relevant result.

The natural language processing field is developing at a fast pace and it makes sense that more and better tools will also be developed to cater for the request of more and better tools for natural language processing development. These tools will make natural language processing more easily accessible to developers in a number of different contexts and the number of applications using some form of natural language processing will increase. This chain of events will affect economic, sustainability and ethical aspects of society. Using information extraction in the finance industry has clear connections to effects on the economy, while using information extraction for healthcare triggers possible ethical concerns.

Information retrieval can be used to improve efficiency and increase utility. The combinability of information retrieval and natural language highlights the importance of ethical concerns. Companies that develop domestic natural language processing solutions, like Google home and Amazon alexa, get potential access to a lot of data that ‘the generators of the data’ (i.e. the end users of the applications implementing natural language
processing.) have not necessarily given approval of for collection, in addition this data is going to be used for commercial purposes. This raises ethical concerns regarding the commercialization of such data [69][70].

Other ethical concerns within the field of NLP, are demographic bias. These concerns are mainly related to voice recognition, even though voice recognition is not the main focus of this project. The data set used usually does not take into consideration how the demographics of our societies look in reality. In NLP, the ethical concern raises due to the assumption that all language are identical to the training data, which is why they perform worse on certain demographics. An example of this is that standard language technology may be easier to use for white males from California, than for women or citizens of Latino or Asian descent. This is because white males from California are the group taken into consideration when developing standard language technology [71].

6.5 Concluding remarks

6.5.1 The project in context

We believe that this project will be useful for engineers as well as, in a broader context, to people who want insight into the field of natural language processing. This report gives a good overview on how to approach and develop natural language processing support for applications, containing algorithms used within the field, tools and frameworks that can be used for development of both high and low-level techniques depending on requirements. It also provides a tested method of approach to natural language processing and its various levels of implementation and how to conduct the project from start to end, taking potential limitations of the project into consideration.

6.5.2 Future work

Suggestions for where future work can continue on from where this project ends are to use the same approach used in this project, but add more iterations to every step to research the possibilities of creating customized solutions that can handle multi-intent recognition, look into approaches utilizing neural networks as well as a combination of statistical and neural network approaches and experiment with bigger datasets. We recommend iterative and incremental development, starting from a simple case and then go into more detail, to better understand the tools and techniques as well as
their advantages, disadvantages and limitations. It would also be relevant to shed more light into implementation of speech recognition and fine tuning of acoustic models, as this was a highly relevant area that had to be excluded from the scope to limit the project. What are the requirements for ‘in real time’ speech recognition? Relevant for future work could also be to approach natural language processing, in the context of information retrieval, from an ethics, sustainability and economical viewpoint, as these are topics that are highly relevant when developing natural language processing for the future.
References

**GitHub repository:** github.com/NLPproject2017


4. Condliffe J. This year, we learned to love AI assistants in our homes [Internet]. MIT Technology Review. 2016 [cited 7 January 2018]. Available from: https://www.technologyreview.com/s/603228/in-2016-ai-home-assistants-won-our-hearts/


12. Friedman, C., Rindflesch, T. and Corn, M. (2018). *Natural language processing: State of the art and prospects for significant progress, a workshop sponsored by the National Library of Medicine*. [online] Available at:


30. Ling et al. (2015b)


63. GATE.ac.uk - index.html [Internet]. Gate.ac.uk. 2017 [cited 26 November 2017]. Available from: https://gate.ac.uk


67. [Internet]. w3. [cited 6 January 2018]. Available from: http://www.w3.org/


69. Laney D. Infonomics: the new economics of information. Financial Times [Internet]. 2013 [cited 30 December 2017]. Available from: https://www.ft.com/content/205ddf5c-1bf0-11e3-b678-00144feab7de

