FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR’S THESIS

Title:	Offline call center analysis
Student:	Boris Pazdera
Supervisor:	Ing. et Ing. Martin Švík Ph.D.
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of summer semester 2019/20

Instructions

The bachelor thesis should focus on analysis of the current state of the art of methods of call center deflection and to design and implement a solution that will automatically transcribe audio calls, analyze content and show the results of the analysis (annotations, sentiment,...) of the call.
Steps:
1) Research the existing methods.
2) Analyze transcription solutions and select one of the best technology for transcription of call center calls.
3) Analyze text analysis solutions and choose the most suitable technology for the implementation.
4) Design architecture of the final solution.
5) Implement functional prototype of the proposed architecture of the solution.
6) Test implemented solution.
Details of the project will be clarified by consultation with the supervisor.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 21, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Offline call center analysis

Boris Pazdera

Supervisor: Ing. et Ing. Martin Švík, Ph.D.

May 15, 2019

Acknowledgements

First and foremost I thank my supervisor, Martin Švík. Without his belief, I would never take the courage to write a thesis in English. I thank you very much for your long term lead and challenge.

I would like to also express my sincere gratitude to Martin Bacher for the proofreading of the whole thesis.

I am truly grateful to Mr. Maroš Benka from Phonexia company for guidance on their software.

I also place on record my gratitude to all who directly or indirectly have lent their hand in this venture.

Declaration

In Prague on May 15, 2019 . . . .

Citation of this thesis

Pazdera, Boris. *Offline call center analysis.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

V této práci jsou nastíněny základní možnosti zpracování zvukových nahrávek z call center a jejich hloubkové analýzy z pohledu jak textové, tak zvukové. Toho je dosaženo návrhem architektury, výběrem správných komponent a realizací prototypu.

Klíčová slova call centrum, analýza zvukových nahrávek, analýza textu, vytěžování dat

Abstract

This thesis outlines are outlined the basic possibilities of processing audio recordings from call centers and their in-depth analysis from the perspective of both text and audio. This is achieved by designing of the architecture, choosing the right components and implementing the prototype.

Keywords call center, audio analysis, text analysis, text mining, data mining, voice mining

Contents

List of Figures

Listings

Introduction

Each of us is trying to get as much information as possible before making any decision. We will never have enough of it, as it is impossible and unsustainable to capture 100 % of all data. In addition, it is very unlikely to analyze them all due to the volume of the information.

However, with the development of Information technologies, the possibilities are wider and constantly expanding. This is mainly due to the massive modernization of computer hardware. Previously, we were really limited since we had only one option and that was "just" human reading and evaluation by our own cognitive functions.

Now, we have more possibilities due to contemporary technologies, where we can process, analyze and automatically aggregate data based on many parameters without any human work. This kind of data processing is also easily scalable. In the case of data growth in the past, we had to hire, train and pay new people who would perform "manual" analysis. Today, we can extend our systems and/or rent more computer power and process more data with a minimal financial cost. Especially in this period when we have historically the lowest unemployment rate of 2.4 % (2018) since 1991 when there was 2.3 % [1]. We can let people evaluate already cleaned, grouped data in a form that is easy to read and with no need to read the documents one by one.

We are in a situation when data is one of the most important commodities to help us make any decision. We are trying to collect even more of them to get to the imaginary "golden grail" in the form of all available knowledge. No wonder, because decision-making mechanisms based on the hard data are the best way in the implementation of any new product, in problem detection of already running service or in the fast removal of an existing issue.

We collect a lot of data daily with anticipation of future use, without the current tools for evaluation and analysis. All companies process not only their own sensor data or data from other systems, but additional data collected from client channels. They usually process data from e-mail communication and chat tools, where it is predominantly text analysis. Most of the companies have call centers as well, where they generate a lot of data in different formats (text, call frequency, voice, . . .), but this analysis is much more complicated.

Call centers offer enormous potential for the information extraction, unfortunately, we usually use only metadata (length, number of calls, . . .) and not the content itself. Here is a huge open space and potential to collect very important information and that is the main reason why I chose data extraction from voice recordings in call centers topic as my Bachelor's Thesis. The whole work is based on obtaining data from voice recordings in the call center and enabling the possibility of their analysis mainly from the point of the view of the recording's content.

The main benefit of the thesis will be the ability to analyze the content of the voice recordings deeply.

Goals

The goal of this thesis is to design and implement the prototype of software solution which will provide an opportunity for data analysts to analyze voice recordings from the text content perspective. This will be accomplished by extracting the basic information from voice recordings (length, number of channels, frequency, language, . . .), conversion to the text and then making basic text mining (sentiment expression extraction, entity extraction, . . .).

Data analysts need to have the option to make a full-text search through the content of all recordings and to be able to find the document as a search result with a basic overview of metadata extracted from the voice recording. The data should be analyzable by a human in GUI via the web browser.

Specific goals

To perform research of existing systems for audio and text processing which would be suitable for the on-premise solution.

To find out possibilities of data extraction from the audio tracks point of view, eventually possibilities of entity extraction and making annotations in text content.

Choose the right third-party components or algorithms that will secure the individual parts of the solution with an emphasis on the selected system requirements.

The solution will work with personal data about customers and therefore it is necessary to propose a solution taking into consideration this fact and the overall relation to GDPR.

The final part is the implementation of the GUI itself, in which the results will be displayed and evaluation of the whole implementation.

Chapter 1

State-of-the-art

Each call center keeps voice tracks of their calls, but in most cases, they use only the metadata of those recordings (length, number of calls, occupancy, etc.), or to report problems with some operator (agent). These recordings may be sometimes used also for supervision by the manager or a different case studies. However, the content of individual calls is used only very sporadically, or companies are only reliant on the notes of individual agents that they write during the call.

This is the place to be used to get more information and conduct an in-depth analysis of the call content. However, there is currently no solution that can help with the proper complex analysis of audio call recordings in which both voice and text information can be combined and analyzed using methods for both types of analysis.

1.1 Real-time analysis

In general, some data are extracted directly during a call with the customer, usually, it is information that can directly assist the agent in a call. It is therefore primarily an analysis of sentiment or emotion. This data are very important for a better client experience, where an agent can propose a different solution to a given problem, offer a product/service, or use an "upsell" technique and that all based on emotions that were extracted from recording. Among other things, it will increase the satisfaction of individual clients, the degree of commitment and thus improve the brand in the customer's mind [2]. Nevertheless, in real-time analysis, it is not possible to do a deeper survey into the data (data extraction via multiple annotations, comparison with previous recordings, . . .), which will not allow us to fully use them.

1.2 Content analysis

Data analysis is possible at several levels. Primarily, data can be split and analyzed in two separate parts (audio and text). From both, different data can be extracted, or the same data, but in different ways, such as language via language identification.

1.2.1 Audio recording analysis

Voice recording analysis is possible based on voice amplitude, where recordings can be classified by height, color, sound intensity and much more.

The sound classification is based on a pre-trained model. The accuracy depends on the quality of the model. In our case, it is best to use a language model that is trained directly for the needs of call centers, to get the best accuracy we should use data coming directly from the call center because each recording is distorted several times. The sound is transmitted through mobile phones, where GSM [3] compression is usually performed. Furthermore, the sound is transformed on the side of the call center itself.

From the voice recordings point of view, for the Speech to Text (STT) conversion could be considered several solutions for fulfilling the goals of the bachelor thesis, among which are SpeechTech [4], NEWTON Technologies [5] and Phonexia [6]. We will look into the details in section 2.1.

1.2.2 Text analysis

There are currently a large number of products and algorithms on the market that can perform basic analysis and search in text form. The use of algorithms over textual data is much simpler in most cases. Especially if you need to extract some additional information from the text, such as different entities, where you can create an annotation based on certain REGEX rules. For textual analysis we will consider Solr [7], Elasticsearch [8] and IBM Watson Explorer oneWEX [9], more in section 2.2.

Chapter 2

Analysis and design

Currently, we find many different data solutions (either audio or text) on the market. However, there is no one that can present both audio and text representation of data to an analyst for work.

It is not necessary for data to be processed and visualized in real time, which is almost impossible. The solution must be designed so that data can be uploaded periodically (daily, weekly, . . .) for processing. In addition, the result of the data mining must be visible through a web interface that is intended for authorized users only. At this point, data search and analysis above the textual representation options must be enabled for the user.

2.1 Audio analysis

In this section, we will go through individual systems from section 1.2.1 for voice analysis to explore their features, pros, and cons as the research of the functions which we can add to the whole solution.

2.1.1 SpeechTech

SpeechTech is one of the Czech companies, which are providing voice analysis. Their software is used by many companies, including the Czech Television for making real-time subtitles for deaf people [10].

Their main focus is on the call analysis and conversion of speech to text and vice versa. They provide the whole solution for voice analysis, but they can also provide the basic modules separately. Their technologies are usually used in many domains, but generally, the healthcare segment benefits from these features the most. Speech to text helps them to easily create health records or record business minutes from meetings. [10]

2.1.1.1 Speech to Text

SpeechTech provides STT service in their Cloud, but it could be problematic to GDPR[1] in Europe as the voice data will be sent to other systems. On the other hand, they also provide an option to have the whole solution on your MRCP[2] server which could be hosted on the company's network and you can customize the transferring data to/from services. MRCP would help you also with the connection to your IVR. [11]

2.1.1.2 Voice analysis

For the voice recording, SpeechTech provides a basic categorization feature of the call and semantic analysis, which will allow looking on the recordings based on category. Another function is Dialog analysis, this could also help the agent's manager to analyze the effectivity of an agent's work and behavior. This system helps to get an overview of the customer and their behavior (i.e. abusive language detection, ...). [12]

2.1.1.3 Conclusion

Although the SpeechTech service could be possibly used for our use case, their language support is very limited and for our purpose, it is not enough. Also, they focus mainly on speech analytics directly from the call center/microphone, in opposite to our use case, where we have a WAV (WAVE) file as an input.

2.1.2 NEWTON Technologies

NEWTON Technologies was founded in 2008 and their main focus is on voice recognition especially for the Slavic languages. They provide services for the Speech to Text conversion and basic analytics around the voice tracks. They provide the services on their cloud and/or as the on-premise solution[3] which allows wider usage (because of the customer data processing). [13]

2.1.2.1 NEWTON Dictate

One of their technologies is NEWTON Dictate, which opens the abilities to transform voice to text (speech to text). Their software is prepared for the dictation of voice and transferring to the written form. They propose the solution suitable mainly for the fields of state administration, justice, or healthcare, where is not unique to have longer texts. [14]

[1]General Data Protection Regulation
[2]Media Resource Control Protocol
[3]Solution placed inside the enterprise environment

2.1.2.2 NEWTON Analytics

NEWTON Analytics has a powerful tool for text processing, which provides the ability to recognize the call topics and keywords or full-text search capabilities in recordings, but the solution is not easily customizable. [15]

2.1.2.3 Conclusion

Although NEWTON technologies have a great solution for the audio recordings, there are many things for the nowadays needs as a machine-learned annotation, which the solution does not provide. Also, the language support is not enough for our purposes [16].

2.1.3 Phonexia

The Phonexia engine is great for recordings from a call center because their models are trained on the call center's recordings and their solution is primarily designed for this field. "Phonexia specializes in telephony speech - landline, GSM [4], VOIP[5]" [17]. Among other things, it offers the opportunity to apply model training to match the exact needs of the call center because they can have different transformations and sound compressions.

2.1.3.1 Language identification

Phonexia's Language detection (LID) feature is based on iVectors [18] technology, which is completely independent of the text, language, dialect or audio recording channels. Identification is primarily designed to recognize and classify any recording based on pre-trained models. This technology combines the Gaussian Mixture Model (GMM) [19] and iVectors to generate small but highly representative voice prints.

To obtain representative samples, you need to have a voice recording of at least 9 seconds, which recordings meet reliably because the average call duration across all types of call centers is 5.97 minutes [20]. Each of the languages being tested for a given probability will have a certain likelihood of matching where the use of formula $e^{score} \cdot 100$ is required to obtain a percentage in the range 0 – 100 %. If we summed up all languages, we would get 100 %, thus the distribution of scores between all languages within the classification.

The advantage is that you can use the already pre-trained model that is included within the product (choice of 53 languages and dialects). Alternatively, it gives you the opportunity to train your own language based on your own samples, which can better meet our needs. This is the recommended practice in many cases where we have a call center for non-native speakers

[4]Global system for mobile communications – Mobile phone protocol

[5]Voice over Internet Protocol – delivery of voice communications and multimedia sessions over Internet Protocol

(e.g., English-speaking call center agents for many Asian companies, btw. Phonexia engine included the Indian English model). Another advantage is that the engine also supports the Czech language within its preset package.

Language identification technology gives us the possibility to distinguish them, but it is not recommended to use all 53 pre-trained languages together. It is much more effective to use identification by means of so-called language packs, i.e. to use really only the languages we expect in the recordings. We have all supported languages[6] within the default package "default".

2.1.3.2 Technology of Phonexia's Speech transcription

The technology is divided into two parts, an acoustic and a language model. The language model characterizes how are the words in relation in sentences. On the other hand, the acoustic model describes pronunciation in the record. This technology is prepared for streaming audio and voice file processing. In our case, we do the whole recording conversion after the call is completed, so we do not need to process the audio streaming.

The technology extracts acoustic features out of speech recordings and uses the acoustic and language models (together with pronunciations for individual words) in a recognition network to compute hypotheses of words in the recording and to "decode" possible transcriptions. Based on the requested output types, one or more transcribed texts are returned, possibly with timestamps and confidence scores for individual variants. [17]

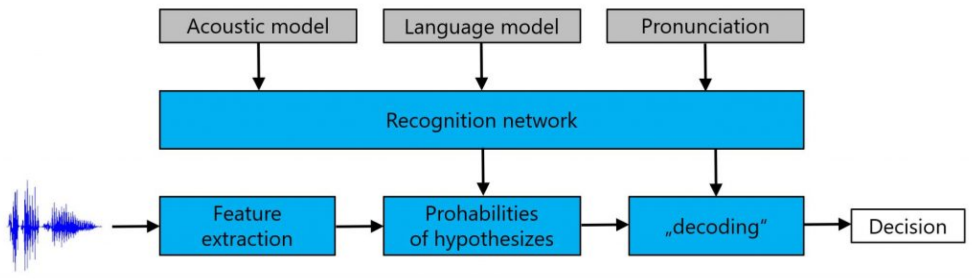


Figure 2.1: The Speech Transcription technology [17]

[6]Afan_Oromo, Albanian, Amharic, Arabic_Iraqi, Arabic_Levant, Arabic_MSA, Azerbaijani, Bangla_Bengali, Bosnian, Burmese, Chinese_Cantonese, Chinese_Mandarin Creole, Croatian, Czech, Dari, English_American, English_British, Farsi, French, Georgian, German, Greek, Hausa, Hebrew, Hindi, Indonesian, Italian, Japanese, Khmer, Kirundi_Kinyarwanda, Korean, Lao, Macedonian, Ndebele, Pashto, Polish, Portuguese, Russian, Serbian, Shona, Slovak, Somali, Spanish, Swahili, Tamil, Thai, Tibetan, Tigrigna, Turkish, Ukrainian, Urdu, Uzbek, Vietnamese

2.1.3.3 Acoustic model

For the acoustic models training, we need the voice recording and manually transcribed text. For this purpose, there is great public domain Mozilla's largest dataset [21] with human voices and appropriate texts which are suitable for the training of the acoustic model.

From the technical perspective, there is no difference between languages and dialects, but we can say that it is good to have the same model for people speaking in a similar way (British and American English).

2.1.3.4 Language model

This model characterizes the composition of words in recording. Basically, language models use similar procedures such as textual annotators. It classifies words based on their occurrences, sequences, n-grams [22] in a training data, in other words, it provides important information for the speech to text feature. It puts to the context how are the words in relation and how to interpret acoustic properties. This is the reason why we need a pretty huge training dataset. [17]

The accuracy of the language model depends on the training dataset and this is necessary for finding the most accurate transcription – think of similar-sounding words that can be distinguished by the context, such as "week" and "weak". The training set also limits the words which could be transcribed, because it is not possible to transcribe words that were not in the training data. [17]

From the 5th generation of the Phonexia engine is possible to use the Language Model Customization tool, which provides the possibility to add new words to the language model. [17]

To get the best results from recordings it is good to customize language models based on the domain you are working in. Most of the domains have different distances between the words, a combination of words, weights of n-grams [23] and many more. Different language models are possible to use with the same acoustic model without losing any accuracy on the acoustic model.

2.1.3.5 Speech transcription

The speech transcription feature is responsible for the conversion of speech signals to text. Then it allows us to work with text in the same way we are accustomed to. With plain text content, we will be able to make textual annotations, classifications, etc. This will open space for classic text-based data mining algorithms. In addition, Phonexia's Speech to Text is specifically developed for the need of converting speech to text in a busy and loud environment. [17]

Although we can use a voice recording for analyzes, for more depth entity extraction will be needed a text representation that we get by converting speech to text.

2.1.3.6 Conclusion

Phonexia is one of the best solutions on the market right now with the focus on the call centers and their engine is pre-trained for this type of input data. From the perspective of the methods, it supports all necessary features, which we will need in the solution and the engine is possible to run on-premise. Overall Phonexia meets all the requirements and we will use it for our prototype of the solution 3.

2.2 Full-text search and text content analysis

In this section we will look into the three candidates for text processing, all of them are based on the Lucene, but only two of them are open source software – Solr and Elasticsearch. Last is closed source software with long-term history but as the new product – IBM Watson Explorer oneWEX. I assumed, that in this case we will decide based on the security view, but I found that there is no disproportionate difference between open and closed source software [24].

2.2.1 Solr

Solr is a full-text search engine with a long history based on the Lucene [25] which was developed in 1999. The engine has a huge developer base and many extensions that could be added. The engine is widely used in many areas and because of his independence of the file format, it can process any file containing textual information. The biggest advantage is his capability to make fuzzy searches and that is the reason, why it is frequently used by website search engines. [26]

2.2.1.1 Conclusion

Solr is great for any solution based only on the full-text search capabilities, but it is not enough when we need to make deeper data extraction from the text content. This is the reason why it is not a good choice for our use case.

2.2.2 Elasticsearch

Elasticsearch is the younger brother of Solr. This engine was developed in 2010 base on Lucene too and right now it is the most popular search engine according to DB-Engines ranking [27]. It provides an option to run the search server on multiple servers simultaneously with web-based REST API. Moreover, it has over 20 built-in language analyzers and allows us to use many types of queries where also a similarity search or clustering and text classification is supported.

2.2.2.1 Conclusion

Elasticsearch is good for the long-term development of the whole solution, but the customizations would be hard to implement. The biggest disadvantage is, that for every change we will need developers and we are unable to effectively split the work between specialists (annotations, ...).

2.2.3 IBM Watson Explorer oneWEX

IBM Watson Explorer oneWEX ("oneWEX"/"IBM oneWEX") is an innovation in IBM's portfolio that specializes in unstructured text analysis. Although it offers many pre-trained annotators, it is also possible to use the ML[7] models. "IBM Watson® Explorer oneWEX components collect data from your enterprise; parse, analyze, and extract meaning from the information; and create a text index that users can query." [28]

At the same time, oneWEX also provides a presentation layer that will be analyzed by an analyst. The whole system provides a view on individual documents (voice recordings in text form) and a comprehensive view of the entire dataset. It can, therefore, perform deeper analysis or filter individual documents based on their own needs.

The whole product contains 4 main parts (Data Ingestion, Data Enrichment, Machine Learning, Exploration), no matter if we use Docker or IBM Cloud Private based solution. [28]

2.2.3.1 Data Ingestion

OneWEX is an end-to-end solution for textual data. It is able to crawl the data, analyze and show all the information and provides us multiple out-of-the-box crawlers, connections (web scraper, filesystem crawler, Box connector, . . .) and CSV importer for speed up of development. These crawlers and importers collect data from the source and after the parsing process, they store them into the index of the dataset. Most of the crawlers have the opportunity to schedule when it should run, which will ensure we have fresh data all the time.

[7]Machine learning

All crawlers and parsers are prepared for textual content. Although the filesystem crawler is highly customizable, the customizations are very limited and they cannot process the voice recordings. This is the huge disadvantage of these crawlers and the reason why we have to use another software for this purpose.

2.2.3.2 Data Enrichment

After the oneWEX obtains the data from the source and prepare them for further processing with the parsers, it starts with data enrichment for every document separately. It uses analytics pipeline in which extracts the data from each document, does linguistic analysis, finds meaningful words and phrases, extracts entities, and performs custom analysis. This step will provide facets that we can use for further exploration of content.

OneWEX uses LanguageWare as a NLP[8] technologies which were developed by IBM. "The LanguageWare libraries provide the following non-exhaustive list of features: dictionary look-up and fuzzy look-up, lexical analysis, language identification, spelling correction, hyphenation, normalization, part-of-speech disambiguation, syntactic parsing, semantic analysis, facts/entities extraction and relationship extraction" [29]

2.2.3.3 Machine Learning

AI[9] and ML are big topics in these days and in previous versions of WEX we did not have this type of functionality out-of-the-box (but it was able to use it inside the UIMA[10] [30] pipeline). Now the oneWEX obtains these features too and it helps this product on two different levels.

From the content perspective, we can use ML for the classifications or annotations of the content. Which provides a wider option of data extraction capabilities.

From the search perspective, oneWEX provides many different collaborative features, which could improve the search experience itself, but with the ML we can finally train the model based on the searches which people do in UI. E.g., every click on the search result is tracked and in addition people can rank the documents via the stars.

2.2.3.4 Exploration

The last part of the oneWEX is primarily for the end user (data analyst in our case). OneWEX allows to create custom UI for search applications in Builder or to use predefined GUI for the data inspection called Miner. Where we have

[8]Natural language processing
[9]Artificial Intelligence
[10]Unstructured Information Management applications

two modes – Expert and Guided. Guided mode is great for the simple data inspection, but if we would like to investigate some specific relations between the facets and data, we will have to use proper Expert mode. In this mode, we will have almost free hands and we can look into the tiniest parts of our data.

Moreover, in this part, we can influence search queries and the relevancy of single fields in our documents, add a whitelist/blacklist and synonym lists which is an important part of every search engine.

2.2.3.5 Architecture

"Watson^{TM} Explorer oneWEX uses cloud-based resource management based on container technologies." [28] and it is separated into two main components - Data Acquisition and Search/Analytics as you can see in figure 2.2.

The whole left-hand part (blue) describes obtaining the data from the source and storing them to the dataset. In this section, we can also add the injection of data from the REST API to the dataset, which is not shown in the figure 2.2.

The right-hand part (green) shows how is the text content proceeds when we create a collection from the dataset(s) and how is every document processed by the data enricher based on the predefined annotators, dictionaries, etc. Then we can see how exploration part from the Miner or the Search Application works.

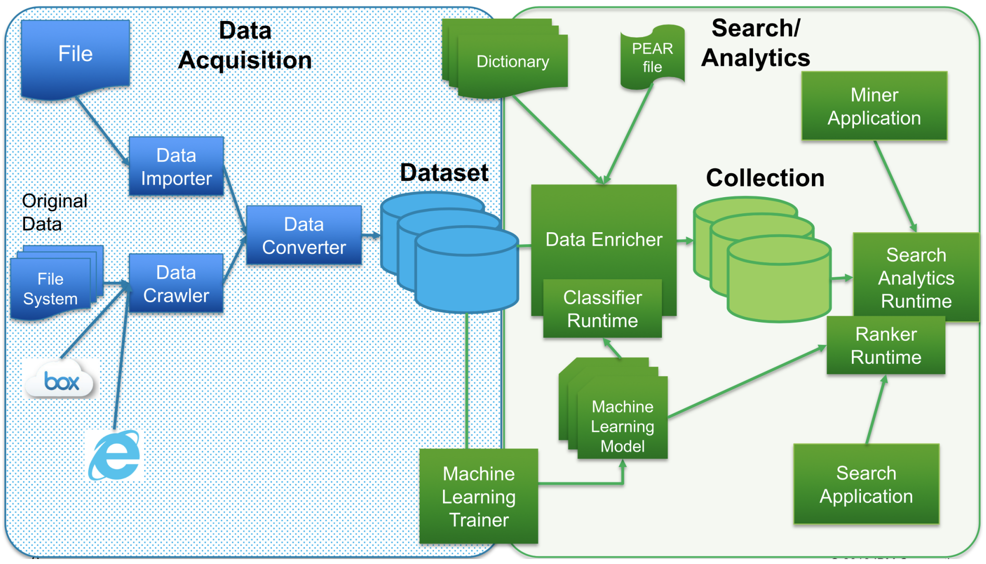


Figure 2.2: IBM Watson Explorer oneWEX Data processing scheme

2.3 Cogniware DataCollector

Although the oneWEX product itself has one of its components designed to crawl data from a variety of sources, such as filesystem, it is very limited and unusable for our needs. That's why we'll use Cogniware DataCollector, which gives us more crawl and data processing capabilities. It is a simple framework, where it is enough to prepare two Java libraries, that can then be linked together as a crawler. One of them is the "Connector" and the other "Data Handler" as you can see in figure 2.3. Both should be by definition general enough, to make it possible to crawl more filesystems or push data to the multiple destinations with the same parameters (i.e., to different databases of the same type, but in different destinations).

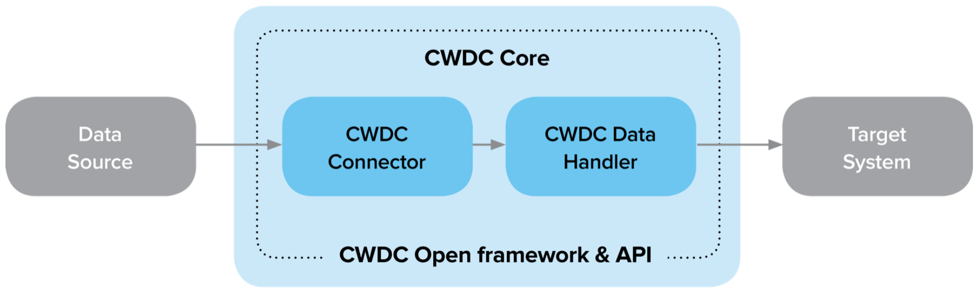


Figure 2.3: Cogniware DataCollector crawler architecture [31]

2.3.1 Connector

The Connector is used to retrieve data from various systems and data sources. It is possible to connect to various database systems, social networks, CRM (Customer relationship management) systems and basically any system with data and accessible via any type of connection (API, SSH, . . .) which is executable from Java library. In our case, the data are stored on the same server where the entire system is running. This means that it was necessary to create a filesystem connector that gets all the file paths that are then passed to the second part (the data handler), which takes care of further data processing.

2.3.2 Data Handler

Data handler takes care of all data processing which obtains from the connector. We are able to do anything that we can do in Java, there are no restrictions. It also supports multithreading, so we can run simultaneously multiple jobs. That is great especially for the CPU heavy job like working with voice data. In addition, we usually use Data Handler for the cleansing of the data which we receive, aggregate them to some type of model and then we send them to the target system.

Although several connections to different databases are prepared, and older IBM Watson Explorer Analytical Components connector is available too, the oneWEX is pretty new and the connection is not developed yet.

2.3.3 Crawler

Crawler in the meaning of DataCollector is a connection of Connector and Data Handler, which is abstract and we can change the data source and target system (destination) for data. All of this we do inside the DataCollector web UI. As we mentioned in section 2.3, both classes should be general and in the Crawler. We should specify which source we will crawl and to which destination we want to send the result. In other words, we can have the same Connector and Data Handler, which will do the same work, but for the different sources and destinations. We can also have multiple crawlers in the same environment and they are distinguished by the crawler ID.

2.4 Conclusion

My proposed software solution consists of three components. The first is the Phonexia engine, which can analyze the voice recording itself and ensure a quality conversion between voice and text representation using the Speech to Text feature.

The second part is the IBM oneWEX, which can retrieve information from text representation and take care of the visual part (GUI for data analytics) of the entire system. Nevertheless, these two systems are completely separated and there is no native connection between them in terms of integration.

This means that we need to use additional software that can perform data orchestration. In other words, getting raw voice data, make the first process with data and extract metadata from the recording via Phonexia, then send the data to oneWEX and let it analyze textual content. For this purpose, we will use Cogniware DataCollector (CWDC).

The final architecture you can see in the figure 2.4.

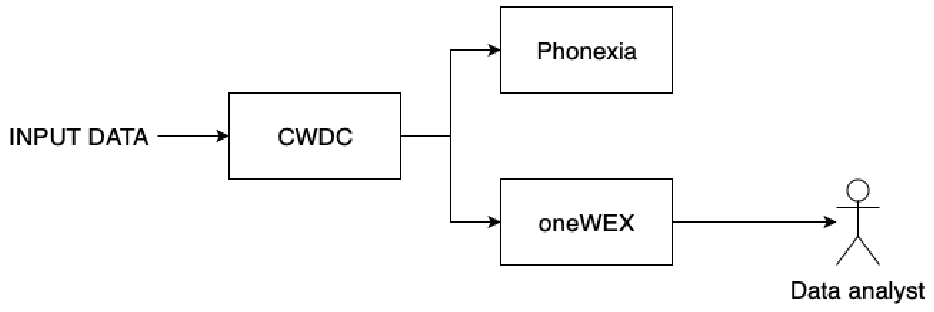


Figure 2.4: Final architecture

Chapter 3

Realisation of prototype

The prototype of this solution should be able to crawl the audio recordings in WAV format from the filesystem, extract metadata from voice, convert speech to text, perform basic data mining above the text data and visualize them in GUI.

3.1 Architecture

All of these components could work inside the Docker images, which gives us much wider usage and other advantages. The whole solution is designed as an on-premise solution because of the fact, that we will process sensitive information from the client.

That is the reason why we selected all parts to be (or to have an option) as on-premise software. The solution is a combination of three main modules, where every module is responsible for a certain part of the whole solution and is represented by standalone software, which provides the necessary functionality as you can see in figure 4.1.

The first part of this the prototype is Cogniware DataCollector as the orchestrator 3.2, which will allow us to crawl the data from the source filesystem and handle the extraction of data from voice tracks and store them for text processing 3.4.

The orchestrator picks up the file from the source, then stores the file on a Phonexia server for voice processing 3.3. This is the most crucial part because the accuracy of the rest of the text processing will be just as good as speech to text conversion. After the orchestrator extracts all necessary data from Phonexia's engine, it will create the JSON object with all these data and push them to the oneWEX index via REST API for further text mining 3.4.

The last part of the solution is IBM Watson Explorer oneWEX and it secures two parts of this solution. From the data processing view, it makes annotations and data extraction above the textual content via the ML annotators and holds the whole data inside the index. In addition, it extends the solution with GUI, where are the extracted data presented to the data analyst or another type of user.

3.2 Orchestrator

The main part of the whole solution is an orchestrator, which will allow us to perform the whole crawling and data handling between the modules. For this purpose, I chose Cogniware DataCollector (CWDC), which is a highly customizable solution prepared for similar use cases. This software is divided into three parts as I described in 2.3.

Both classes are very similar and CWDC is basically just a framework where we need to develop the main parts. We have to specify what the configuration files (JSON) should contain, how it will process the classes and many more. The biggest advantage is, that when we specify fields in configuration files, we do not need to redeploy the classes when we need to make the change (e.g., change of the source/target system, . . .), but we will be able to specify it in web-based GUI of CWDC.

3.2.1 Data connector

The data connector is responsible for crawling the source and extracting certain documents, from which we are going to take the content and push to the Data handler for further processing.

In our case, we have a filesystem as the source of data. I had to prepare a Java class, which extends the CommonCrawler. It needs to implement several methods for the purpose of crawling.

The first method is *init*, where we should add the connection to the database (usually RDBMS[11]), or connection to the logger of the CWDC. In this part, we will also parse the configuration JSON file where we have stored the information about the folder which we would like to crawl and/or the extensions of files which we will extract. After this method, we should have prepared the whole connector for the filesystem.

The next important method is *execute*. This method starts the whole crawling and it is executed after the *init*. The function is responsible for extracting the data from the source and make all the processing from the connector part. Inside this method, we have to write the code which extracts the data from the source and sends individual files (file paths) to the data handler for further processing. This code is also responsible for the change

[11]Relational Database Management System

of the Crawler status (Running/Stopped/Completed/Failed). In this method, we call another method named *crawlData*.

The *crawlData* method is responsible for extracting file paths of WAV files from the given folder via file path (specified in the configuration JSON). First of all, it extracts all the file paths and stores them in variable *filepaths* as *set*[12] of paths. This also gives us the number of all documents (recordings) inside this folder and then the CWDC GUI provides us a visualization of progress. After this stage, we have to start with the processing of these files.

Algorithm 1 Filepaths processing

```
 1: procedure CRAWLDATA
 2:     for each filepath $i$ in $filepaths$ do
 3:         if not shouldFileBeCrawled($i$) then //test to included/excluded
    file extension
 4:             continue;
 5:         end if
 6:         if processObject($i$) then //test if modified
 7:             Process($i$)
 8:         end if
 9:     end for
10: end procedure
```

First of all, we need to check if we want to process the file. The first check is based on the file extension (included/excluded file extension), if the file should be crawled, then we continue to the next check. The processing of the voice data is very difficult and it takes a lot of computer power, so before we start with the data processing, we should check, if the file was or was not proceed before. For this purpose, we have the database and we will start with check if the document is stored in the database (hash, the ID of the document or any other identification of file modification) or if the file was changed[13] For unchanged files we will skip data processing, in other cases, we will send the file for processing to the Data Handler. After every file, we increment the "unchanged" or "changed" counter. This is what we can see in the progress bar in CWDC's GUI.

3.2.2 Data Handler

The Data Handler is responsible for the whole data processing and extraction. First of all the whole part is initialized similarly to the previous data connector 3.2.1. Because of the fact that this class will send the data to the oneWEX for further processing, we have to specify the URL in the configura-

[12]Unordered collection of objects without duplications

[13]Database is a part of the CWDC, but we have to develop custom logic.

tion file, credentials, and target destination (dataset) for the data. After the initialization part, the module is prepared for data processing.

The Data Handler's class also prepares the JSON object, which we send to the oneWEX for further processing and visualization. This object contains the data from different parts (file's metadata, content, content's metadata, voice analysis, . . .).

Processing of the file starts, when the connector sends a filepath to the handler. First of all, we extract basic metadata from the file (e.g. file extension, size of the file, ...). Although we do not need this information, it will be valuable in case of extending the whole solution for example with the e-mail or other crawlers. This stage is the easiest from the performance perspective. In the second part, we will start with content extraction from the file containing the sound track.

3.2.2.1 Implementation of voice processing

We use Phonexia for voice processing. In our case, we make all requests via the *PhonexiaHelper* class, where we have a definition of all methods for the data extraction from voice tracks. We have also *RecordingModel* class in which we store all extracted data from track. This second class secures also conversion from Java object to a JSON object.

Because Phonexia has it's own web server, every file which we want to process we have to upload to this server. For every request, we have to be authenticated. "Phonexia Speech Engine provides two authentication methods: Token authentication and HTTP basic authentication." [32] We use the first method, where we obtain the token during the construction of the Phonexia-Helper object because it is more secure [33]. With every new crawled file, we create a new token, so it is harder to hack it. On the other hand, we usually work with one file just for a limited time and the time is usually less than 10 min for approx. 8 min voice recording.

After authentication and built of the helper object, we can start with the data processing. As we mentioned, we have to upload a file to the Phonexia's server. There we have a logical filesystem, where we can have folders, files and many more. In our case, we use only the root folder, where we are mirroring the files from our filesystem. In case of multiple folders, or even multiple data connectors, it will be a good idea to create folders for every connector (i.e. based on crawler ID) or to create some folder structure on the server, because if we upload file to the same destination (same file path) it rewrites the previous file without warning. When we have the file on the server, we can start with the running of Phonexia's functions above this file and every time we are pointing on the file via the file path.

Work with the audio track is CPU[14] intensive and it is very time-consuming. In these cases, we usually make these procedures asynchronous and the Phonexia

[14]Central processing unit

engine is no exception, most of the methods are asynchronous. Phonexia provides three options, how to process the requests – Polling [34], WebSocket [35], Webhook [36]. We chose polling for the simplification and because we are doing all requests sequentially anyway.

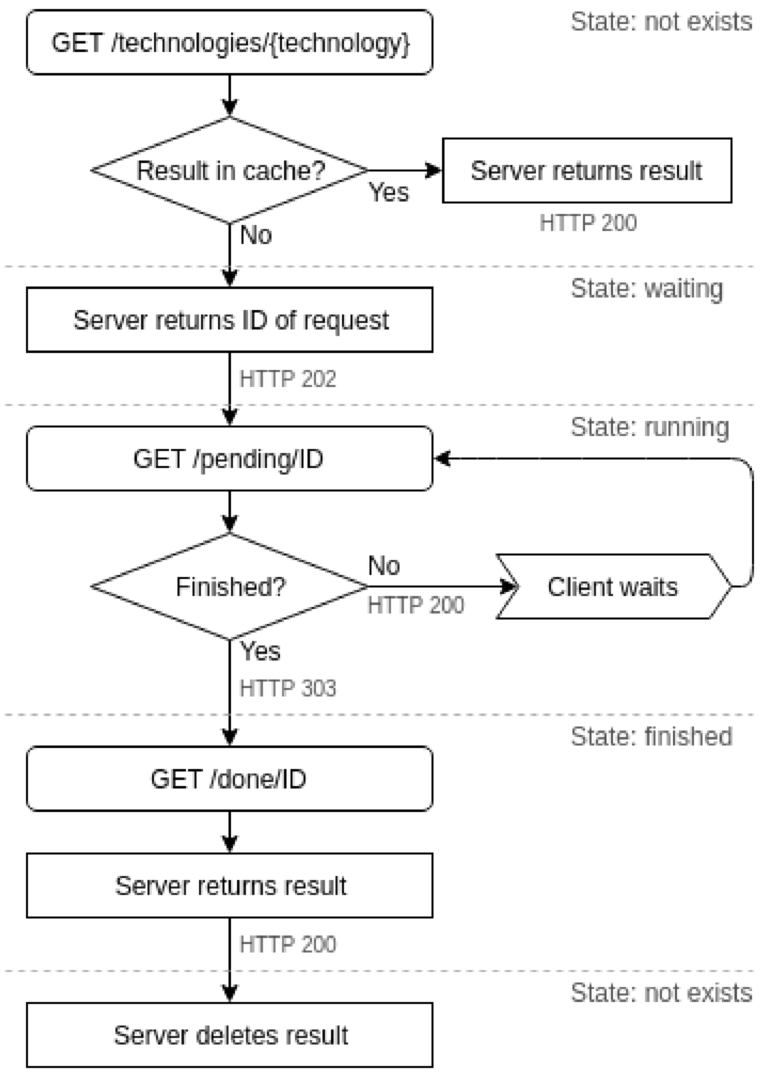


Figure 3.1: Scheme of polling request and response [32]

The main principle of polling is, that when we make the first request, it usually takes a longer time to complete the work, so we will receive the URL (or just ID[15] of the request) where we can check the status of the process (in our case it returns URL */pending/ID*). We will send GET requests to this URL until it returns response code 303, in other cases, it will give us 200 and

[15]Identifier

in the header, it will have “Location” redirection. When we will get 303 code, we know that the job is done and we can pick up the data from the server. You can see the whole scheme in the figure 3.1.

3.2.2.2 Methods used for data extraction from voice

All methods for voice extraction have a very similar run, we call the REST endpoint with the technologies, which we want to use for the file on specific space on the server (specified by file path). Then we are polling for data in 5-sec intervals. For almost all features we have to specify also the language of recording because we do not know what will be on the input, we have to start with language detection (LID).

Then we can call the rest of the services with detected language from the previous step. The rest of the calls are independent and we would call it simultaneously. In this prototype, we did not implement it and all requests run in one thread. The multithreading would be massive performance improvement and I would recommend implementing it in case of running in production (or improvement of this thesis). We run these features step by step (under every feature is sample information which we extracted in this step):

1. Upload file to Phonexia server
    - Number of channels
    - Length of recording
    - Format
    - Frequency
2. Language identification
    - Language and score of language (confidence) for every channel
3. Speech to Text
    - The textual representation of content for every channel

3.3 The Voice processing module

In this section, we will look into the implementation of voice recording analysis. The main part of voice processing is software, which will perform all metadata extraction from voice recordings. For this purpose I choose a Phonexia platform, mainly because it could work as an on-premise solution, their specialization is call centers and their engine is highly customizable for better accuracy.

3.3.1 License cycle management

Their server with the whole engine has REST API for voice recording management and data extraction. It is licensed in many ways based on our requirements (USB dongle, HW license, license server).

Our solution is hosted in the VMWare [37] environment on the cloud, so it is unable to connect USB dongle and because of the virtualized environment, we also cannot use HW license. In our case, we chose licensing via the license server. For licensing, we received a file where we have specified how many concurrent methods of the same features can be executed.

E.i., we have license 6 times LID[16], which means that we can run LID for 6 concurrent files, but it does not mean, that we can not run STT[17] simultaneously, because for other features we have other licenses. So the license is based on the method and number of concurrent executions. When we use any method, we connect our system to the Phonexia's environment and check if we have or do not have a free license for this job. Every job (method execution) consumes 1 license from the pool and when the work is done, the license server is notified and 1 license is refilled. That means we need an internet connection from the server to the public internet, or at least to the licensing server.

3.3.2 Technologies customization

Phonexia has many options to improve the already good accuracy. Some of them are possible to make just by choosing the languages to the package with which we will work, other improvements are more difficult as the training new language. In our case, we improved the solution via Langauge identification package.

3.3.2.1 Language identification

Thanks to the chosen technology and the distribution of the recording among all languages in the package, it is not recommended to run the identification over the "default" package, but we should choose and create our own package (whether using pre-trained languages or our own model). As part of the solution planned in accordance with the objectives of the Bachelor thesis, we selected only 2 languages. These are Czech and English_American because we expect that all recordings will be in Czech or English only. There is just one model for the detection of the Czech language, so there are not many other ways to choose, but they have more models for English. So we had to choose only one language with prefix "English" in our package, or all languages with this prefix.

[16]Language identification feature
[17]Speech to text feature

If we used both (English_American and English_British), the probability of the English recording would be significantly reduced for both languages. It is because of the confidence which would have to be divided among two languages in addition to the Czech language. At the same time, only one English model offers Speech to Text conversion (described in the following subsection 2.1.3.5), so it is not so important for us to know the exact dialect of English, but to have a higher value of confidence and the wider difference between Czech and English would be more important. That is why we used only two languages in our language pack and these are “Czech” and “English_American”. With this decision, we made a greater difference in confidence and reduced the possible error rate.

It is this voice-based language identification that allows us to run the correct language model on the recording when we run the Speech to Text conversion.

3.4 Text processing module

Textual data mining is another crucial part of the whole solution. We could create this part from scratch and create it on top of Elasticsearch or Solr for full-text searching capabilities, but in these cases, the development itself would be too hard. Instead of that, we chose the IBM Watson Explorer oneWEX, which has already undergone a long development process and has all the features which we needed for the prototype.

OneWEX has one big disadvantage. Although the product was coming out from the older IBM Watson Explorer (containing Analytical and Foundation components), it is still in development and some of the features are on the roadmap, but they are not implemented yet. At the same time, there is a significant number of small bugs and the documentation is weak. This complicated the whole development process and some of the parts were more about guessing, based on the experiences with older products, how it could work.

Our main goal in this section was divided into two parts. In the first part, we need to investigate possibilities of the REST API which has great documentation via Swagger [38], but some of the methods are not implemented without any warning. The second part was in the dataset and collections preparation.

3.4.1 REST API investigation and data injection

Although oneWEX provides options to crawl data directly via the predefined crawlers, we cannot use them as we discussed in section 2.3. That is the reason, why we have to use the REST API capabilities of oneWEX for data injection into the dataset. We can manage almost everything via the API calls, but as we said, some of the features are not implemented yet and working with datasets and collections is very problematic. For this reason, we have to decide to make this part via administration GUI of oneWEX.

In the end, we chose to use the REST API only for the data injection, there were some other issues and in some cases, we had to go with method of Trial and error[18]. E.g., when we sent the JSON POST request to inject the data to the dataset, oneWEX was unable to process the nested JSON (on listing 3.1) and we were able to process and index only JSON with flat structure (on listing 3.2).

```
{
"id": "file.wav",
"filepath": "/example/file.wav",
"duration": 230.53,
"channels": [
{
"number": 0,
"language": "english",
"score": 0.8903
},
{
"number": 1,
"language": "english",
"score": 0.9402
}
]
}
```

Listing 3.1: Example of nested JSON

```
{
"id": "file.wav",
"filepath": "/example/file.wav",
"duration": 230.53,
"language": "english"
}
```

Listing 3.2: Example of flat JSON

[18]Method of problem-solving where we repeatedly try to change attempts until we solve the problem

The second biggest problem during the process of developing the data injection via REST API was some unspecified keys that could not appear in the JSON. E.g., the *length* is one of them. When we would like to use it for the length of the recording, but after the oneWEX detects this key, it somehow bypasses the processing of the request. The number of documents in dataset was increased, but the record did not appear and we could not show this document.

All of these bugs and weird behavior of the oneWEX were reported to the development team of the product. We could not wait for the fix now, so we dealt with it by renaming of the field. Hopefully, we will see the fix of these issues in one of the next fix packs (or at least documentation of them).

3.4.2 Dataset and collection creation

As previously shown in section 2.2.3, data is stored in the dataset and one or more datasets can be connected to the collection. Then all exploration is made above the specified collection (same for searching and data analysis).

3.4.2.1 Dataset

Dataset is a big index where are the data stored. In our case, we have only one dataset where we have textual representation of voice recordings with metadata and extracted information. First of all, we have to create the dataset and specify the fields of documents which we want to have there.

To create the dataset is the easiest part, we will just specify the name and description of the dataset. The hard part is to prepare it for data injection. Unfortunately, there is no option to create fields (specification of name and type) and the REST API for this purpose did not work properly.

So we were forced to do it via the CSV file importer. We created a CSV file with a proper header and one row as a sample document (row has to have at least one field filled). During the import of the CSV, we can add these fields from CSV to the dataset and it is now only way how you can specify your own fields and prepare it for the data crawling via API. The dark side of this approach is, that we will have one document in the index which we cannot delete (because oneWEX has not implemented method for the deletion of the document). After we have specified all necessary fields, we can start with the creation of a collection.

3.4.2.2 Collection

Collection is a combination of multiple datasets, above which we are making all searches or data analysis. We can add more datasets to the collection, but in our case, we have only one dataset which we connect to this collection. During the collection creation, we will select, which fields we want to include into the collection. This gives us an option to exclude some fields, which are in the dataset (in documents), but which we do not want to have in the collection. In some cases, this could make better security when we want to have the same data (same dataset), but for some people, we want to show different fields than for others (e.g., information for data analyst and publicly available data). During this field setup, we can make the field searchable or not and also create some fields facets.

For the typical application, we have to specify which fields will be a body, title, and date. This is usual for the search applications, for basic data mining it is not mandatory.

Next interesting part for all of the data mining application is the option to extract information from the text. All of the annotators/classifiers are language dependent and we cannot start without knowing the language. This part is similar to the Phonexia and the first thing that we need to identify is the language of the document. If we know languages, which could be on the input, we can specify them, or we can let the oneWEX automatically decide which language will be the best for the document. Based on our dataset and experience we will choose the Czech and English language.

Now we have set up most of the static parts, oneWEX knows languages for the document and we can start with data extraction. OneWEX gives us many possibilities, in our use case we will use out-of-box annotators, we will opt-in *Part of Speech*[19]*, Sentiment Analysis*[20]*, Named Entity Recognition*[21] *and PII Annotator*[22].

[19]Extract words and phrases from unstructured content and mark these extractions as annotations

[20]Extract phrases and expressions which convey sentiment

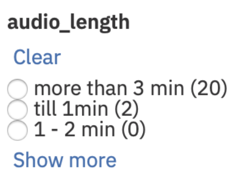
[21]Extract person names, locations, and organizations from unstructured content

[22]Extract personal identifiable information with built-in regular expressions.

All of these annotations are stored as facets and we have to specify how we want to show them and then how we can filter results based on that. Facets which are numbers, and we do not need each of these files separately, but we will appreciate to have them in some ranges (length of recording, date) we usually specify as ranges with the structure you can see in listing 3.3 (this needs to be specified for every facet/filter) and an exmaple of this filter you can see in figure 3.2.

```
[
{
"label":"till 1 min",
"query":"[0, 60)"
},
{
"label":"1 - 2 min",
"query":"[60, 120)"
},
{
"label":"more than 3 min",
"query":"[120, 9999999)"
}
]
```

Listing 3.3: Example of length of recording range


audio_length

Clear

- more than 3 min (20)
- till 1min (2)
- 1 - 2 min (0)

Show more

Figure 3.2: Range filter example

Other facets (textual, individual numbers) we can aggregate and visualize as a table or as world cloud. All of these facets could be used for the filtering of the documents to obtain a better view and more precise insight.

The last part which could possibly improve the search result list is the Exploration part of collections. For this thesis, this is not the most important part, but it gives the opportunity to study these sections more in i.e. diploma thesis. On the other hand in this section, we can set up more precisely NLQ[23],

[23]Natural language query

where we can extend the built-in black/white list of oneWEX. We can also add stop words, synonyms, and boost (or make a lower score) any field within the collection. In the end, we can use ML training for better results (this was added in Fix pack 12.0.2.2). OneWEX has many options to rank higher or lower some of the results based on Collaborative features and there we have the option to train a ML model which will influence the results based on the number of clicks on the specific document.

3.5 Graphical user interface

We extracted a lot of information from the file, voice and text data. But we have it only in text representation, which is good for the computer, but it is not enough for the data analysts. They need to have all the crucial data in dashboards and in an eye-pleasant way. For this purpose, we use also Watson Explorer oneWEX. These features for showing the information are separated into two different parts. The first one is Builder, where we can create our own UI on top of collection/s and Miner.

Whereas Builder is prepared for the simple development of UI, Miner is a standalone part prepared for the data analyst, where we do not need to change anything and we will have great UI for data insight. We wanted to have a search and mining part in the same application, where we do not want to change the URL when we want to do different work. We decided to create UI where we have the ability to easily switch between these two parts. We developed two tabs – Search and Miner, in the UI. That will allow data analysts to change the tool whenever they want.

3.5.1 GUI – Search

The Search tab is designed for full-text searching capabilities and to have an option to filter the documents which we have in the collection. The whole window is divided into 6 parts as you can see in the figure 4.2.

1. Tabs to switch between Miner and Search parts
2. Search input field
    - Natural language queries
    - Field/facet search
    - Supports wildcards, logic operators, ...
3. Refinements (filters/facets)
    - Filter based on the facets defined in collections 3.4.2.2
4. Search results list

- Shows results which satisfy the search query and set refinements

5. Document's metadata and text content
   - Shows metadata of selected document
   - Shows extracted textual content of selected document

3.5.2 GUI – Miner

The second tab is prepared for the data analyst to give better insight into the data and we call it Miner. This part is built-in in oneWEX, but it is separated from other parts and we need to think about it, when we want these capabilities in our application. In our case, we accomplished that via the iframe, where we injected the proper miner part of oneWEX. This will allow data analyst to easily switch between the miner and search without high effort and gives the opportunity to use both parts of the solution on one page. In figure 4.3 you will see expert mode for content analysis.

1. Tabs to switch between Miner and Search parts
2. Facet Analysis
   - Selection of facets to analyse
   - Selection of feature you want to analyse with
3. Analysis dashboard
   - Shows insights of data you have in the collection based on your selection in Facet analysis part
4. Other tools
   - Ability to make a deeper analysis
   - Show documents which suit to the selected conditions in dashboard

Chapter 4

Outputs

The output of the thesis is a design of the architecture of software solution and prototype implementation which is able to extract data from the WAVE audio files, make annotations (extraction of phrases, sentiment, named entity recognition, ...) and show this information to the agent for further decision making.

The final software solution prototype is able to automatically (based on scheduling) take audio recordings from the filesystem, extracts information from audio content and convert it into the text. Then it looks into the text content, makes automatic in-depth text mining and performs predefined annotations. After the whole processing, the information are showed in UI where the agent has an ability to search in the content (showed on figure 4.2) or look into the analysis (showed on figure 4.3 and). This thesis and prototype bring the possibility to improve decisions based on the new data extracted from audio recordings.

In the architecture in figure 4.1 we used three modules, where we have Phonexia for audio information extraction, IBM Watson Explorer oneWEX for text analysis and information visualisation and Cogniware DataCollector for the orchestration of all requests and data crawling.

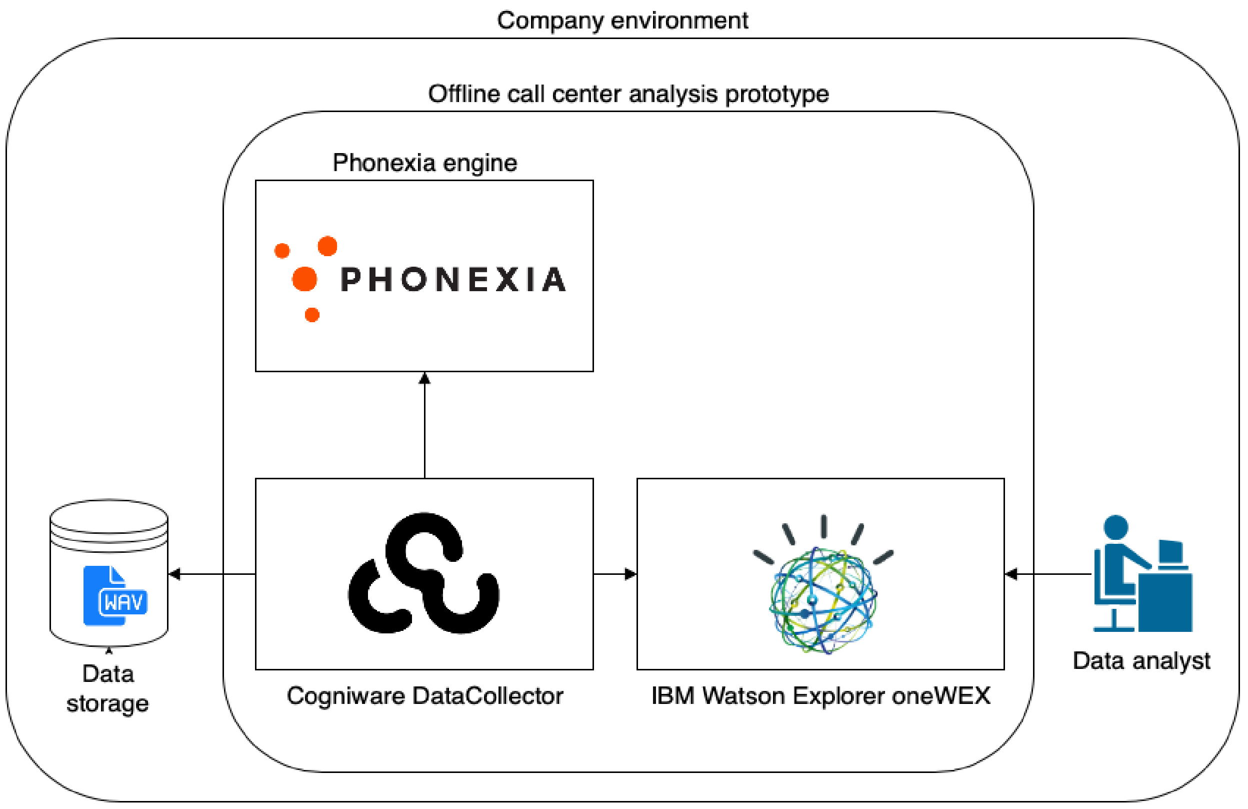
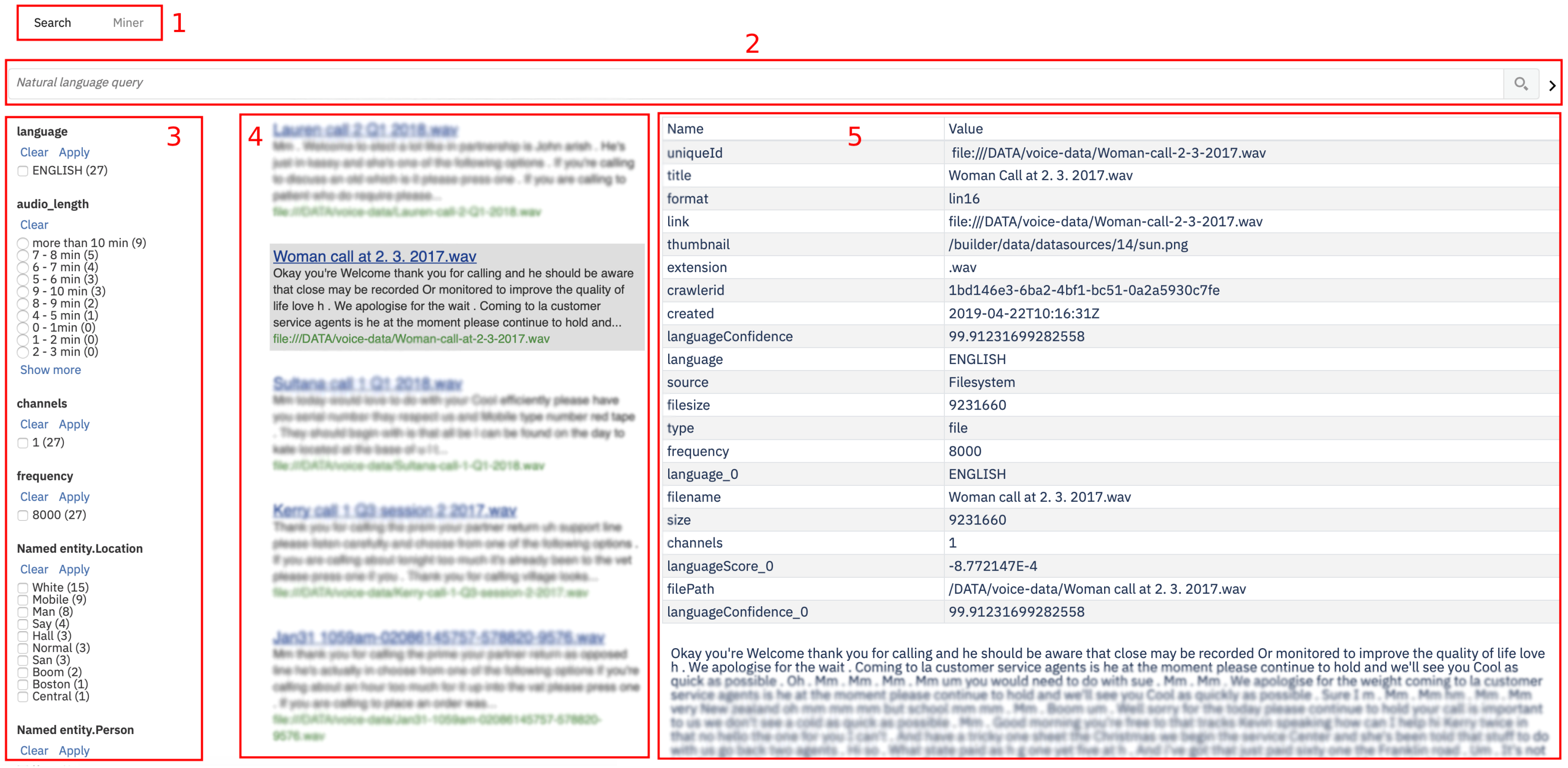


Figure 4.1: High-level architecture



language
Clear Apply
☐ ENGLISH (27)

audio_length
Clear
○ more than 10 min (9)
○ 7 - 8 min (5)
○ 6 - 7 min (4)
○ 5 - 6 min (3)
○ 9 - 10 min (3)
○ 8 - 9 min (2)
○ 4 - 5 min (1)
○ 0 - 1min (0)
○ 1 - 2 min (0)
○ 2 - 3 min (0)
Show more

channels
Clear Apply
☐ 1 (27)

frequency
Clear Apply
☐ 8000 (27)

Named entity.Location
Clear Apply
☐ White (15)
☐ Mobile (9)
☐ Man (8)
☐ Say (4)
☐ Hall (3)
☐ Normal (3)
☐ San (3)
☐ Boom (2)
☐ Boston (1)
☐ Central (1)

Named entity.Person
Clear Apply

[illegible]

Woman call at 2. 3. 2017.wav
Okay you're Welcome thank you for calling and he should be aware that close may be recorded Or monitored to improve the quality of life love h . We apologise for the wait . Coming to la customer service agents is he at the moment please continue to hold and...
file:///DATA/voice-data/Woman-call-at-2-3-2017.wav

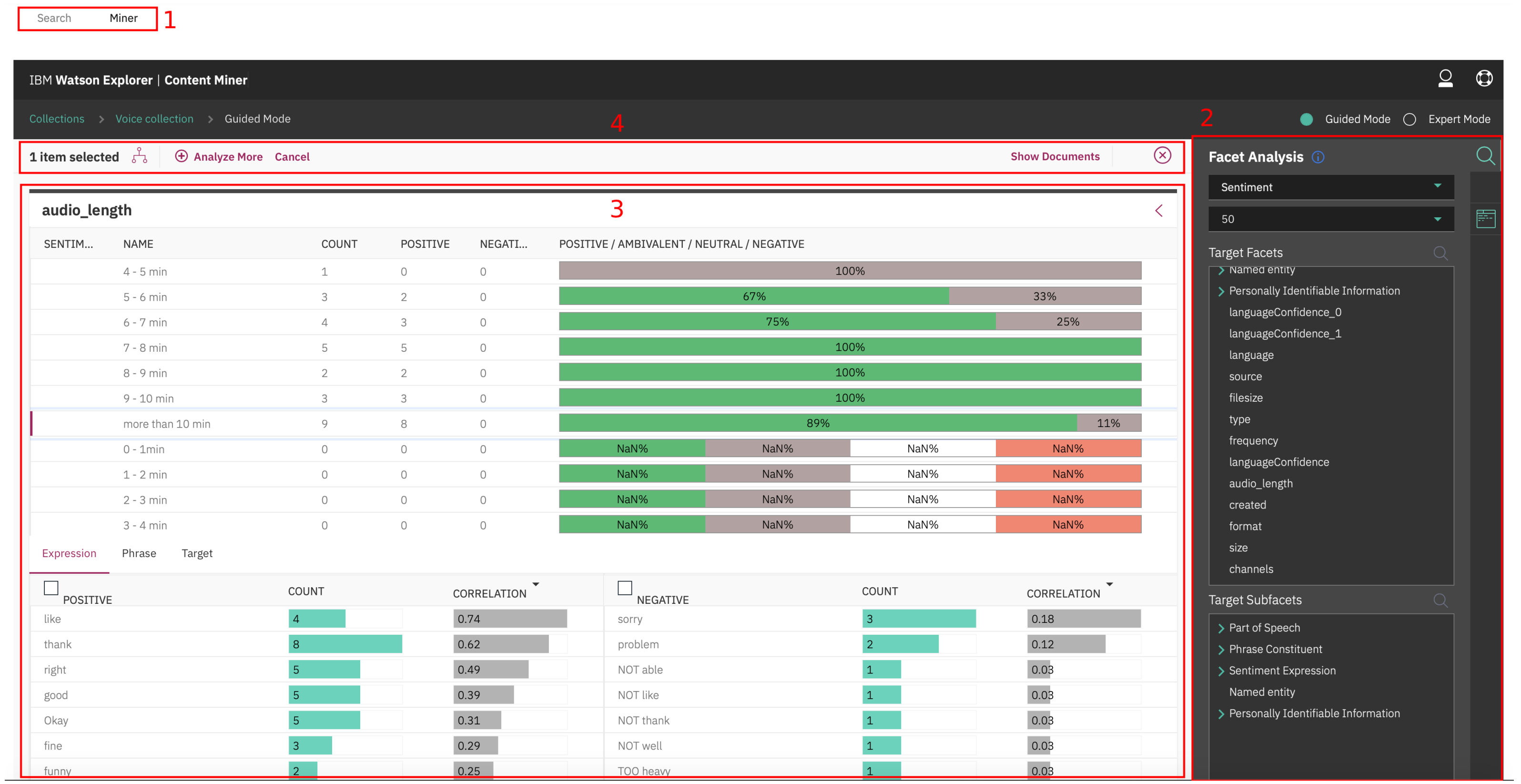
[illegible]

Name	Value
uniqueId	file:///DATA/voice-data/Woman-call-2-3-2017.wav
title	Woman Call at 2. 3. 2017.wav
format	lin16
link	file:///DATA/voice-data/Woman-call-2-3-2017.wav
thumbnail	/builder/data/datasources/14/sun.png
extension	.wav
crawlerid	1bd146e3-6ba2-4bf1-bc51-0a2a5930c7fe
created	2019-04-22T10:16:31Z
languageConfidence	99.91231699282558
language	ENGLISH
source	Filesystem
filesize	9231660
type	file
frequency	8000
language_0	ENGLISH
filename	Woman call at 2. 3. 2017.wav
size	9231660
channels	1
languageScore_0	-8.772147E-4
filePath	/DATA/voice-data/Woman call at 2. 3. 2017.wav
languageConfidence_0	99.91231699282558

Okay you're Welcome thank you for calling and he should be aware that close may be recorded Or monitored to improve the quality of life love h . We apologise for the wait . Coming to la customer service agents is he at the moment please continue to hold and we'll see you Cool as quick as possible . Oh . Mm . Mm . Mm . Mm um you would need to do with sue . Mm . Mm . We apologise for the weight coming to la customer [illegible]

Figure 4.2: Prototype GUI – Search

1. Tabs to switch between Miner and Search parts
2. Search input field
3. Refinements (filters/facets)
4. Search results list
5. Document's metadata and text content



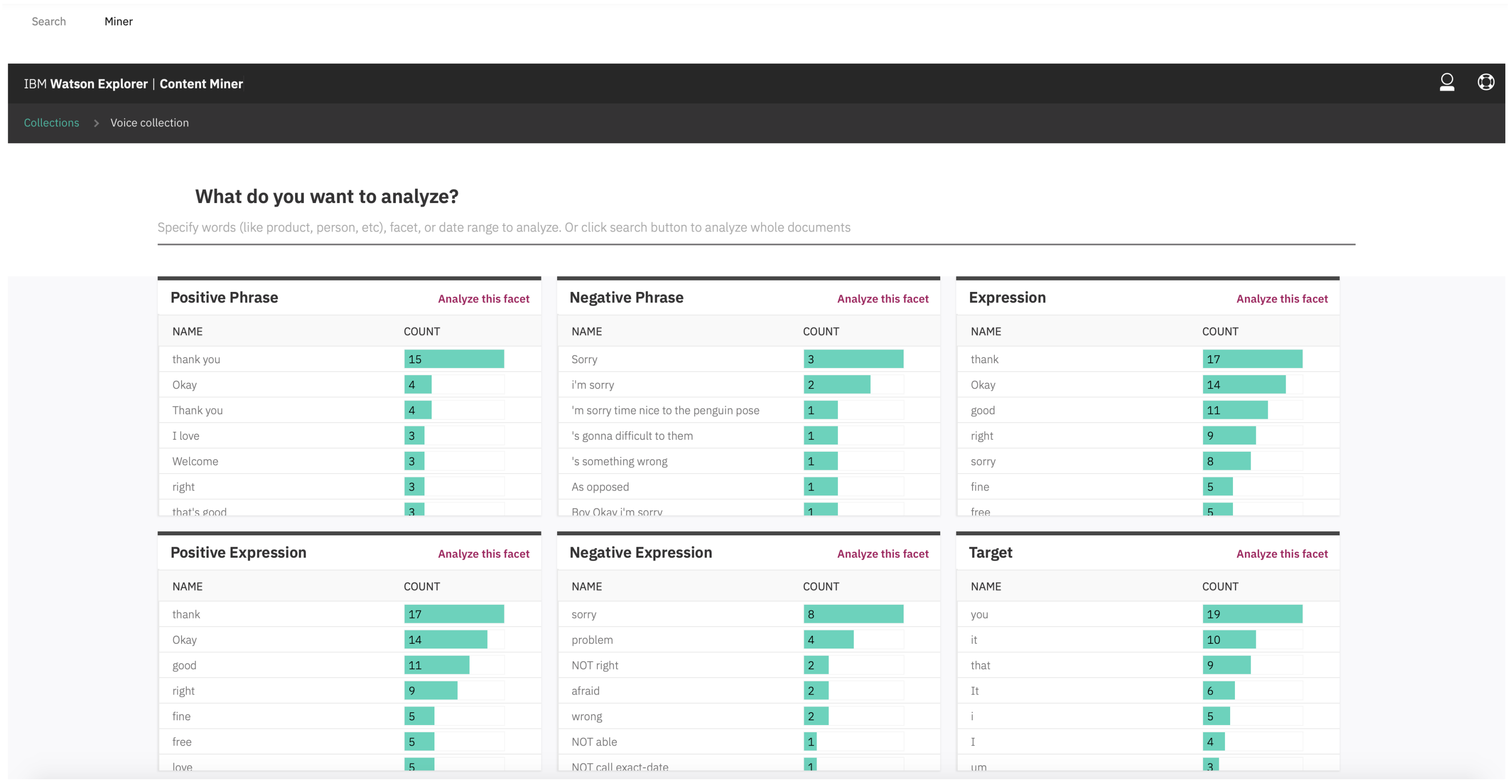
audio_length

SENTIM...	NAME	COUNT	POSITIVE	NEGATI...	POSITIVE / AMBIVALENT / NEUTRAL / NEGATIVE
	4 - 5 min	1	0	0	100%
	5 - 6 min	3	2	0	67% 33%
	6 - 7 min	4	3	0	75% 25%
	7 - 8 min	5	5	0	100%
	8 - 9 min	2	2	0	100%
	9 - 10 min	3	3	0	100%
	more than 10 min	9	8	0	89% 11%
	0 - 1min	0	0	0	NaN% NaN% NaN% NaN%
	1 - 2 min	0	0	0	NaN% NaN% NaN% NaN%
	2 - 3 min	0	0	0	NaN% NaN% NaN% NaN%
	3 - 4 min	0	0	0	NaN% NaN% NaN% NaN%

POSITIVE	COUNT	CORRELATION	NEGATIVE	COUNT	CORRELATION
like	4	0.74	sorry	3	0.18
thank	8	0.62	problem	2	0.12
right	5	0.49	NOT able	1	0.03
good	5	0.39	NOT like	1	0.03
Okay	5	0.31	NOT thank	1	0.03
fine	3	0.29	NOT well	1	0.03
funny	2	0.25	TOO heavy	1	0.03

Figure 4.3: Prototype GUI – Miner – Expert mode

1. Tabs to switch between Miner and Search parts
2. Facet Analysis
3. Analysis dashboard
4. Other tools


Search
Miner
IBM Watson Explorer | Content Miner
Collections > Voice collection
Analyze this facet

What do you want to analyze?

Specify words (like product, person, etc), facet, or date range to analyze. Or click search button to analyze whole documents

Positive Phrase

NAME	COUNT
thank you	15
Okay	4
Thank you	4
I love	3
Welcome	3
right	3
that's good	3

Negative Phrase

NAME	COUNT
Sorry	3
i'm sorry	2
'm sorry time nice to the penguin pose	1
's gonna difficult to them	1
's something wrong	1
As opposed	1
Boy Okay i'm sorry	1

Expression

NAME	COUNT
thank	17
Okay	14
good	11
right	9
sorry	8
fine	5
free	5

Positive Expression

NAME	COUNT
thank	17
Okay	14
good	11
right	9
fine	5
free	5
love	5

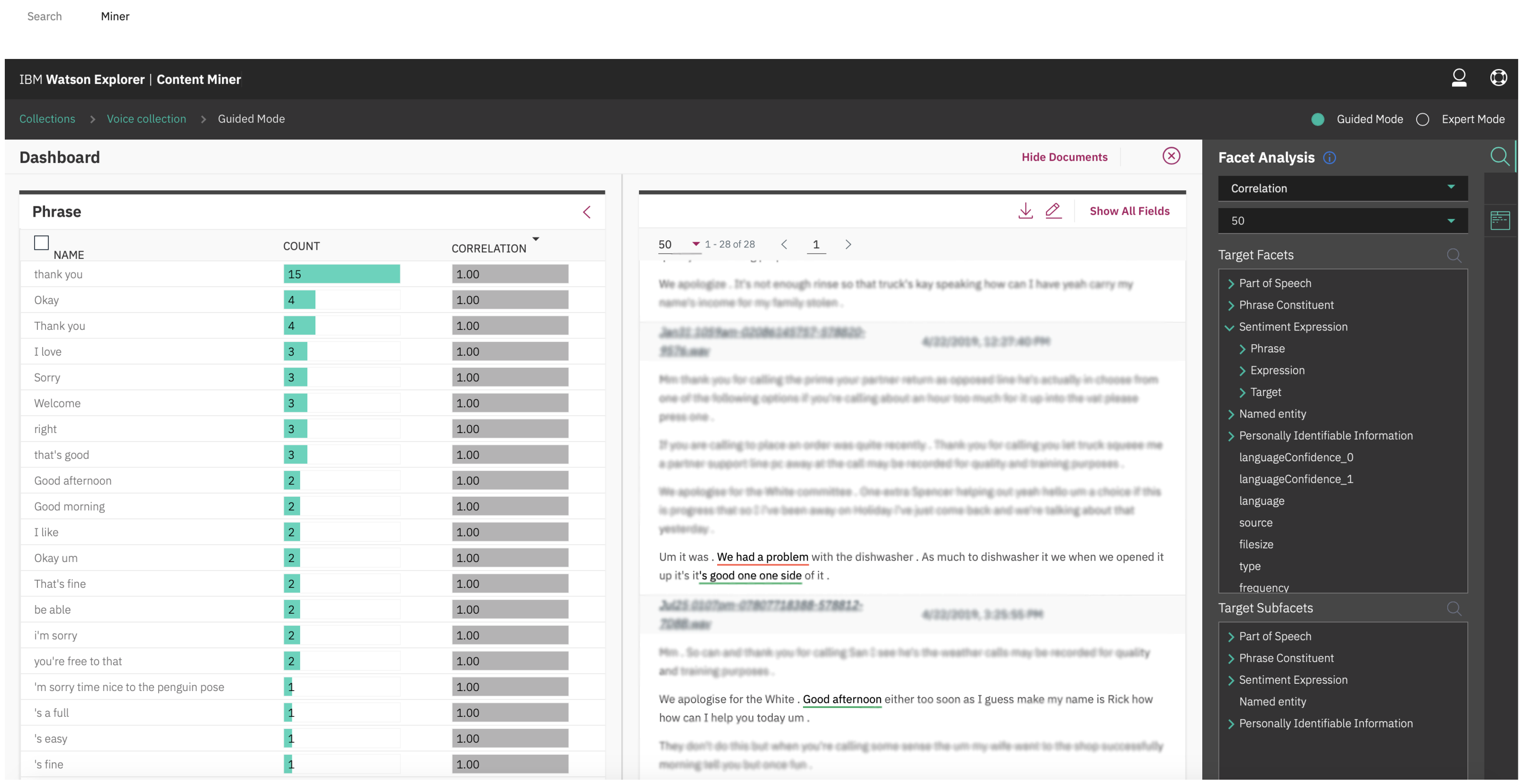
Negative Expression

NAME	COUNT
sorry	8
problem	4
NOT right	2
afraid	2
wrong	2
NOT able	1
NOT call exact-date	1

Target

NAME	COUNT
you	19
it	10
that	9
It	6
i	5
I	4
um	3

Figure 4.4: Prototype GUI – Miner – Sentiment facets analysis



NAME	COUNT	CORRELATION
thank you	15	1.00
Okay	4	1.00
Thank you	4	1.00
I love	3	1.00
Sorry	3	1.00
Welcome	3	1.00
right	3	1.00
that's good	3	1.00
Good afternoon	2	1.00
Good morning	2	1.00
I like	2	1.00
Okay um	2	1.00
That's fine	2	1.00
be able	2	1.00
i'm sorry	2	1.00
you're free to that	2	1.00
'm sorry time nice to the penguin pose	1	1.00
's a full	1	1.00
's easy	1	1.00
's fine	1	1.00

[redacted]

Um it was . We had a problem with the dishwasher . As much to dishwasher it we when we opened it up it's it's good one one side of it .

[redacted]

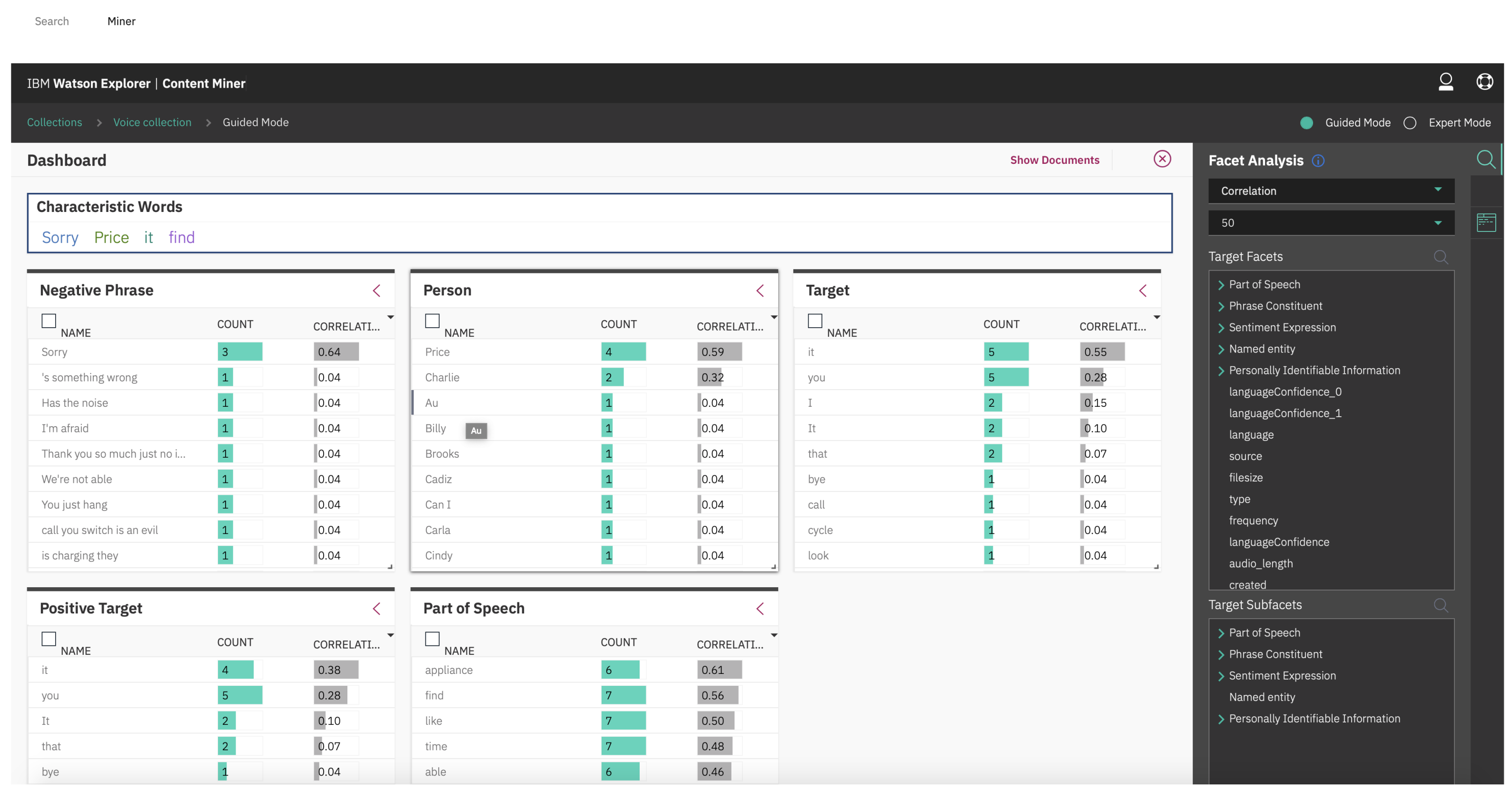
We apologise for the White . Good afternoon either too soon as I guess make my name is Rick how how can I help you today um .

[redacted]

Target Facets
- Part of Speech
- Phrase Constituent
- Sentiment Expression
  - Phrase
  - Expression
  - Target
- Named entity
- Personally Identifiable Information
- languageConfidence_0
- languageConfidence_1
- language
- source
- filesize
- type
- frequency

Target Subfacets
- Part of Speech
- Phrase Constituent
- Sentiment Expression
- Named entity
- Personally Identifiable Information

Figure 4.5: Prototype GUI – Miner – Phrase sentiment analysis


Search Miner

IBM **Watson Explorer** | **Content Miner**

Collections > Voice collection > Guided Mode

● Guided Mode ○ Expert Mode

Dashboard

Show Documents

Characteristic Words

Sorry Price it find

Negative Phrase

NAME	COUNT	CORRELATI...
Sorry	3	0.64
's something wrong	1	0.04
Has the noise	1	0.04
I'm afraid	1	0.04
Thank you so much just no i...	1	0.04
We're not able	1	0.04
You just hang	1	0.04
call you switch is an evil	1	0.04
is charging they	1	0.04

Person

NAME	COUNT	CORRELATI...
Price	4	0.59
Charlie	2	0.32
Au	1	0.04
Billy	1	0.04
Brooks	1	0.04
Cadiz	1	0.04
Can I	1	0.04
Carla	1	0.04
Cindy	1	0.04

Target

NAME	COUNT	CORRELATI...
it	5	0.55
you	5	0.28
I	2	0.15
It	2	0.10
that	2	0.07
bye	1	0.04
call	1	0.04
cycle	1	0.04
look	1	0.04

Positive Target

NAME	COUNT	CORRELATI...
it	4	0.38
you	5	0.28
It	2	0.10
that	2	0.07
bye	1	0.04

Part of Speech

NAME	COUNT	CORRELATI...
appliance	6	0.61
find	7	0.56
like	7	0.50
time	7	0.48
able	6	0.46

Facet Analysis

Correlation

50

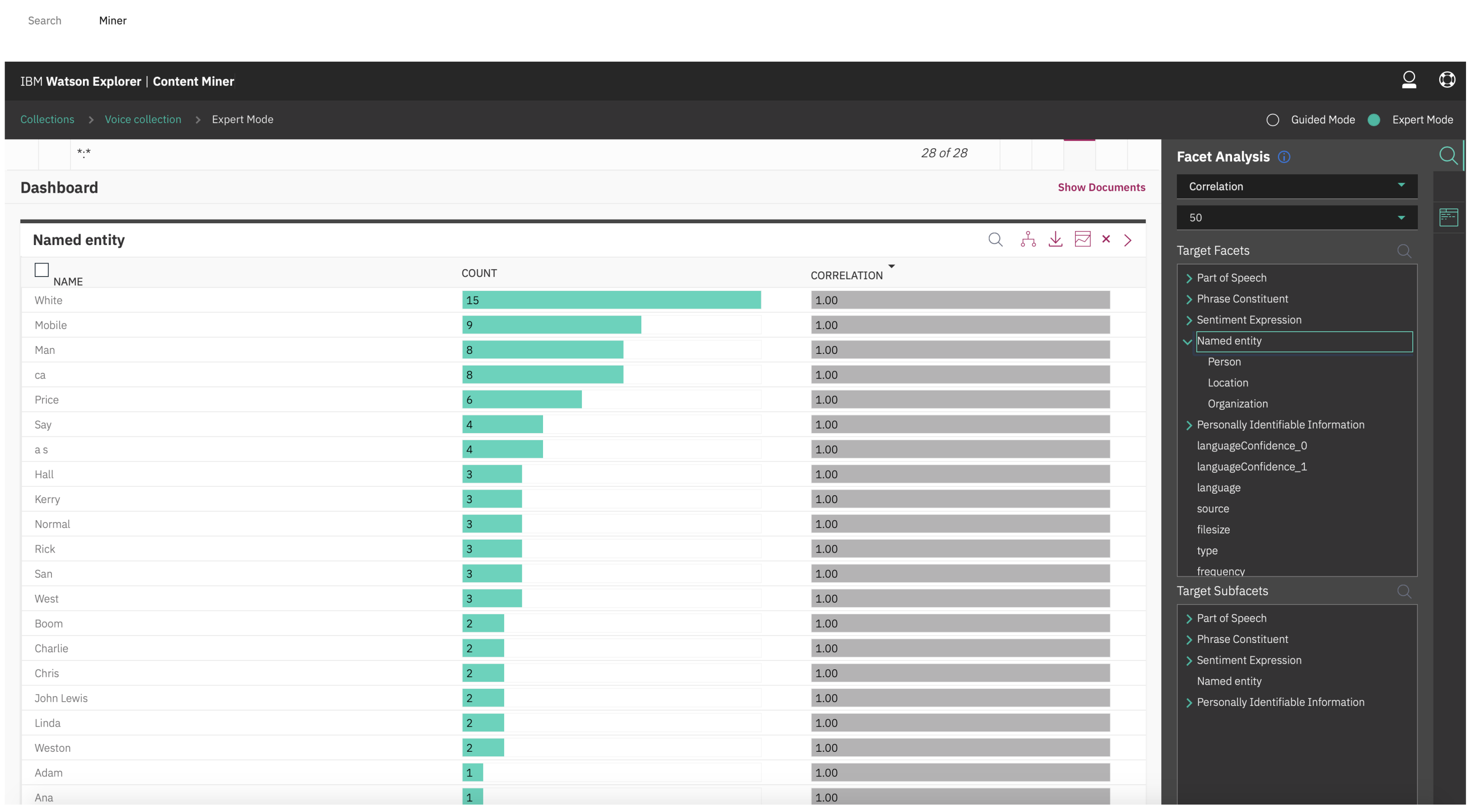
Target Facets

- Part of Speech
- Phrase Constituent
- Sentiment Expression
- Named entity
- Personally Identifiable Information
- languageConfidence_0
- languageConfidence_1
- language
- source
- filesize
- type
- frequency
- languageConfidence
- audio_length
- created

Target Subfacets

- Part of Speech
- Phrase Constituent
- Sentiment Expression
- Named entity
- Personally Identifiable Information

Figure 4.6: Prototype GUI – Miner – Cause or characteristics analysis


Search Miner

IBM Watson Explorer | Content Miner

Collections > Voice collection > Expert Mode

Guided Mode ● Expert Mode

: 28 of 28

Dashboard

Show Documents

Named entity

NAME	COUNT	CORRELATION
White	15	1.00
Mobile	9	1.00
Man	8	1.00
ca	8	1.00
Price	6	1.00
Say	4	1.00
a s	4	1.00
Hall	3	1.00
Kerry	3	1.00
Normal	3	1.00
Rick	3	1.00
San	3	1.00
West	3	1.00
Boom	2	1.00
Charlie	2	1.00
Chris	2	1.00
John Lewis	2	1.00
Linda	2	1.00
Weston	2	1.00
Adam	1	1.00
Ana	1	1.00

Facet Analysis

Correlation

50

Target Facets

- Part of Speech
- Phrase Constituent
- Sentiment Expression
- Named entity
  - Person
  - Location
  - Organization
- Personally Identifiable Information
- languageConfidence_0
- languageConfidence_1
- language
- source
- filesize
- type
- frequency

Target Subfacets

- Part of Speech
- Phrase Constituent
- Sentiment Expression
- Named entity
- Personally Identifiable Information

Figure 4.7: Prototype GUI – Miner – Named entity extraction and analysis

Conclusion

The aim of this thesis was to analyze the requirements, design an architecture and implement the prototype of the solution, which will be able to analyze the content of voice recordings based on the textual representation.

In the State-of-the-art section we investigated call center's methods for analyzation of voice calls, where we found out that most of the companies measure only frequency and length of call analysis, but they do not deal with the content itself.

In the Analysis and design section we look into the audio and text analysis software solutions, where we recognize their pros and cons. Based on the parameters we have chosen Phonexia for the audio analysis and speech to text conversion, mainly because of their focus on call centers. For the text analysis we chose IBM Watson Explorer oneWEX, because of its wide variety of options for annotations, classifications, and simplicity of implementation. We found the need to use different crawler for data injection, where we chose Cogniware DataCollector.

In the Realisation of the prototype section we introduced the architecture of the whole solution and described the implementation of the prototype with the customizations for better accuracy of the speech recognition. In this part we also created a GUI for data analysts.

The software solution was designed and developed on the basis of technologies, which allow to convert the audio recording into a textual form and enable future analyzation of the content. Advantage of the solution architecture is its ability to run on one or more servers, providing tremendous flexibility and easy scalability. It also means that the entire server may be in the company network and no data can be accessible without authorization to the system where they are processed, thus preventing GDPR violations.

This solution is a great tool for analyzing audio recordings from call centers. Of course, this is an application that can be run for any contact center independent on the company's products or their target group. Some other data might be important for analyzing results, or other visualization, but this is set for individual company needs because every company wants to see different pieces of information on their dashboards in GUI.

The data processing progress is customizable thanks to the selected component, such as DataCollector where it is possible to change the settings to get more information from the REST API or from any data from other sources (CRM, etc.). The biggest advantage is language independence which does not affect supported languages.

Eventhough the IBM Watson Explorer oneWEX is still in development and many parts have their own limitations, it has already become a quality element that is difficult to replace. Among other things it helped me a lot with the processing of data and the extraction of knowledge from the textual form. In the future it would be interesting to use it independently to retrieve data from the audio track or at least get their text data, for example using one of the IBM Cloud services such as Watson Speech to Text. On the other hand oneWEX main focus is the text processing and we can expect connections to other services for text mining from IBM's portfolio, instead of audio manipulation.

The whole solution is very robust, which means that it is possible to use many languages, especially thanks to the wide conversion of voice tracks in Phonexia and supported languages in text form in oneWEX.

During the analysis part I had to analyse the requirements which are placed in the final solution. At the same time I had to recognize which modules need to be in the architecture to get all necessary informations. Depending on the information I wanted to extract I had to determine features which I would request from the specific software solutions. Based on the requirements I chose appropriate software solutions which would enable data extraction from voice recordings and which would allow me to extract information from textual data and define how these modules connect together via orchestrator where I implemented the connections. Penultimate task was the decision how to show the information to agents where I had to choose software which simplifies and implements the customizations. Finally, I checked that all the modules were working properly together, and all important information are showed to the agent for further decision making.

Based on the goal the thesis succeed in the design of the solution. Regarding IBM there are now 3 opportunities on the market.

Possible further improvements

During the development and analysis, we found a lot of different improvements, which are out of the scope of the thesis. However their implementation would be tremendously useful. These proposed extensions may be proposed for the implementation in a follow-up diploma thesis.

In general, the most important disadvantage of sound conversions is the need for more computational power as I have seen on my own test data. Single recording of approx. 8.9 min duration on one CPU is processed for 9.5 min but this problem can be solved by parallelizing individual tasks.

As it results from the text, the solution could be extended in the future by acquiring data, for example from e-mail communication, where we would be able to add a new crawler and data handler in DataCollector or other sources. Video processing could be added simultaneously as a massive expansion of video reviews, etc., can be expected in the upcoming years.

Big companies have many clients which usually ask similar questions and call centers could use it. If we create knowledge base from these recordings it will help the agents to solve the client's problems. In this case we will need to extract some information from the recordings or at least classify and categorize them.

Bibliography

[1] THE WORLD BANK. Unemployment, total (% of total labor force) (modeled ILO estimate) - Czech Republic, International Labour Organization, ILOSTAT database [online]. Retrieved in 23 April 2019. Available from: `https://data.worldbank.org/indicator/SL.UEM.TOTL.ZS?end=2018&locations=CZ&start=1991&view=chart`

[2] Garnett, Ofer and Mandelbaum, Avishai and Reiman, Martin. Designing a call center with impatient customers. *Manufacturing & Service Operations Management*, volume 4, no. 3, 2002: pp. 208–227.

[3] Stephen Temple. Who created GSM? [online] Retrieved 23 April 2019. Available from: `http://www.gsmhistory.com/who_created-gsm/`

[4] SpeechTech, s.r.o. SpeechTech, Founded: 2010. Available from: `https://www.speechtech.cz`

[5] NEWTON Technologies, a.s. NEWTON Technologies, Founded: 2008. Available from: `https://www.newtontech.net/en/`

[6] Phonexia s.r.o. Phonexia Speech Platform, Founded: 2006. Available from: `https://www.phonexia.com`

[7] Apache Software Foundation. Solr, Founded: 2004. Available from: `http://lucene.apache.org/solr/`

[8] Elastic NV. Elasticsearch, Founded: February 8, 2010. Available from: `https://www.elastic.co`

[9] IBM. IBM Watson Explorer oneWEX, Released: 2018. Available from: `https://www.ibm.com/support/knowledgecenter/en/SS8NLW_12.0.0/com.ibm.watson.wex.ee.doc/explorer_onewex.html`

[10] SpeechTech, s. r. o. O společnosti SpeechTech, [online] Retrieved 23 April 2019. Available from: `https://www.speechtech.cz/#o-spolecnosti`

[11] SpeechTech, s. r. o. SpeechTech Text to Speech, [online] Retrieved 23 April 2019. Available from: `https://www.speechtech.cz/speechtech-text-to-speech/`

[12] SpeechTech, s. r. o. SpeechTech analytics, [online] Retrieved 23 April 2019. Available from: `https://www.speechtech.cz/speechtech-analytics/`

[13] NEWTON Technologies, a.s. About our company — NEWTON Technologies, [online] Retrieved 23 April 2019. Available from: `https://www.newtontech.net/en/about-us/`

[14] NEWTON Technologies, a.s. NEWTON Dictate, [online] Retrieved 23 April 2019. Available from: `https://www.newtontech.net/en/newton-dictate/`

[15] NEWTON Technologies, a.s. NEWTON Analytics, [online] Retrieved 23 April 2019. Available from: `https://www.newtontech.net/en/newton-analytics/`

[16] NEWTON Technologies, a.s. Languages supported, [online] Retrieved 23 April 2019. Available from: `https://www.newtontech.net/en/languages/`

[17] Roman Polok. Speech transcription [online] 2018-04-08. Available from: `https://partner.phonexia.com/phonexia-academy/product-training/speech-transcription/`

[18] Martinez, D.; Plchot, O.; et al. Language recognition in ivectors space. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[19] Reynolds, D. Gaussian mixture models. *Encyclopedia of biometrics*, 2015: pp. 827–832.

[20] Peggy Carlaw. Customer Service by the Numbers: Average Call Duration [online] June 9, 2010. Available from: `https://customerthink.com/customer_service_by_the_numbers_average_call_duration/`

[21] George Roter. Sharing our Common Voices – Mozilla releases the largest to-date public domain transcribed voice dataset [online] 2018-04-08. Available from: `https://blog.mozilla.org/blog/2019/02/28/sharing-our-common-voices-mozilla-releases-the-largest-to-date-public-domain-transcribed-voice-dataset/`

[22] Brown, P. F.; Desouza, P. V.; et al. Class-based n-gram models of natural language. *Computational linguistics*, volume 18, no. 4, 1992: pp. 467–479.

[23] Ramaswamy, Ganesh N and Printz, Harry W and Gopalakrishnan, Ponani S. Apparatus and method for building domain-specific language models. Feb. 13 2001, uS Patent 6,188,976.

[24] Schryen, G. Security of Open Source and Closed Source Software: An Empirical Comparison of Published Vulnerabilities. 01 2009, p. 387.

[25] Apache Software Foundation. Lucene, Founded: 1999. Available from: `https://lucene.apache.org/`

[26] Apache Software Foundation. Apache Lucene - Query Parser Syntax, [online] Retrieved 23 April 2019. Available from: `https://lucene.apache.org/core/2_9_4/queryparsersyntax.html`

[27] solid IT gmbh. DB-Engines Ranking of Search Engines, [online] Retrieved 23 April 2019. Available from: `https://db-engines.com/en/ranking/search+engine`

[28] IBM. Product and system architecture overview for Watson Explorer oneWEX on IBM Cloud Private, [online] Retrieved 23 April 2019. Available from: `https://www.ibm.com/support/knowledgecenter/en/SS8NLW_12.0.0/com.ibm.watson.wex.ee.doc/c_arch_onewex.html`

[29] IBM. BM LanguageWare Resource Workbench, [online] Retrieved 23 April 2019. Available from: `https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=6adead21-9991-44f6-bdbb-baf0d2e8a673`

[30] Apache Software Foundation. Welcome to the Apache UIMA project, Founded: 2006. Available from: `https://uima.apache.org`

[31] Cogniware, s.r.o. Getting started with CWDC, [online] Retrieved 25 April 2019. Available from: `http://docs.cogniware.com/cogniware-data-collector/getting-started-with-cwdc/`

[32] Phonexia s.r.o. Phonexia Speech Engine API Documentation [online] 2018-04-08. Available from: `https://download.phonexia.com/docs/spe/#Authentication`

[33] Jamie L. Using Token-Based Authentication to Improve Your Website [online] November 23, 2017. Available from: `https://swoopnow.com/token-based-authentication/`

[34] Haupt, Florian and Leymann, Frank and Pautasso, Cesare. A conversation based approach for modeling REST APIs. In *2015 12th Working IEEE/IFIP Conference on Software Architecture*, IEEE, 2015, pp. 165–174.

[35] Fette, Ian and Melnikov, Alexey. The websocket protocol. Technical report, 2011.

[36] Jin, Brenda and Sahni, Saurabh and Shevat, Amir. *Designing Web APIs: Building APIs that Developers Love.* " O'Reilly Media, Inc.", 2018.

[37] VMware, Inc. VMware, Inc., Founded: October 26, 1998. Available from: `http://www.vmware.com/`

[38] SmartBear Software. Swagger, Founded: 2011. Available from: `https://swagger.io`

Appendix A

Acronyms

AI Artificial Intelligence

API Application programming interface

BFS Breadth-first search

CPU Central processing unit

CRM Customer-relationship management

CSV Comma-separated values

CWDC Cogniware DataCollector

GDPR General Data Protection Regulation

GMM Gaussian Mixture Model

GSM Global System for Mobile communications

GUI Graphical user interface

HW Hardware

IBM International Business Machines

ID Identifier

IP Internet Protocol

JSON JavaScript Object Notation

LID Language identification

ML Machine Learning

MRCP Media Resource Control Protocol

NLP Natural language processing

NLQ Natural language query

REGEX Regular expression

REST Representational State Transfer

SSH Secure Shell

STT Speech to Text

TV Television

UIMA Unstructured Information Management applications

URL Uniform Resource Locator

USB Universal Serial Bus

VOIP Voice over IP

WAV/WAVE Waveform Audio File Format

WEX Watson Explorer

XML Extensible markup language

Appendix B

Supplemental Material

The complete source code of the thesis, snippets of the codes which were described within thesis and the photos of the final prototype GUI are attached in the medium alongide with the thesis.

```
│ README.md ................................. brief description of contents
├── BT_Pazdera_Boris_2019.pdf .................. thesis in the PDF format
├── src ...................................... the directory of source codes
│   ├── datahandler ................ snippets of CWDC's DataHandler codes
│   │   └── phonexia .... implementation of PhonexiaHelper class with models
│   └── connector .................... snippets of CWDC's Connector codes
├── thesis ..................................... source of thesis in LaTeX
└── photos ........................................... photos of the GUI
```