LAPPEENRANTA-LAHTI UNIVERSITY OF TECHNOLOGY
School of Engineering Science
Software Engineering

TRANSCRIBING SERVICES AND TEXT ANALYSIS

Master´s thesis that was sent for inspection on 2.10.2019. Thesis subject was approved on 27.9.2019.

Author: Timo Kiviharju
Examiners: Professor Ajantha Dahanayake
MPhil Jarno Tenni

ABSTRACT

Lappeenranta-Lahti university of technology
School of Engineering Science
Software Engineering

Timo Kiviharju

Transcribing services and text analysis

Master´s thesis
2.10.2019
57 pages

Examiners: Professor Ajantha Dahanayake
MPhil Jarno Tenni
Keywords: transcript, speech-to-text, text analysis, data mining

The goal for this project is to analyze how to successfully transcribe phone calls in English and Finnish by using transcript services, and how to store the resulting data so that it can be used in computer analysis. In this project the performance, quality, usability and limitations of a transcribing service will be examined. It is investigated what is required from storing transcripts in order to be able to use them efficiently with computer analysis.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
School of Engineering Science
Tietotekniikan koulutusohjelma

Timo Kiviharju

Litterointipalvelut ja tekstianalyysi

Diplomityö
2.10.2019
57 sivua

Työn tarkastajat: Professori Ajantha Dahanayake
FM Jarno Tenni

Hakusanat: transkriptio, litterointi, tekstianalyysi, tiedonlouhinta

Projektin tavoitteena on analysoida miten litteroida puhelinkeskusteluja suomeksi ja englanniksi, ja miten tallentaa tuloksena syntyvä tieto siten, että sitä voidaan käyttää tietokoneanalyyseissä. Tässä projektissa tutkitaan litterointipalveluiden suorituskyky, laatu, käytettävyys ja rajoitukset. Lisäksi tutkitaan mitä vaaditaan transkriptioiden tallennukselta, jotta niitä voitaisiin tehokkaasti käyttää tietokoneanalyyseissä.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the help of some key persons and institutions. I´d like to thank the two most important persons in my life: my wife and daughter. They encouraged me to finalize my studies even if that meant being widowed for many Saturdays and Sundays and even a major part of Summer holidays. I´d also like to thank my employer, who not only encouraged, but also facilitated this thesis. Esa Kuparinen, my dear colleague, thank you for moral and peer support.

And last but not least thanks to Lappeenranta University of Technology for having me as a student for 21 years. I´m not sure if I will bear the cloak of studying the longest and still managing to graduate from LUT, but I will be quite high on that list. This is a huge personal triumph. I can finally put down the huge boulder that I´ve been carrying on my back for all these years.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND SYMBOLS

API	Application Programming Interface
ASR	Automatic Speech Recognition
CER	Character Error Rate
CS	Customer Service
CSV	Comma Separated Values
DBMS	Database Management System
DM	Data Mining
EU	European Union
GCS	Google Cloud Storage
GDPR	General Data Protection Regulation
HHM	Hidden Markov Model
IBM	International Business Machines Corp.
iMDB	Internet Movie Database
LER	Letter Error Rate
NLP	Natural Language Processing
NoSQL	Not only SQL
RDBMS	Relational Database Management System
REST	Representational State Transfer
SAS	Shared Access Signature
SQL	Structured Query Language
TF	Term Frequency
TF-IDF	Term Frequency and Inverse Document Frequency
TM	Text Mining
USD	United States Dollar
WER	Word Error Rate

1. INTRODUCTION

The complexity of the world, fierce competition and the proliferation of data has greatly affected how business strategies are formed. Instead of just going with a gut feeling, decisions are based on actual data (Singh, 2018). This is called data-driven or data-informed management (Perkin, 2017). The amount and quality and most of all the ability to understand that data is critical to companies to allow them to make correct decisions. For companies, that offer products or services, some metrics can be acquired by using for example questionnaires like webforms. This will provide data that is already in a format easily used in computer analysis. However there´s still a lot of data available that is completely left unused simply because the format was not suitable for automatic processing. Textual review is a good example of data that in the past required a person to analyze. In the past it would have required a person just to tell if a review was positive or negative, not to mention what is good and bad about the product. Combine this with a huge amount of reviews, and the result would have been a time consuming and tedious process. With the evolution of machine learning, natural language processing (NLP) and text mining etc. it has become possible to acquire useful information out of human written textual documents by computer analysis (Vivek, 2018). But there´s still a lot of data that is left unused: the data that is non-textual format.

Companies in general provide services over phone, for example customer services or informational services. In some cases, especially with insurance and banking sector, the discussions are recorded for future analysis and legal purposes. In the past all this data, a huge amount of phone calls, was either entirely left out of the scope of computer analysis or it required heavy human interactions and convert them into a compatible format. If these were usable in computer analysis, companies could use the data to find out ways to optimize the product: what are the reasons why customers are calling a customer service line and what´s good and bad in the product (Vivek, 2018). Or the same for customer service itself: which person has been most successful in solving which customer service issue. Advances in speech recognition, a machine learning and NLP task itself, have made automated transcribing possible. This makes it possible to convert

audible speech into textual format without human interaction. In the past automated transcripting was limited to specific speakers and vocabulary, for example doctors doing medical dictations (Enarvi, 2018). Current transcription services claim to be able to decipher colloguial speech, previously a big challenge in automated transcripting (Lardinois, 2019). Many information industry giants, like Google, Microsoft and IBM, already offer transcription services for this purpose. An unknown factor with all these services is their transcription quality: how accurately can a service transcribe a phone call recording, where the speech is conversational instead of grammatical and audio quality is less than optimal.

After getting the transcripts the next problem manifests; what is required from the data so that it can be used for text analysis ranging from simple keyword searches to complex mining and machine learning solutions. There are several different ways how the data can be stored, whether it be in basic text files or in a database of sorts. Most likely the amount data resulting from the transcriptions will be big, even to the point that it can be called Big Data. Big data brings it´s own set of problems which have to be considered.

1.1. Research questions

The thesis has two main focus areas: the transcribing of phone calls and storing the transcripts in such a way that they can be used for computer analysis. The main research questions are:

How to transcribe phone calls with speech-to-text services?

What is the best way to store the transcripts?

The transcribing part has following sub-questions:

How do the properties and usage vary between speech-to-text services?

What is the quality of the transcripts with phone calls?

What is the quality of the transcripts with English and Finnish language?

The storage part has following sub-questions:

What are the benefits and drawbacks of using text files or databases?

How do different database solutions differ from each other regarding computer analysis?

1.2. Goals and delimitations

The main objectives of this thesis are to investigate and compare speech-to-text services with phone calls and preparing the resulting transcripts for computer analysis. The properties, usability and resulting quality of two transcription services shall be investigated and compared to each other. Storing of the resulting transcripts shall be investigated to make the data suitable to be used in computer analysis.

1.2.1. Transcribing phone calls

The main problem is to find a suitable service which is able to transcribe phone call recordings, in other words recordings that are with less than optimal sound quality. The service also must have support for Finnish language, a possible problem for the language being one of the hardest and least spoken languages globally. Secondary factors that have to considered are privacy, usability and pricing issues. Phone calls are user sensitive data, so great care must be taken to ensure that all regulations, like GDPR, are met. The ease-of-use of the API is another factor, which has direct impact to how fast or expensive the API is to take into use and maintain. Transcribing is not free so the pricing models have to be investigated.

Google speech recognition API is the service that was chosen as the main candidate for this investigation. It supports Finnish language and the company is a known performer in speech recognition solutions (Google, 2019). Examples of this are YouTube and Google Maps, which both feature speech recognition on mobile devices and are able to interpret even long and hard Finnish sentences. Google´s service shall be compared to Microsoft Azure.

There seem to be no studies available on how any of the available services perform with poor quality recordings or with Finnish language, so these have to be tested and evaluated. The properties of the services, usability of their APIs and the transcription results of the two services are investigated and compared.

Due to the costs of the services, as using them is not free, massive quality or performance testing cannot be done. The speed of transcripting will be measured by one-by-one tests instead of parallel transcripting.

1.2.2. Storing the transcripts

Databases are the most convenient method for collecting data for data science. However transcribing a huge amount of calls will result in a "big data" problem. It has to be thought out carefully how to store the data in such a way that is easily available for future use. Just dumping it into database will most likely be very slow to read and thus making it hard to use for making deductions. This thesis should able to describe how to store the data in such a way that it would be best available for computer analysis ranging from simple searches to complex machine learning implementations. Database technologies like MySQL and ElasticSearch should be studied. MySQL might not be the fastest or most modern way to store data, but it´s free, supports free text search to some extent and is the most commonly used database solution. The ability of a database to be able to quickly join all kinds of data makes a huge difference compared to a set of csv files for example (Zuriaga, 2013). ElasticSearch on the other hand natively supports Term Frequency and Inverse Document Frequency (TF-IDF), a common technique used in text mining (Elasticsearch, 2019) (Leskovec;Rajaraman;& Ullman, 2014).

This thesis will investigate how to store phone call transcripts in such a way that they can used in for example computational analysis. For example what would be required from to data so that it could be used with a well known machine learning algorithms like Naive Bayes classifier. Doing actual computational analyses, like testing with actual machine learning algorithm, is outside the scope of this project. Rather this thesis will provide information what to consider from data storage point-of-view be-fore starting to work with actual machine learning implementations. Using the data in machine learning will be a fairly complex ordeal, because Finnish language will make using the data more difficult. In machine learning the text is usually preprocessed by methods like stop-word removal and stemming, or an advanced version of stemming called lemmatization. These can be problematic with Finnish language which has complex inflected forms and grammatical cases, hence it will only be studied how to make the data available instead of doing actual machine learning trials.

1.3. Research methods

The main research methods for this thesis are literature review and experimenting. For transcripting part the goal of literature review to help understand the problem field of converting speech to text and also to investigate any existing studies about implementations. This should provide a view what the current technology and implementations are capable of and direct the research into using implementations that support the goals of this thesis. Transcripting shall be then experimented by doing trials with speech-to-text services. This will result in an analysis of service properties, usability and transcripting quality.

For storing the transcripts the literature review should provide ideas how the transcripts can be utilized and what is required from storing the them. This will result in an analysis of potential storage options; what is possible and what isn’t, and what are their benefits and drawbacks.

1.4. Document structure

Chapter 2 covers related works about both transcripting and data analysis. Chapter 3 presents an overview to the field of converting speech to text. Chapter 4 investigates the properties of Google Speech-to-Text API and how to use it. Chapter 5 does the same with Microsoft Azure Speech to Text API. Chapter 6 compares the actual transcription results of the two services and analyzes how successful the transcriptions are. Chapter 7 investigates what is required from the data, ie. the transcripts, so that it could be used for computer analysis. Chapter 8 contains the conclusion of this thesis.

2. RELATED WORKS

This chapter contains the analysis of books and studies related to this thesis. Since transcripting and text analysis are only indirectly related to each other, this chapter is split into two subchapters: one about transcripting and one about text analysis.

2.1. Transcribing

One of the specific goals of this thesis is to convert colloguial Finnish dialog into text. A study that covers this extremely well is Seppo Enarvi´s doctoral dissertation "Modelling Conversational Finnish for Automatic Speech Recoqnition". Enarvi describes how the technology has evolved, from early solutions which were tied to specific persons and to lexicon of a specific field, to more modern that even capable of recognizing colloguial speech. What differentiates Enarvi´s study from most of the others is the focus on recognizing Finnish language, especially colloquial Finnish. What makes speech recognition of conversational Finnish challenging is differences in pronunciation and grammar if compared to grammar. According to Enarvi, conversational Finnish had no research nor corpora available when the work on his thesis started in 2012. The resulting ASR solution was able to achieve 27,1% WER for Finnish and 21,9% WER for Estonian. (Enarvi, 2018)

In a study about under-resourced languages from 2014 there was no mention of Finnish being under-resourced (Besacier;Barnard;Karpov;& Schultz, 2014). However Finnish language is described as morphologically rich, agglutinative and inflective. In agglutinative languages "word-forms can be composed of root (stem) preceeded or followed by up to a dozen grammatical affixes" (Besacier;Barnard;Karpov;& Schultz, 2014). The study also discusses morpheme-based models, and an unsupervised word decomposition software known as Morfessor, which is also mentioned by Enarvi (Enarvi, 2018).

Regarding the automatic speech recognition services and APIs, a study that is directly related to the experimental phase of this thesis is Këpuska`s and Bohouta´s speech-to-

text API comparison done in 2017 (Këpuska & Bohouta, 2017). They compared Microsoft API, Google API and CMU Sphinx-4 to each other. Interestingly the study also sheds some light to the background of each of the implementations. The experiments were done with audio files from TIMIT corpus and ITU. Audio files were all in English and were 8kHz and 16 kHz WAV files. The conclusion of the test was that Google was superior with 9% WER (word error rate), while Microsoft got 18% WER and Sphinx 37% WER. (Këpuska & Bohouta, 2017)

In 2019 Bogdan Iancu studied the use of Google Cloud Speech-to-Text API to be used for speech recognition of multimedia e-learning resources in Romanian (Iancu, 2019). The reasoning behind this was to make the material more searchable, which related to the text analysis part of this thesis. The source data was in format of Youtube videos. Romanian language is listed as one of the under-resourced European languages (META-NET, 2019). According to the study, Google was the only one providing support for Romanian language, and hence it was the only service that was experimented with. The result was 30,96% WER, though some videos were able to achieve as low as 9,93% WER. The study suggests that the reason behind the varying transcription quality is the lexicon used in the videos, as the videos were from different fields of study, and also the qualitative properties of the videos. The latter relates to the fact that this study is going to utilize phone calls which are of sub-optimal quality. (Iancu, 2019)

In comparative study of speech recognition systems CMU Sphinx was deemed the most interesting of open source systems. From the closed source options Dragon Naturally-Speaking as said to be the most accurate one but also had the most complex pricing model. Google API was described as convenient, embeddable and fast. The study suggests that closed source solutions will be faster and more accurate than open source solutions. (Matarneh;Maksymova;Lyashenko;& Belova, 2017)

Many articles and comparisons found regarding speech-to-text services were not of scientific origin and were often written by companies that are offering their own solutions.

The latter makes the objectivity of such articles questionable as they are often just for promoting the products of companies.

2.2. Text analysis

The literature review for text analysis focuses on various data storage and management technologies, what benefits and disadvantages do they have and how to they compare.

Regarding text mining and databases, there was a comparative study about NoSQL and relational databases (Ribeiro;Henrique;Ribeiro;& Neto, 2017). The goal of the study is to experiment the differences in obtaining the most frequent N-grams with relational (MySQL) and NoSQL (HBase) databases. N-gram is an N-sized vector of letters or words and is often used in text mining solutions. Performance testing shows that NoSQL solution was 18 times faster than a relational database. The authors highly recommend the use of a NoSQL solution for text mining.

Another comparative study compared MySQL and MongoDB (Györödi;Györödi;Pecherle;& Olah, 2015). The study highlights that while RDBMSs can be a good performance with a limited amount of data, MongoDB is a lot faster. This was experimented by creating a database for a forum. MongoDB was on par or faster than MySQL in all operations. MongoDB is also a lot more flexible because it does not has adhere to the static structure that RDBMSs rely upon. The study acknowledges that RDBMSs have their use in the future and while transition from an RDBMS to MongoDB can be challenging, there´s nothing that prevents using them at the same time. MongdoDB is optimized to store large amounts of data and still provide good performance. (Györödi;Györödi;Pecherle;& Olah, 2015)

A similarly themed study was done in the same year (Kumar;Rajawat;& Joshki, 2015). In the study the performance and ease of use of MongoDB was rated a lot higher than MySQL. The study also promotes using MySQL and MongoDB side by side. (Kumar;Rajawat;& Joshki, 2015)

Greca et al. showed that the best parts of MongoDB and Elasticsearch can be combined by using them together (Greca;Kosta;& Maxhelaku, 2018). According to the study "MongoDB is a great general-purpose database, but one place where limitations show up is full text search. … Elasticsearch is the best engine for on filtering and searching text." (Greca;Kosta;& Maxhelaku, 2018). In the study MongoDB was used for storing the data and Elasticsearch for searching data. The conclusion was that combination is perfect for storing great amounts and/or complex data and making fast searches.

Shah et al. state that MongoDB is not fast enough to handle big real time data. Solution was to use Elasticsearch which gave better performance (Shah;Mago;& Willick, 2018). The article also presents a couple of real-life cases. For example an RDBMS based health care system lacked required integration functionalities and was too slow to perform even near-real-time queries. Solution was to use Hadoop (NoSQL) for the needs of raw data and Elasticsearch for real-time analysis. In another example Elasticsearch was taken into use alongside existing MySQL. A potential issue was raised about the data size, as the data is duplicated, but the system is flexible. (Shah;Mago;& Willick, 2018)

3. SPEECH TO TEXT

Like Seppo Enarvi mentions in his doctoral dissertation, speech recognition is a complex machine learning task (Enarvi, 2018). The exact inner workings of speech to text solutions and algorithms are not in the focus of the thesis, but Enarvi´s work can be recommended for those interested. It explains the evolution from early digit and vowel recoqnizers to more modern HHMs (hidden Markov models), neural network HHM hybrid solutions (Enarvi, 2018). In essence converting speech to text is a classification task where the search space is very large. This is made more complex with the unknown quantities of audio like alignment of text and acoustic signal quality. Automatic speech recognition has already worked well for planned English language from the turn of the century. Early adopters were for example medical personnel doing dictations (Enarvi, 2018). In the past a medical doctor would do store his dictation by using a dictation machine, which stored the speech into a storage media like a tape. An assistant would then listen through the tape and write down the speech with a computer. This process makes for a perfect example where automated transcription either produced immediate cost savings or freed work force for other more meaningful tasks. At that time though automated transcription was not very flexible as it required a well-defined domain, in this case the domain being medical. Early on the systems were even speaker dependent: the system had to trained to understand a specific speaker (Enarvi, 2018). They also required well-articulated, non-colloquial speech, something that persons often do naturally if they know they are talking to a machine. Systems then would not have been able to transcribe spontaneous conversations by random persons, something that they are certainly capable of now.

Challenges, however, persist with less-common languages and with colloquial speech which often differs greatly from standard language (Enarvi, 2018). The research on speech-to-text has concentrated on English language and other languages have far less resources. The smaller the speaker base of a language is, the less research resources it will get. The more complex a language is, the more resources it would require. Finnish language is problematic in both ways, having only 5,5 million speakers and generally

considered to be one of the hardest languages to learn. In addition to this colloquial Finnish substantially differs from the standard language. Even the grammar and pronunciation of words can be quite different (Enarvi, 2018). The premise for successfully transcribing Finnish seems challenging. META-NET (Multilingual Europe Technology Alliance) tracks the status of European languages and especially the status of computer services for languages. As seen in Table 1 the support for Finnish ranged between moderate to poor/no support, but more importantly speech processing was listed as moderate.

Table 1. Status and support of Finnish language (META-NET, 2019)

Machine translation	Weak / no support
Speech processing	Moderate support
Text analysis	Fragmentary support
Speech and text resources	Fragmentary support

3.1. Speech-to-text services

There are already many of speech-to-text services available, both from industry giants (Google, Microsoft, IBM, Amazon) and from smaller companies. Most of them do not support Finnish language. The transcription quality of these services with Finnish language is an unknown factor. The same uncertainty is also with audio files of less-than-optimal quality, in this case phone calls. There seemed to be no studies on how the transcribing solutions work with either of these. Internet contains many comparisons, but many of them seem to be biased, being published by a transcription service provider to promote their own product over others. Scientific publications on the matter could not be found. This means that services selected have by trialed and the quality of the transcripts analyzed.

According to their service documentation, Microsoft and Google both offer transcription services for Finnish. The two were in top places in Nordic APIs´ "5 best speech to text APIs" blog post (Simpson, 2019). The post also praises Google for proper noun

recognition and noise cancellation, both important factors when transcribing phone call recordings. In Këpuska`s and Bohouta´s comparison done in 2017 Google was deemed superior and Microsoft also fared better than CMU Sphinx-4. Matarneh et al. also regarded Google highly (Matarneh;Maksymova;Lyashenko;& Belova, 2017). In another study Google was used for transcribing Romanian educational videos, and in best cases WER rates were as low as 10% (Iancu, 2019). As Romanian language is even less supported than Finnish, it can be presumed that Google´s performance would be even better with Finnish language. With these articles as a base and with a presumption that a well-known company is more likely to offer secure, functional and robust services, Google and Microsoft were selected to be used in trials. Because of the studies, and the fact that Google is a known performer with Finnish, as many of its services already contain speech recognition that works with Finnish, Google was selected to be the main focus of this thesis. Google is to be compared to Microsoft´s implementation.

3.2. Measuring speech-to-text error rate

Enarvi presents terms that are commonly used for evaluating speech-to-text quality: WER (word error rate), LER (letter error rate) and CER (character error rate) (Enarvi, 2018). WER is the standard measure for assessing speech recognition accuracy, however for agglutinative languages it can be too inaccurate. Finnish researches often use LER instead of WER. LER penalizes more for completely incorrect words. In Finnish language the change of a few characters can not only cause the inflection to change, but it can also change the meaning of the entire word as seen in the following example.

Näitkö sitä? = Did you see it?
Naitko sitä? = Did you have sex with it?

Do note that in conversational Finnish the word *sitä*, a partitive case of *se*, the exact English translation is *it.* In Finnish *it* can refer to anything, be it a person or an object. WER is calculated by summing the number of word substitutions (*S*), the number of

word deletions (*D*) and the number of word insertions (*I*) needed to correct the result, and dividing these by the total amount of words (*W*) (Enarvi, 2018).

$$WER = \frac{S + D + I}{W}$$

WER shall be used in this thesis when the transcription accuracy is analyzed in chapter 6.

4. GOOGLE SPEECH-TO-TEXT API

Several companies already provide APIs for speech-to-text conversion. For this project Google Speech-to-Text API was chosen as the primary service to be investigated because it is widely used and well proven. For example YouTube and Google Maps, both of which presumably use the same API, can recognize difficult Finnish street names and voice spoken by a child. All information regarding Google API presented in this chapter was retrieved from Google Speech-to-Text API documentation web pages (Google, 2019).

Google API offers different transcription options for different kinds of audio: synchronous for ~1 min recordings, streaming for ~5 min recordings and asynchronous for maximum of 480 min recordings (Google, 2019). In this project only the asynchronous option was used as phone calls often last over 5 minutes, so Google Cloud Speech-to-Text REST API v1`s command called *longrunningrecognize* was used.

Pricing of the API usage is 0,006 USD for every starting 15 seconds of audio to be transcribed, so transcribing 1 hour of audio would cost 1,44 USD (Google, 2019). Google API has some limits in place dictating how rapidly one can send requests as seen in Table 2. The daily limit is slightly strange considering that transcribing audio costs money, producing income for Google. The limits were reached while doing testing with a mass of recordings. Requesting a quota increase is possible though.

Table 2. Google API usage limits. (Google, 2019)

Type of limit	Usage limit
Requests per 100 seconds	500
Processing per 100 seconds	10 800 seconds of audio
Processing per day	480 hours of audio

4.1. Using the API

Using the Google API is relatively straight forward, since API documentation is quite comprehensive and intelligible. The process starts with creating a Google Cloud Platform account and acquiring an API key. Transcribing call recordings in a non-streaming way requires the user to upload recordings into Google Cloud Storage (GCS). This can be an issue for example with clients who are worried about information security requirements like GDPR. Phone calls are user sensitive data and uploading them outside EU might be a costly mistake. Fortunately GCS allows one to determine storage location for the account, so files can be kept inside EU area, allowing to meet GDPR regulations. It was chosen to transcode the recordings into wav format since this way we can omit some of the parameters from the API requests like encoding and sample rate, as Google API is able to figure this out by itself.

Transcription operation is started by issuing a POST request like one shown in Listing 1. The parameters shown here are the bare minimum required for a successful transcription, but the API offers a plethora of other options.

Listing 1. Google API transcription request.

```
POST https://speech.googleapis.com/v1/speech:longrunningrecognize?key=<API key>
Body in JSON: {
  “config”: {
    “languageCode”: “en-US”
  },
  “audio”: {
    “uri”: <GCS URI to a recording file>
  }
}
```

The API then responds in JSON including a parameter called “name”. This is a unique operation id that we´ll need for fetching the transcription results. To fetch the transcription results we´ll issue a request as in Listing 2:

Listing 2. Google API fetching transcription results request.

```
GET https://speech.googleapis.com/v1/operations/<operation id>?key=<API KEY>
```

This API responds in JSON. If the operation is still ongoing, the key “done” will be false and key “progressPercent” will be empty or have a value less than 100. If the operation has been completed, the response will look like the one shown in Listing 3.

Listing 3. Google API response of a successful and finished transcription.

```
{
    "name": "5279452236348712299",
    "metadata": {
        "@type":
"type.googleapis.com/google.cloud.speech.v1.LongRunningRecognizeMetadata",
        "progressPercent": 100,
        "startTime": "2018-11-08T12:13:06.552149Z",
        "lastUpdateTime": "2018-11-08T12:20:51.949774Z"
    },
    "done": true,
    "response": {
        "@type":
"type.googleapis.com/google.cloud.speech.v1.LongRunningRecognizeResponse",
        "results": [
            {
                "alternatives": [
                    {
                        "transcript": "tekninen tuki ",
                        "confidence": 0.77321124
                    }
                ]
            },
```

```
            {
                "alternatives": [
                    {
                        "transcript": " meillä on yks tommonen käyttämättä jäänyt aiemmin
Käytössä ollut puhelinnumero niin",
                        "confidence": 0.83118135
                    }
<response continues containing full transcript splitted into “alternatives”>
```

The implementation that uses the API would pick all the different parts of dialog splitted into “alternatives” and for example combine them into a single piece text, and this would be our transcript. The API was found to be easy to use and no major hindrances were met during the trial.

4.2. Performance

Audio files used for testing were actual phone calls stored in mono mp3 format. Sample frequency was 8 kHz, which is typical for phone calls, and bitrate was 32 kb/s. Transcribing a single file is somewhat slow as seen in Table 3.

Table 3. Google API single file processing times.

Audio file length	Processing time	Processing time per one minute of audio
2s	3,5s	105s
2min 6s	41s	20s
18min 49s	7min 45s	25s

However the API allows parallel processing, something that can be used as a workaround if the amount of files to transcribed is high. A separate implementation was created to allow testing with a mass of recordings. However as shown in Table 2 Google has limits in place that prevent the users of the API from pushing audio files as rapidly as possible. A rudimentary request limiter was implemented to ensure that none of the limits are exceed. The result was that transcribing 500 phone calls having total length of 51

hours were processed in 1 hour 30 minutes, which equals average process time of 1,8s per one minute of audio. The cost of the operation was roughly 80 USD, unit cost being 0,006 USD for every starting 15 seconds, so transcribing audio certainly is not free.

4.3. Quality

During the testing it was found that while the overall quality of the transcripts was acceptable the transcript quality varies a lot. As predicted English language had more accurate transcript results than Finnish, but even the English ones were not flawless. Often there were some words missing or they misinterpreted. It was found that audio quality plays a huge difference in the results. Even though the phone call recordings have the same nominal quality, like compression and sample frequency, it was often the case that audio quality of one of the speakers was a lot poorer than the other one. With the recordings it was often the case that the other party was using a computer with VoIP and a proper headset for communication and the other one was just using cell phone, and the cell phone user would have a lot worse audio quality. This can be caused by bad reception and service provider setup like codecs and bandwidth, and to lesser extent the properties of the phone itself. Another technical issue that leads into issues is the difference in gain levels. In some cases the audio level of other speaker was considerably lower than the other one, and in some rare cases both were low. This leads to dialog missing completely in the transcripts and in worst cases there are only a couple of sentences or even words in the transcript.

Then there are human issues like articulation, pronunciation and dialects, which cause poor transcript results even if the technical quality of the recording is good. In difficult cases these often result in misinterpreted words. When a human knows that they are speaking to a machine, they will automatically articulate to the best of their abilities, making the speech easier for computer to decipher, but this does not happen with human-to-human dialog. Proper nouns like company names are another hard-to-cipher factor. The API requests were given company names as keywords (parameter "phrases") but this did not improve the results. The previously mentioned issues are even worse

with Finnish language, leading to misinterpreted words more often than with English language. Mixing languages makes it even worse and this happens a lot with company names which are often in English. While testing with a specific company name it was properly interpreted in 1/5 of the cases. While the technical quality of the recordings can be improved by the user itself, there´s not much what can be done with human related issues as these can only be solved by the transcript service provider. The only solution is to test different service providers or wait for improvements in quality.

Steps were taken to improve the recording quality by storing the audio as dual mono. This means that one speaker is on one channel and the other one is on other channel. Using recordings like this requires the user to define "audioChannelCount" and "enableSeparateRecognitionPerChannel" parameters in the API requests. However this did not improve the transcript quality. It most likely would in cases where speakers are talking over each other, and it would help in separating different speakers (aka speaker diarization), but with calm mannered dialog it showed no improvement.

Whether the quality is good enough depends on the end user and the purpose of use. If the use is only about looking for certain keywords from the content, then even the less successful transcripts can still be useful. For actual comparison of transcription results, see chapter 0 for transcription results.

5. MICROSOFT AZURE SPEECH TO TEXT API

Microsoft Azure Speech to Text API was used for comparison purposes to see if the transcribing speed or quality would be any different to Google. Microsoft offers a Speech-to-Text SDK and a REST API. In this project we will solely concentrate on the REST API. However the standard REST API has a maximum recording length of 10 seconds which is way too short for our purposes (Microsoft, 2019). Fortunately Microsoft offers separate dedicated REST API for this called Batch Transcription API. This is said to be ideal for call centers, capable of transcribing large volumes of audio recordings (Microsoft, 2019). Microsoft API also supports setting up web hooks, something that was not found from the Google API. This is extremely beneficial as it allows the API to notify the user when a transcribing task has been finished instead of the user having to poll the API. The pricing model for Azure is can be seen in Table 4.

Table 4. Azure pricing for standard web/container feature set. (Microsoft, 2019)

Type	Price
Standard	1 USD / 1 h
Custom speech	1,40 USD / 1 h
Custom Speech endpoint hosting	40 USD per month

Microsoft does not elaborate the difference between “standard” and “custom speech” features, but during the testing phase “Custom Speech endpoints” from Speech Services API v2.0 will be used, so it was assumed feature set is “custom speech” and thus the price is 1,40 USD per hour (Microsoft, 2019). Usage is billed in one second increments. The price is very close to Google´s 1,44 USD per hour. The API has a limit of a maximum of 20 concurrent requests.

5.1. Using the API

Taking Azure API into use is not as straightforward as it was with Google. Then main problem is with the documentation of batch transcript API. A good thing is that there is

a “getting started” type of guide that takes to a point where one should be able to use the standard speech-to-text API, but as mentioned before, standard API has the 10 second limit, so this example was useful only to the point where one is able to get an access token.

Using speech to text API starts with creating a new Cognitive Services resource in Azure portal. Location of the service was selected to be North Europe. After the resource is created, one is given a subscription key and an API endpoint. With this key we can request an access token from the API. Since batch transcription requires the audio files to reside in Azure storage blobs, a storage account had to be created as well. The test recordings were then uploaded into a blob, much like what was done with Google GCS. The storage account was set to use North European region, which should be enough to meet GDPR regulations.

After getting the subscription key an access token can be requested from the API. Now we were ready to start using the batch transcript API and this is when the problems started. Microsoft´s batch transcription API “how-to” page was quite poor as the only examples it provided were related to the request body (Microsoft, 2019). There were no examples of which REST API resources to use, in other words after reading this the developer would not know how to formulate the request URL nor headers. From another webpage a link to ”REST API: Batch transcription and customization” Swagger documentation was found (Microsoft, 2019). Swagger is commonly used web tool that allows users to define API documentation with a markup language. However Microsoft´s Swagger documentation has nothing mentioned about batch transcription, so it was assumed that POST /api/speechtotext/v2.0/transcriptions is the resource that should be used, since the request body looks the same as the example on batch transcription API documentation. Lack of proper request examples lead to guessing, and this resulted in time wasted on trial and error.

The Swagger page itself was nice as it allows using the API usage from the web page, showing templates for request parameters and headers and so on (Microsoft, 2019). It

also allows user to enter subscription key or access token and then remembers them, so the user can concentrate on making actual API requests. One of the most helpful examples it gives are complete cURL examples including headers and parameters, allowing the developer to copy these to their implementation. This allowed us to start issuing requests to the API and observing what kind of cURL requests it made. Transcript request is issued like shown in Listing 4.

Listing 4. Microsoft API transcription request.

```
POST https://northeurope.cris.ai/api/speechtotext/v2.0/transcriptions
Header: accept: application/json
Header: Ocp-Apim-Subscription-Key: <API key>
Header: Content-Type: application/json
Body in json:
{
  "recordingsUrl":    "https://dippastorage.blob.core.windows.net/dippa-blob/brooklyn.wav
?sp=r&st=2019-08-01T11:13:22Z&se=2019-08-01T19:13:22Z&spr=https&sv=2018-03-
28&sig=ZHIUBT0ekBuoiIH3Wuf%2BX7YmdX946ilcNjTCXnu123%3D&sr=b ",
  "models": [],
  "locale": "en-US",
  "name": "Transcription using locale en-US",
  "description": "An optional description of the transcription.",
  "properties": {
    "ProfanityFilterMode": "Masked",
    "PunctuationMode": "DictatedAndAutomatic"
  }
}
```

The value set for parameter recordingsUrl is a URL of a file stored in “dippastorage” Azure blob. The URL includes a shared access signature (SAS), something that is needed when using files from Azure blob. The URL was generated in the Azure portal´s resource management tool. The request returns data in JSON. In it there´s a transcription operation id, in this case “3fee72c8-dfb8-4bce-88f4-19e7eb1c0b61“. For getting the transcription result we´ll issue a request like in Listing 5.

Listing 5. Microsoft API fetching transcription results request.

```
GET  https://northeurope.cris.ai/api/speechtotext/v2.0/transcriptions/3fee72c8-dfb8-4bce-
88f4-19e7eb1c0b61
Header: accept: application/json
Header: Ocp-Apim-Subscription-Key: <API key>
```

This request will return a json that can contain information of a possible transcription failure, like URI not found, or if the operation is ongoing. If the transcription was successfully finished, we get a JSON containing a link to the actual transcript, as shown in Listing 6.

Listing 6. Part of Microsoft API response of a successful and finished transcription.

```
{
  ...
    "channel_0":   "https://spsvcprodneu.blob.core.windows.net/bestor-d22af23b-e7fb-4bb2-
9280-3ed688b07456/TranscriptionData/2a6958d4-a8c5-49fc-9eb0-661e281d4123.json?sv=2017-04-
17&sr=b&sig=2rxuLsGus6A1HJAIeoKwUjyCvwb6fi1GRnhpHVw123AE%3D&st=2019-08-
01T12:23:53Z&se=2019-08-04T12:28:53Z&sp=rl"
  },
  ...
```

With this link we can get the actual transcript of the audio file, giving us as response like in Listing 7.

Listing 7. Microsoft API transcript.


```
{
  "AudioFileResults": [
    {
      "AudioFileName": "Channel.0.wav",
      "AudioFileUrl": null,
      "AudioLengthInSeconds": 1.79,
      "LexicalFull": "how old is the brooklyn bridge",
      "ITNFull": "how old is the Brooklyn bridge",
      "MaskedITNFull": "how old is the Brooklyn bridge",
      "DisplayFull": "How old is the Brooklyn bridge?",
      "SegmentResults": [
        {
          "RecognitionStatus": "Success",
          "ChannelNumber": null,
          "SpeakerId": null,
          "Offset": 400000,
          "Duration": 17500000,
          "OffsetInSeconds": 0.04,
          "DurationInSeconds": 1.75,
          "NBest": [
            {
              "Confidence": 0.9707699,
              "Lexical": "how old is the brooklyn bridge",
              "ITN": "how old is the Brooklyn bridge",
              "MaskedITN": "how old is the Brooklyn bridge",
              "Display": "How old is the Brooklyn bridge?",
              "Sentiment": null,
              "Words": null
            }
          ]
        }
      ]
    }
  ]
}
```

5.1.1. Criticism regarding the API and documentation

The trial was plagued by issues with API documentation. Either the information that was needed to make progress was not directly available as a link or the documentation was not detailed enough or there not good enough examples.

Finding the Swagger documentation was essential in testing this API, but it was not linked or referred to in the “How-to” guides. When the link that was found redirects to Western US site. API key from Northern Europe could not be used in there. The host name for Northern Europe region, northeurope.cris.ai, had to found first and then the Swagger API documentation page reopened on that particular host.

Here´s an example on the lack of detail in specification. The documentations state the following about the recording url parameter regarding starting a transcript operation:

- Swagger: `"recordingsUrl": "https://contoso.com/mystoragelocation"`
- How-to: `"recordingsUrl": "<URL to the Azure blob to transcribe>"`

Neither one accurately explains that what is really needed is an URL to a file inside an Azure blob, and that the URL must contain SAS.

When using the API with Finnish locale “fi-FI” the API returned an error 400 with a message that locale “fi-FI” is not supported, even though it is clearly listed on the Speech-to-Text supported locales. This will make Microsoft´s API unusable for some Finnish customers.

I found Microsoft´s documentation a lot harder to understand that Google´s and the usage itself was a bit harder as well. Especially the issues documentation caused a lot of time wasted on trial and error and just trying to understand the logic behind the services and API. In general Google´s API was easier to understand and it was faster to take into use.

5.2. Performance

I retrieved performance figures by comparing createdDateTime to lastActionDateTime, which are available when retrieving transcript results. Table 4 shows that the speed seems to be on par with Google, with averaging 25-30 seconds for one minute length recordings. Mass transcriptions could not be measured as time and budget prevented adding the Azure support to our own service and running the transcriptions en masse.

Table 5. Microsoft Azure API single file processing times.

Audio file length	Processing time	Processing time per one minute of audio
2s	21s	630s
1min 2s	30s	29s
18min 49s	8min 50s	28s

5.3. Quality

As it was with Google, also Microsoft´s API has issues with proper nouns. However the quality of English transcripts is remarkably good. This was even to a point that the API was able decipher data from a recording where a real person could not. The quality of Finnish transcripts cannot be commented as the API does not seem to support Finnish language, giving errors if Finnish locale is used. The actual transcription results are presented in chapter 0.

6. TRANSCRIPTION RESULTS

All tests results listed here were done using 8 kHz mono 32 kb/s mp3 audio files. All of them were actual phone call recordings where one line is using a cell phone and one line is using a VoIP connection. The English recordings documented in this chapter were one of the worst in technical quality that were used during trials. The Finnish recordings were of above average quality for a phone call. WER rates are calculated for every transcript, see chapter 3 to see how it is calculated.

English recording where the other speaker has a very bad line and the other one is ok. The audio level with the problematic line was ok but the voice itself somewhat unclear. Good enough for a person to understand but most likely problematic for a computer. The speaker on the bad line also had a notable accent, being non-native English speaker. Actual audio:

> *Hello. Hello. I need some help um we have um account. It´s called demo. Yeah. Yeah. Could you at some point give me the password. I don't have any don't have access for it. 2004 is what you are looking for.*

Google result:

> *hello Google I need some help 18 door but 2004 is what you're looking for*

Microsoft result:

> *hello hello I need some help we have everything there an account it's called IMO yeah yeah could you at some point give me the password I don't have any don't have access for it 2004 is what you're looking for*

Google´s result is a failure. “I need some help” was the only thing that the API was able to get out from the person who was on a bad line. The person who was on a good line was transcribed just fine. However Microsoft´s result was very good indeed. The only thing that failed was transcribing “demo” into “IMO”. WER rates are 69,2% for Google and 12,8% for Microsoft. The difference is huge in Microsoft´s favor.

Another English recording where the audio level of the GSM caller was really low and speech was unclear. Over 90% of speech come from the bad line. Again the speaker on the bad line is non-native English speaker.

Actual audio:

> *Hi. Hi. Could you do me favor on account erhm demo. Yeah. Erhm I believe we cannot use the chat in office because <some unrecoqnizable speech> in an hour. Uh*

Google result:

> *I follow could you do me a favor on the counter*

Microsoft result:

> *hi could you give me a favor and account I think it's important apparently we cannot use the chat in the office because you also have it in an hour uh*

Some parts of this recording were challenging even for a real person to understand. The voice quality of the caller was subpar. Google´s result is a failure as the service was unable pick most of the content, resulting in severe loss content. However the result provided by Microsoft was really good, almost better than a real person. WER rates are 79,2% for Google and 20,8% for Microsoft. Like in the previous case, the difference is huge in Microsoft´s favor.

A Finnish recording which worked surprisingly well, just a few words a missing or wrongly transcribed. Audio quality was fine on both ends.

Actual audio:

> *Tekninen tuki. No <name> täällä terve. Terve. Hei meillä olis yhdellä sellainen ongelma, että linja pätkii pahasti. Okei. Ei meinaa niinku puheluista tulla mittää ja hän soittaa eri tota maasta. Hän on niin kuin fyysisesti <place>, niin voiko siinä olla jotain vaikutusta tähän että. Hyvin todennäköisest joo, et katotaas vähän miten noi on säädetty tualla.*

Google result:

> *tukisanalista terve terve Hei Meillä on semmonen ongelma että ja pätkii pahasti meillä niinku puhelusta tulla mitään ja hän soittaa eri tota maasta hän niin Kivistön <place> vai voiko siinä olevan vaikutusta tähän että hyvin todennäköisesti Joo katotaan vähän miten noi on säädetty tuolla*

A surprisingly good result. Company and person names were wrongly interpreted. Also some other words are missing or wrongly interpreted, but I´d would still call this a successful transcript. No results for Microsoft for as it could not be made to work with Finnish recordings. WER rate for Google is 32,1%.

A Finnish recording where the author personally did his best to mimic Finnish dialects and way of speech, simulating a case where the caller speaks standard Finnish and the call receiver speaks in a dialect in a fairly quick pace.
Actual audio:

> *No se on Kiviharjun Timppa terve. Kuule luetteko Aku Ankkaa? Nyt olisi timanttinen tarjous. No enmä kyllä tollasta roskaa alkaa tilaamaa. Onko teillä Erä-lehteä tai jotain muuta kunnon lukemista.*

Google result:

> *No se on Kiviharjun Timppa terve Kuule Luetteko Aku Ankkaa nyt olisi timanttinen tarjous lähinnä kala tommosta roskaa alkaa tilava Onko teillä erälehto jotain muuta kuin lukemista.*

The one speaking in standard Finnish was transcribed just fine, including proper nouns as well. However the one speaking in dialect was less successfully transcribed. No results for Microsoft for as it could not be made to work with Finnish recordings. WER rate for Google is 23,3%.

6.1. Conclusion

Generally speaking the transcribing of English language works well, especially with Microsoft´s API. The biggest difference between the two services was the sensitivity to low recording quality. While Microsoft was having WER of 13-21%, Google had 70-80%. The difference is staggering. With Finnish recordings Google fared better, even though the language support for Finnish cannot be as good as it is for English. This resulted in WER between 23-32%. The improved result is most likely caused better recording quality. Another common problem in both English and Finnish transcripts was that proper nouns like names of companies and persons were not correctly transcribed.

For Google it can be said that transcribing works well if the recording quality is high enough and speakers articulate well. It seems to be especially sensitive to recording quality. Fast pace of speak and strong dialects will also decrease the results, but even in these cases the results might still be usable. The most damaging effect comes bad quality audio files, resulting in wrongly transcribed words, and in worst cases, a massive loss of content.

Microsoft´s ability to dig information from even the poorest of recordings is simply amazing. The biggest shortcoming for it is the lack of support for Finnish, made worse with contradicting API documentation.

7. PREPARING DATA FOR COMPUTER ANALYSIS

After having successfully transcribed phone calls there´s going to be a hefty amount of textual data available. In order for the data be usable in searches or computer analysis it has to be thought how the data should be stored. In this chapter it is investigated how the data can be utilized and what options are there for storing it. There are several books and scientific publications that cover the fields text mining, natural language processing and machine learning, but they do not seem to focus on the data itself. However there are many articles and blog posts in the web that do cover this problem field. Based on the literature and examples the storage options can be divided into two: text files and databases. These will be investigated in sub-chapters, comparing their advantages and disadvantages.

7.1. How to utilize transcripts

Some might use the data for simple searches with a goal just to find certain keywords from the corpus. Typical use case would be to identify swearing. Keyword searches can be seen as a part of a field known as information retrieval, where typical solutions are systems that provide full text search or content-based indexing. Performance of single keyword searches can vary greatly between systems. Even different database management systems can have notable performance differences. A term that often appears with information retrieval is text mining. Aggarwal states that “The area of text mining is closely related to that of information retrieval, although the latter topic focuses on the database management issues rather than the mining issues.” (Aggarwal, 2018).

More elaborate ways to utilize the data are text mining and natural language processing methods (NLP). Depending on source the definition of the two term are either separate or very much intertwined. Bieman & Mehler say that “Text Mining (TM), a variant of Data Mining (DM) on text data, is an important discipline of Natural Language Processing (NLP)” (Biemann & Mehler, 2014). Kao & Potect describe the differences between the two by stating that “Text mining is the discovery and extraction of interesting,

non-trivial knowledge from free or unstructured text. This encompasses everything from in-formation retrieval (i.e., document or web site retrieval) to text classification and clustering, to (somewhat more recently) entity, relation, and event extraction. Natural language processing (NLP), is the attempt to extract a fuller meaning representation from free text. This can be put roughly as figuring out who did what to whom, when, where, how and why." (Kao & Potect, 2007). In other words text mining is about extracting relevant information from text, as NLP is about extracting a fuller semantic meaning.

Typical uses for text mining example are for example web search engines, spam filters, and recommender systems that offer the user items of possible items of interest. There are several approaches to this. The most common one is bag-of-words model. In it the order of the words have meaning. Typical approach to bag-of-words is to TF (term frequency) in which the count of each word or term is calculated. An evolution of this is TF-IDF (term frequency – inverse document frequency), intended for finding out the importance of terms in a document. An adaptation of bag-of-words is using term sequences instead of words, which is a step towards NLP. Note that some sources list bag-of-words as part of information retrieval and NLP, showing how hard it is to differentiate the related terms. (Aggarwal, 2018)

7.1.1. Preprocessing transcripts

In order to be able to get good results from text mining, the text must undergo preprocessing. Some of the more common steps are text extraction, stop-word removal, stemming. Test extraction is about identifying primary block from other blocks that contain unrelated content like advertisements. Doing this will need special techniques. Stop word removal is cleaning the text from words of little meaning like common pronouns and prepositions. Typical examples of this are *the, is, at* and so on. Problems may arise if one is for example trying search for the band "The Who". Word length can also be as a basis of removal. The benefit of this is improved mining process. Both text mining and NLP utilize machine learning, NLP methods being more complex. The difference

between the two is that text mining usually has document collections as source data and uses statistical indicators like frequency of words and word patterns. (Aggarwal, 2018)

Stemming means returning words, for example verbs, back into their basic form aka common root (Aggarwal, 2018). In most sources the term *stemming* includes everything from simple and crude methods to sophisticated solutions that include dictionaries and morphological analysis, but Manning et al. separate the two. They describe stemming being usually a crude process that removes the end of words. While this might work in most cases, it will sometimes fail to do the job properly. Lemmatization is the more advanced way of doing stemming, using vocabulary and morphological analysis to return the word to its base form (Manning;Raghavan;& Schütze, 2009). An example of this are words *sinking* and *sank*, which should result in word *sink* after lemmatization. All these operations are often highly language specific, again something to consider when processing Finnish language.

7.2. Text files

The beauty of text files is that they are universally compatible. Writing text into a file is the simplest form of storage, making for example migration an easy task. In practice all platforms and implementations can read them. Downsides are that one has come up with a way how to manage the data on disk, so that the desired data set can even be found. Also combining data from different files would be difficult as well. As a workaround one can always combine technologies, like having data in a database, making it easily customizable and combinable, and then exporting it as csv for a machine learning solution. This is investigated in chapter 7.3.

Keyword search is as one of the possible use cases mentioned in chapter 7.1. It can be done with text files but the search performance will be no match to databases. This simply because of the lack of indexing when searching from a file system. Database solutions like Elasticsearch and MongoDB, or even relational database like MySQL, will always perform better. The two are investigated in chapters 7.4 and 7.5.

While investigating examples about machine learning it seemed that almost all of them used text files as source data, often csv files. Csv files, short for comma separated values, are essentially tables in text format where values are separated from each other by delimiters like semicolons. An Excel table can be easily converted into a csv file. Using table-like data when running a training set into machine learning solution is beneficial, as one can have the corpus and the verdict in the same file. Verdict can be for example a boolean or numerical value describing whether a review was positive or not, or whether customer service was able to solve the problem at hand.

In Javaid Nabis´ article the goal was to be able to make a sentimental analysis on movie reviews. The source data consisted of 25 000 highly polarized iMDB (Internet movie database) movie reviews to be used for training a classifier and another 25 000 for testing the classifier. It was pre-known which reviews are positive and which are negative. The reviews were in text files, a single review in a single file. The files when then loaded into a Python dataframe and saved in csv format further use using three columns: index, review text, sentiment (0 for negative, 1 for positive). The corpus was then cleaned by removing words shorter than three characters and stop words like “the”, “this” and so on. Then the corpus was pre-processed into feature vectors by using term frequency vectorizer. Then 12 500 positive and 12 500 negative reviews were used for training a machine learning solution, in this case a basic Naïve Bayes classifier, resulting in a functional “Sentiment Analyzer”. A test set consisting of 25 000 reviews was run through the analyzer, resulting in 79% accuracy. This example clearly shows that textual files can be successfully used as source data. (Nabi, 2018)

7.3. Relational database management systems

Two of the most used database systems in the world, Oracle and MySQL, are relational databases (DB-Engines, 2019). While they might not be the best choices for storing great amount documents or for having the fastest search speed and the best analytics tools, the popularity of them mandates the investigation what they have to offer for

searches and text analysis. Companies might already have existing an RDBM up and running, so it would make sense to investigate how the existing infrastructure could be utilized.

Using a relational database for storing the data instead of text files brings many benefits. Arguably the most important benefit text analyzing wise is the ability to manipulate the data. It can be filtered, combined, modified and formatted. However a common issue with some RDBMSs is that the keyword search performance can be poor, and is highly dependent on how indexes have been formed and what kind of table structures are used. Also horizontal scalability not as easy as with some other databases like MongoDB. With increased amounts of data, like storing documents, the performance does suffer and is no match for modern NoSQL solutions (Ribeiro;Henrique;Ribeiro;& Neto, 2017). If a need for performance improvements should rise, an RDBM can use alongside other solutions. Elasticsearch could be used for taking search speed into an entirely new level (Greca;Kosta;& Maxhelaku, 2018). MongoDB could be used for increased document write and search speeds (Györödi;Györödi;Pecherle;& Olah, 2015) (Kumar;Rajawat;& Joshki, 2015). This kind of approach would allow the user to use the best parts of each database system.

7.3.1. Keyword search with traditional indexes

Let´s take MySQL as example, since it is the second most used RDBMS in the world, and the first if we take only open-source free-to-use solutions into account (DB-Engines, 2019). In MySQL any non-indexed column will have poor search performance. Solution is either to create indexes or use the full text search functionality. Creating an index allows fast searches from the start or the end of the column value, but not both nor from the middle. Table 6 shows an example how much time it will take to search to search matches from a MySQL database table. The column is of type VARCHAR(100) and has been indexed. The “without index” test was done with wild cards on both sides of the keyword. The search not utilizing the index takes 35 times longer than the one that is using the index.

Table 6. MySQL keyword search performance from 5 000 000 rows

Without index	5,2s
With index	0,15s

Indexes will not be used if keyword is search from middle of the text, or both from the beginning and the end. For example searching from the start of the string (LIKE "<keyword>%") or from the end (LIKE "%<keyword>") would perform just fine, this will not be the case when wildcards surround the keyword (LIKE "%<keyword>%"). Doing a search like this would cause the query to disregard the indexes entirely. The solution for this is full text search, which is also supported by MySQL nowadays. This is investigated in next chapter.

7.3.2. MySQL with full text search

The general concepts of MySQL were already covered in chapter. What was not done was to investigate the full text search capability. Similar to defining indexes for a column, one can define a full text index for a column. After this the column can be used with full text search functions, something that would not work with normal indexes. In Listing 8 FULLTEXT index is creating comprising of columns title and body.

Listing 8. Creating FULLTEXT index in MySQL (MySQL, 2019)

```
CREATE TABLE articles (
    id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
    title VARCHAR(200),
    body TEXT,
    FULLTEXT (title,body)
) ENGINE=InnoDB;
```

Full text search functions are split into natural language full text searches and boolean searches. Boolean mode allows user to define and weight words that are desired and words that must be absent. Natural language mode interprets the search string as phrase in human language and returns results in relevance order. An example of a natural lan-

guage type query is shown in Listing 10. The query Both modes use stopword removal. This is built-in to MySQL but by default supports only English. Custom stopword lists can be added though. (MySQL, 2019)

Listing 9. MySQL query using FULLTEXT index (MySQL, 2019)

```
SELECT COUNT(*)
FROM articles
WHERE
    MATCH (title,body)
    AGAINST ('database' IN NATURAL LANGUAGE MODE);
```

One of the best things that MySQL with full text search brings is the ability to combine traditional RDBMS data management and filtering to that of free text search. The problem is that this might not always work. Even with standard indexes MySQL has sometimes difficulties in choosing which indexes to use and in which order. The way how MySQL choses to use indexes, or chooses not to use, is difficult to predict. This can lead to surprises and be devastating to query performance, where for example a query that filters it´s data by using indexed columns only might suddenly take overly long time to run.

To demonstrate the nuances of MySQL, this time purely within full text search context, let´s get back to Listing 9. The query can also be written like shown in Listing 10. The results will be the same, but the performance won't. According to MySQL reference manual the latter query should be quicker. However the first query can be faster if the search matches only few rows. The latter will be faster if there are hits on many rows.

Listing 10. MySQL query using FULLTEXT index (MySQL, 2019)

```
SELECT
    COUNT(IF(MATCH (title,body)
        AGAINST ('database' IN NATURAL LANGUAGE MODE), 1, NULL))
        AS count
FROM articles;
```

The problems with searches gets much worse when combining standard and full text indexes in filtering. Grilly gives an example where combining standard indexed column filtering with full text search increases query time from 50ms to 50s (Grilly, 2017). That is a 1000 time increase in query time and shows how MySQL can have problems merging indexes. There may be workarounds for this though. Backus recognizes the same problem and uses temporary tables as a workaround (Backus, 2016).

While the full text search in MySQL does work, it does not contain the built-in functionalities that Elasticsearch has. Also combining the indexes might cause surprising performance issues to which there might be workarounds, but then the usage will become more complicated.

7.3.3. Benefits of being able to manage and manipulate the data

Other ways of utilizing databases is data storage and manipulation, and especially the ability to combine different datasets into a single set. Just by altering database queries, like joining tables, and grouping and filtering the data it is possible to easily come up with the desired set of test data. An article by Candido Zuriaga gives a good example about the difficulties of using csv files and the solution is to use a database, in this case MySQL. The goal was to build an analyzer that would predict whether the certain type of restaurant would be committing to an inspection violation. Source data was in three csv files: list of businesses, inspections and violations. Considering data relations that a business can have multiple inspections and an inspection can have multiple violations. Combining the three csv files into something that would ready for computing, usually a single file, is difficult, and hence the data was imported into a MySQL database. After

import the data can be easily transformed. For example there can be missing data or numerical codes that need to be transformed into human readable format. After the data in the database is ready, it can be combined into a single set of results by using joins in an SQL query, and exported in a single text file. This would then be something that could be used in machine learning, in other words MySQL database was used for transforming and combining the data. Zuriaga then uploaded his result file into BigMLer which is a command line tool for machine learning. (Zuriaga, 2013)

In another blog post by Nishant Upadhyay a similar thing was done with MySQL, but the data then was used a bit differently. Jupyter notebook was used for gathering the data as it can interact with MySQL database and then data the was converted into Pandas (a Python data analysis library) data frames to be used in further analysis. The thing to note in both examples is that MySQL was used for storing and manipulating the data, but later it was exported into text files to be used in computer analysis. (Upadhyay, 2018)

7.4. Elasticsearch

Elasticsearch is an open-source, distributed, scalable, real-time search and analytics engine. DB-Engines classifies Elasticsearch as a search engine and document store. It the most popular search engine and 7th most popular database engine overall (DB-Engines, 2019). According to Tong & Gormley “It exists because raw data sitting on a hard drive is just not useful.” (Gormley & Tong, 2015). For example Wikipedia uses Elasticsearch to provide full-text search functionalities.

Elasticsearch is stated to be very good for indexing and searching big datasets and excels in finding records matching some search criteria (Vimal, 2017). The emphasis is on search and read performance, however write performance is considered to be slow. Because Elasticsearch is not an RDBMS, the commands and functions like joins and subqueries are not available (Konnyu, 2018). This is something to consider if one plans to switch from an RDBMS like MySQL. Kuzzle goes as far as stating that Elasticsearch

should be used purely as a search engine, not as a database (Kuzzle, 2017). A good example why is the fact that Elasticsearch has had some issue with resiliency which can result in data problems in some cases. The company has a resiliency status webpage that explains the current status (Elasticsearch, 2019). Elasticsearch is often used alongside other DBMSs, so that for example MySQL RDBM is the persistent storage providing robustness and Elasticsearch is only used for making searches (Vimal, 2017). While MongoDB offers fast search speeds, it is not match for Elasticsearch, but these two can also be used together (Greca;Kosta;& Maxhelaku, 2018). A nice thing about Elasticsearch is that features a REST API. Unlike MongoDB, in which the interfaces are limited to certain programming languages, Elasticsearch gives it full compatibility to all environments.

While Elasticsearch looks like an excellent solution for searching with keywords, what has it to offer for NLP? Elasticsearch has some built-in functionalities that use NLP methods. One of them is “more like this” query. An example of this is shown in Listing 11. User selects a set of terms or documents and passes them to Elasticsearch, and the database then returns a set of documents that it thinks are closest to the given parameters. The this is accomplished is picking terms with highest TF-IDF and then forming a disjunctive query. This is something that could be used for example in digital shopping for recommending other products.

Listing 11. Elasticsearch "More like this" query (Elasticsearch, 2019)

```
GET /_search
{
    "query": {
        "more_like_this" : {
            "fields" : ["title", "description"],
            "like" : "Once upon a time",
            "min_term_freq" : 1,
            "max_query_terms" : 12
        }
    }
}
```

How about text analyzing tools and preprocessing that was mentioned in chapter 7.1.1? Elasticsearch contains a set of language analyzers that include stopword recognizing, stemming and lemmatization. The list of supported languages includes Finnish. In other words for data preprocessing and traditional NLP approaches like bag-of-words Elasticsearch has everything built in. (Elasticsearch, 2019)

7.5. MongoDB

MongoDB is an open-source document-oriented NoSQL database. According to DB-Engine it is the most popular solution for document store databases and 5th most popular database engine overall (DB-Engines, 2019). Unlike MySQL, which uses strictly defined tables and rows, MongoDB works with documents and collections. It is an unstructured database and promotes horizontal scalability. It does not attempt to do the work of an RDBMS and lacks for example transactions and joins. This means that it is not suited for every task, like accounting for example, but there´s nothing that would prevent using MongoDB with an RDBMS together. Dropping the transactions is trade-off that allows MongoDB to be simpler, faster and more scalable (Hows;Membrey;& Plugge, 2015). In a forum-like performance study MongoDB was on at least on par with MySQL in all operations, and considerable faster in most cases (Györödi;Györödi;Pecherle;& Olah, 2015). While using Elasticsearch as a persistent

database is strongly inadvisable, MongoDB can used as a persistent database if it fits the use case. It is more resilient than Elasticsearch.

Unlike Elasticsearch, which can be accessed via REST API, MongoDB interface is limited to certain programming languages (Hows;Membrey;& Plugge, 2015). This can be seen as a downside, because if the desired language is not the supported list, then one has to resort into 3rd party libraries or drop MongoDB entirely. On positive side, MongoDB is supposedly very to use. Listing 12 contains a text search example which orders the results according to score how well each document matches the specified query.

Listing 12. MongoDB text search example

```
db.stores.find(
   { $text: { $search: "java coffee shop" } },
   { score: { $meta: "textScore" } }
).sort( { score: { $meta: "textScore" } } )
```

According to Kuzzle full-text search is possible, but it is doesn't support functionalities like 'more documents like this' which would allow to find other related documents more easily (Kuzzle, 2017). Performance-wise it seems to be no match for Elasticsearch as Kaminskyy describes MongoDB to have low performance and high resource using full-text search, and simple recommends using Elasticsearch for searches (Kaminskyy, 2018). MongoDB is not a search engine but then again nothing prevents using MongoDB with one. For real-time or near-real-time searches the speed might not enough good enough (Shah;Mago;& Willick, 2018). In a study by Greca et al. MongoDB was used with Elasticsearch to get the best parts from both solutions (Greca;Kosta;& Maxhelaku, 2018).

Nicolas Png gives a good example on how to utilize MongoDB with machine learning like TF-IDF (Png, 2018). Notable is that MongoDB was used as data storage and for providing data for models. While MongoDB does not have all the built-in NLP functionalities that of Elasticsearch, it does offer stopword filtering, simple suffix stemming,

and both of the functionalities support Finnish (MongoDB, 2019). As a highly scalable platform providing flexible data model, indexing and reasonably highspeed querying it does work.

7.6. Conclusion

For document storage and keyword search in mind MongoDB, or most likely any other similar NoSQL solution, seems to be the perfect solution. It offers considerable faster read, write and search operations than any RDBMS, and can be easily scaled horizontally. It does not have the search speed nor the advanced indexing and NLP functionalities of Elasticsearch. However these two technologies can be combined to get the best out of both.

For companies having existing RDBMS infrastructure, for example MySQL can be made to support full text search. If more search performance or advanced features are required, it too can be used alongside Elasticsearch. Also MongoDB can be used as a document storage alongside MySQL.

8. CONCLUSION

Transcribing services have matured to the point where automated transcribing of conversational audio is feasible if not 100% accurate. This opens entirely new possibilities how audible data can be used. These may range from simple keyword searches, like identifying cursing, to analyzing the calls with text mining, NLP and machine learning. Companies could use the data to find out specific faults and occurrence rates with their products, or optimizing their customer service process and so on.

The speech-to-text services are not perfect though. While transcribing English generally performs well, there are many variables that affect the quality of the transcripts and can lead to unsatisfactory results. Trials showed that the services available are not equal. Recording quality can play a huge role in transcribing. Different transcribing services react differently to issues with this. It is made worse by a caller having a bad line, or strong dialects and poor articulation. The most surprising part of testing was how differently the two services performed under these circumstances. While Microsoft´s service coped surprisingly well with these issues, Google struggled to provide results. In worst cases the transcripts were missing most of the content.

All the effects of these issues are made worse if the language in question is complex and rare. Finnish language is both. Google does support Finnish but the results vary, even more so that with English language. Unfortunately Microsoft´s service proved to have no support for Finnish language at all, so the best performer in English had to be left out from Finnish trials.

After the transcripts have been made available, there are multiple ways how to store and use the data. Most of the text mining and machine learning examples just utilize data in raw text format such as csv files. However managing such data is hard as a person or implementation has to maintain the data on disk. Also search performance would be poor. Alternatives to this are databases, which vary from traditional RDBMS like MySQL to search engines and document storages. Studies show that modern document

storages, like NoSQL-based MongoDB, can outperform MySQL by wide margin. Promoting speed (both search and write), ease of use and horizontal expandability, MongoDB seems like an excellent choice for storing transcripts. If even more search speed is required, or advanced indexing and NLP features, Elasticsearch is an excellent option.

None of the database implementations investigated in this thesis is a do-it-all solution: they all have their own distinctive pros and cons. Either the end user picks one and accepts the missing or less-optimal features as tradeoffs, or one combines two or more technologies to work side-by-side. If one was already using MySQL, they might want to take the search speed to a new level and have Elasticsearch run on the side. Someone might want to combine easy-to-use document storage like MongoDB to something that offers even faster searches and built-in text mining and NLP functionalities, again like Elasticsearch. The amount of choices is vast and it falls to the end user to pick the solution that fits their needs the best.

REFERENCES

Aggarwal, C. (2018). *Machine learning for text.* Switzerland: Springer.

Backus, K. (2016). *Using Full-Text Index For InnoDB When a Search Engine is not Feasible.* Retrieved 09 01, 2019, from https://medium.com/@kirkbackus/using-full-text-index-for-innodb-when-a-search-engine-is-not-feasible-d666830b4000

Besacier, L., Barnard, E., Karpov, A., & Schultz, T. (2014). Automatic Speech Recognition for Under-Resourced Languages: A survey. *Speech Communication*.

Biemann, C., & Mehler, A. (2014). *Text mining: From ontology learning to automated text processing applications.* Switzerland: Springer.

DB-Engines. (2019). *DB-Engines ranking.* Retrieved 08 31, 2019, from https://db-engines.com

Elasticsearch. (2019). *Elasticsearch Resiliency Status.* Retrieved 08 31, 2019, from https://www.elastic.co/guide/en/elasticsearch/resiliency/current/index.html

Elasticsearch. (2019). *The Definitive Guide: Theory behind Relevance Scoring.* Retrieved 08 03, 2019, from https://www.elastic.co/guide/en/elasticsearch/guide/current/scoring-theory.html

Enarvi, S. (2018). *Modeling conversational Finnish for automatic speech recoqnition.* Helsinki: Aalto university.

Google. (2019). *Cloud Speech-to-Text documentation.* Retrieved 08 18, 2019, from https://cloud.google.com/speech-to-text/docs/

Gormley, C., & Tong, Z. (2015). *Elasticsearch: The Definitive Guide.* Sebastopol, USA: O´Reilly.

Greca, S., Kosta, A., & Maxhelaku, S. (2018). Optimizing data retrieval by using MongoDb with Elasticsearch. *Recent Trends and Applications in Computer Science and Information Technology. 2280.* CEUR-WS.org.

Grilly, N. (2017). *Don't Waste Your Time With MySQL Full-Text Search.* Retrieved 09 01, 2019, from https://hackernoon.com/dont-waste-your-time-with-mysql-full-text-search-61f644a54dfa

Györödi, C., Györödi, R., Pecherle, G., & Olah, A. (2015). A Comparative Study: MongoDB vs. MySQL. *The 13th International Conference on Engineering of Modern Electric Systems, Oradea.*

Hows, D., Membrey, P., & Plugge, E. (2015). *The definite guide to MongoDB* (3rd ed.). Apress.

Iancu, B. (2019). Evaluating Google Speech-to-Text API's Performance. *Informatica Economică*.

Kaminskyy, A. (2018). *Moving from MongoDB Full-Text Search to ElasticSearch (for Node.Js Applications).* Retrieved 09 02, 2019, from http://bitcom.systems/blog/moving-mongo-to-elasticsearch/

Kao, A., & Potect, S. (2007). *Natural language processing and text mining.* London: Springer.

Këpuska, V., & Bohouta, G. (2017). Comparing Speech Recognition Systems (Microsoft API, Google API and CMU Sphinx). *International Journal of Engineering Research and Applications*. Retrieved from https://pdfs.semanticscholar.org/2e7e/bdd353c1de9e47fdd1cf0fce61bd33d87103.pdf

Konnyu, J. (2018). *MongoDB vs. Elasticsearch.* Bitninja. Retrieved 08 31, 2019, from https://bitninja.io/blog/2018/02/23/mongodb-vs-elasticsearch-by-bitninja

Kumar, L., Rajawat, S., & Joshki, K. (2015). Comparative analysis of NoSQL (MongoDB) with MySQL Database. *International Journal of Modern Trends in Engineering and Research*.

Kuzzle. (2017). *What NoSQL solution should you choose?* Retrieved 08 31, 2019, from https://blog.kuzzle.io/what-nosql-solution-should-you-choose-mongodb-elastisearch-oriendb

Lardinois, F. (2019). *https://techcrunch.com/2019/07/23/google-updates-its-speech-tech-for-contact-centers/.* TechCrunch. Retrieved 09 22, 2019, from https://techcrunch.com/2019/07/23/google-updates-its-speech-tech-for-contact-centers/

Leskovec, J., Rajaraman, A., & Ullman, J. (2014). *Mining of massive datasets.* Cambridge: Cambridge University Press.

Manning, C., Raghavan, P., & Schütze, H. (2009). *Introduction to information retrieval.* Cambridge University Press.

Matarneh, R., Maksymova, S., Lyashenko, V., & Belova, N. (2017). Speech Recognition Systems: A Comparative Review. *IOSR Journal of Computer Engineering, 19*(5).

META-NET. (2019). *Key Results and Cross-Language Comparison.* Retrieved 09 22, 2019, from http://www.meta-net.eu/whitepapers/key-results-and-cross-language-comparison

Microsoft. (2019). *REST API: Batch transcription and customization (Swagger documentation).* Retrieved 08 03, 2019, from https://westus.cris.ai/swagger/ui/index

Microsoft. (2019). *Speech service regions.* Retrieved 08 03, 2019, from https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/regions

Microsoft. (2019). *Speech services documentation.* Retrieved 08 03, 2019, from https://docs.microsoft.com/fi-fi/azure/cognitive-services/speech-service/rest-speech-to-text

MongoDB. (2019). *The MongoDB 4.2 Manual.* Retrieved 9 29, 2019, from https://docs.mongodb.com/manual/

MySQL. (2019). *MySQL 8.0 reference manual.* Retrieved 09 01, 2019, from https://dev.mysql.com/doc/refman/8.0/en/

Nabi, J. (2018). *Machine Learning — Text Processing.* Towards Data Science. Retrieved 08 23, 2019, from https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958

Perkin, N. (2017). *Data-Informed vs Data-Driven.* Business Agility. Retrieved 09 22, 2019, from https://agilebusinessmanifesto.com/agilebusiness/data-informed-vs-data-driven/

Png, N. (2018). *Training Machine Learning Models with MongoDB.* Retrieved 09 02, 2019, from https://www.mongodb.com/blog/post/training-machine-learning-models-with-mongodb

Ribeiro, J., Henrique, J., Ribeiro, R., & Neto, R. (2017). NoSQL vs Relational Database: A Comparative Study About the Generation of Most Frequent N-Grams. *4th International Conference on Systems and Informatics, At Hangzhou, China.*

Shah, N., Mago, V., & Willick, D. (2018). A framework for social media data analytics using Elasticsearch and Kibana. *Wireless networks*.

Simpson, J. (2019). *5 Best Speech-to-Text APIs.* Nordic APIs. Retrieved 8 17, 2019, from https://nordicapis.com/5-best-speech-to-text-apis/

Singh, H. (2018). *Using Analytics for Better Decision-Making.* Retrieved from https://towardsdatascience.com/using-analytics-for-better-decision-making-ce4f92c4a025

Upadhyay, N. (2018). *Artificial Intelligence series_part 5: Data Analysis using SQL.* Medium. Retrieved 08 23, 2019, from https://medium.com/datadriveninvestor/artificial-intelligence-series-part-5-data-analysis-using-sql-7e61bee24b85

Vimal, R. (2017). *elasticsearch vs mongodb.* Medium. Retrieved 08 31, 2019, from https://medium.com/@ranjeetvimal/elasticsearch-vs-mongodb-631f410cd317

Vivek, S. (2018). *Analyzing Customer reviews using text mining to predict their behaviour.* Medium.com. Retrieved 09 22, 2019, from https://medium.com/analytics-vidhya/customer-review-analytics-using-text-mining-cd1e17d6ee4e

Zuriaga, C. (2013). *Data Preparation for Machine Learning using MySQL.* BigML. Retrieved from https://blog.bigml.com/2013/10/30/data-preparation-for-machine-learning-using-mysql/