TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Siim Talts 176965IAPM

ANALYSING ELECTION CANDIDATE EXPOSURE IN BROADCAST MEDIA USING WEAKLY SUPERVISED TRAINING

Master's thesis

Supervisor: Tanel Alumäe
PhD
Senior Researcher

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Siim Talts 176965IAPM

VALIMISKANDIDAATIDE RAADIOS JA TELEVISIOONIS EKSPONEERITUSE ANALÜÜS KASUTADES KAUDSE JUHENDAMISEGA TREENIMIST

Magistritöö

Juhendaja: Tanel Alumäe
Doktorikraad
Vanemteadur

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Siim Talts

07.05.2019

Abstract

This thesis studies the possibility to use weakly supervised training in a more complex and real world scenario for identifying speakers. Previously this kind of task has been done only by using pre-validated training data from a single source. In this work the data is aggregated automatically from multiple sources and it might contain impure samples. The setting for the experiment is the 2019 Estonian Parliament elections and speaker models are created for the election candidates. With weakly supervised speaker identification the candidates exposure to broadcast media is analysed.

The conducted experiment proves that it is possible to train speaker identification models by automatically aggregating data from different sources. For the current attempt data from YouTube and Estonian Public Broadcasting archive was used for training the models and for experiments a hand picked set of debates, which did not occur in the training set, were used. The observations did not directly indicate towards the improvement of election result by having more exposure to broadcast media. The experiment shows that it is possible to use this kind of method to analyse such public events.

This thesis is written in English and is 64 pages long, including 8 chapters, 24 figures, 8 tables and 6 equations.

Annotatsioon

Valimiskandidaatide raadios ja televisioonis eksponeerituse analüüs kasutades kaudse juhendamisega treenimist

Käesoleva magistritöö eesmärgiks on valideerida kaudse juhendamisega kõnelejatuvastamise meetodit. Antud töös viiakse selle tarbeks läbi eksperiment milles modelleeritakse 2019. aasta Riigikogu valimiste kandidaatide kõnemudelid. Kui varasemate eksperimentide puhul on kasutatud süvanärvivõrgu treenimiseks hea kvaliteediga andmeid mis pärinevad ühest allikast, siis antud töö puhul kasutatakse paralleelselt mitut allikat, mis võivad sisaldada ebatäpseid ning valepositiivseid kandeid. Selle jaoks luuakse tarkvara lahendus, mis võimaldab soovitud isikutele automaatselt koguda treenimisandmeid mitmetest allikatest mis sisaldavad selle isiku kõnet. Mudelite valideerimiseks kasutatakse valimisdebatte, mida ei esinenud treeningandmetes. Need annoteeritakse käsitsi ja võrreldakse tulemustega mida kaudse juhendamisega treenitud süvanärvivõrk tagastab.

Peale kõnelejamudelite treenimise uurib töö ka võimalusi kuidas oleks võimalik antud meetodit kasutades analüüsida suuri avalike üritusi. Selleks viidi läbi eksperiment valimiskandidaatidega, kus vaadeldi raadios ja televisioonis toimuvaid valimisdebatte ning koguti andmeid kandidaatide kõnelemissageduse ja kestuse kohta. Vaatluste läbiviimiseks kasutati eelnevalt kaudse juhendamisega treenitud süvanärvivõrku. Lõpuks summeeriti vaatluse tulemused ning võrreldi neid reaalsete valimistulemustega.

Töös läbiviidud eksperiment kinnitab, et mitmest allikast automaatselt kogutud treeningandmete komplektiga on võimalik edukalt treenida kõnelejatuvastuse mudelid. Samuti sobib selline meetod avalike ürituste analüüsimiseks. Antud juhul ei tuvastatud otsest mõju kandidaatide raadios ja televisioonis eksponeerituse ning valimistulemuste vahel.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 64 leheküljel, 8 peatükki, 24 joonist, 8 tabelit ja 6 valemit.

List of abbreviations and terms

DNN	Deep neural network
UBM	Universal Background Model
GMM	Gaussian Mixture Models
HMM	Hidden Markov model
DNN	Deep neural network
ANN	Artificial neural network
DL	Deep learning
AI	Artificial Intelligent
ML	Machine learning
URI	Uniform Resource Identifier
CSV	Comma-separated values
SDK	Software development kit
ERR	Estonian Public Broadcasting
API	Application programming interface
ReLU	Rectified Linear Units

Table of Contents

List of Figures

List of Tables

List of Equations

1 Introduction

This thesis investigates the possibility to use methods of weakly supervised machine learning for more broader situations, which in this case is a public election. The data for making these evaluations is gathered using an automated tool which scans Estonian Public Broadcasting archive[1] and the popular public media platform YouTube[2]. Afterwards using the gathered data, speaker identification models are trained using methods of weakly supervised machine learning. The setting for the observation is the 2019 Estonian Parliament elections and the subjects are all the candidates who are enlisted.

This chapter will describe the approaches and methods used for accomplishing the goals and will set targets for the results.

1.1 Problem

The thesis investigates two topics. Firstly, the possibility to use effectively machine learning with large sets of weakly labelled and impure data. It has been proven that using weakly labelled data is sufficient for training speaker identification models [8]. Current scenario differs from previous approaches by combining multiple sources for gathering training dataset. Also the setting itself is more specific and for a real-world use-case as the previously done experiments did not serve a purpose besides acting as technical demos for the speaker identification method.

The second topic is a social one – the effects of using broadcast media as the means of communications on public election results. Media plays a huge role for each party's election campaign. Most of the voters get their information through different media channels and because of that, it could be inferred that having greater media exposure is

1 https://arhiiv.err.ee/
2 https://www.youtube.com/

necessary for good results [16]. The area that is not well documented are the individual results of the candidates. Up until now, it can only be assumed that having more outreach through media will have a positive effect on the results. In the recent years much of the work in this field has shifted towards making observations on new media [36] [37]. In the scope of this work, exposure to broadcast media is observed.

1.2 Goals

The first objective of the thesis is to train speaker models for as many candidates as it is possible. After that, the models are validated by calculating recall – the percentage of labelled recordings – and precision – the percentage of the previously labelled recordings which were correct. The goals for these metrics are 70% on recall and 90% on precision. For validation a dataset consisting of real election debates is used, where the candidates are speaking.

The second task is to use the trained models to observe the election campaign and to analyse the exposure of candidates and parties to broadcast media. For that the actual election results will be compared with summarized results from speaker identification. The goal is to see if it is possible to use this kind of machine learning model to analyse such public event.

1.3 Methodology

Firstly an automatic data aggregator is designed by following object oriented software development patterns. That tool will find multiple recordings where a person in interest is speaking and downloads them. Besides the audio recordings, the aggregator also needs to compose metadata for each training sample, which are required for training the speaker models.

Secondly the deep neural network is trained using methods of weak supervision. Validation is done by manually annotating the validation dataset and comparing it with the speaker labels that the trained model returns.

1.4 Outline

The first chapter gives introduction to the thesis. The specific problem is defined and also the metrics and their goals, which are used to evaluate the results are given.

The second chapter gives an overview of the background theory for speech recognition. All the main components – speech diarization, i-Vectors, deep neural networks, weakly labelled training – are covered with the amount that is necessary to understand the speaker identification system.

The third chapter analyses the primary related work of the thesis. Although all of them are connected with the current thesis, none of them deal directly with the problem that is stated in section 1.1.

The fourth chapter describes and gives requirements for the data that is needed for training the speaker models. The data in simplified form consist of two parts – audio recordings of the speakers and the metadata related to them. Also the design and implementation of an automatic data aggregator is given. Lastly, a brief analysis of the gathered data is provided.

The fifth chapter goes into depth of describing the training and validation of the created speaker identification models. Algorithms for both training and later identifying speakers are given. The results of the validation are also revealed.

The sixth chapter describes the experiment which was conducted during this thesis. Detailed descriptions of both the input data and the results are given. This chapter also contains the analysis of the experiment, which is needed for the goals set in section 1.2.

The seventh chapter analyses the created system and proposes alternative use cases for it. Recommendations for future work are also given.

The eight chapter concludes the work and states whether the goals were achieved.

2 Background Theory

This chapter gives an overview of what a state of art speaker identification consists of and details the methods and technologies used in this thesis.

2.1 Speaker recognition

Biometric recognition systems like speaker recognition are used in various fields, like security systems and digital personal assistants like Amazon's Alexa[1]. The general area of speaker recognition is divided into two: identification and verification [1]. With speaker identification the goal is to determine which of the known voices matches the input sample. This is also known as *closed-set* speaker identification. With verification, the task is to determine if the sample voice is who it claims to be. This is also known as an *open-set* problem. The combination of closed-set identification and open-set verification is called closed-set verification. In this case, the system must identify known speakers, but must also be able to classify speakers unknown to the system under the "unknown" category. This thesis focuses on closed-set verification tasks.

2.2 Speaker recognition algorithms

2.2.1 Gaussian Mixture Models (GMM)

A Gaussian Mixture Model (GMM) is a function that is represented as a weighted sum of Gaussian component densities [4]. GMMs are widely used for probability distribution for continuous measurements. One of those are speaker recognition systems, where GMMs are used for representing speech vocal properties, because GMMs have the ability to represent large class of sample distributions.

1 https://developer.amazon.com/alexa

2.2.2 Universal Background Model (UBM)

Universal Background Models (UBM) are often used in biometric verification systems, like speaker recognition system [5]. UBMs are used for representing person-independent biometric features that could be used for comparing against person-specific features that are used for making boolean decisions. In speaker verification an unknown speech sample could be classified, if a speaker-specific GMM has been trained. That test will give a ratio score between the speaker-specific GMM and speaker-independent UBM. If the specified threshold has been reached, the unknown sample will be classified.

2.2.3 Hidden Markov model (HMM)

Hidden Markov model (HMM) is a widely used statistical technique in machine learning (ML) [6]. HMMs can be used to model many different flows such as speech and proteins. HMM displays a probability distribution over a sequence of observations. It does that by invoking a separate sequence of unobserved or hidden states. The hidden states have Markov dynamics and the observed states are all independent from all other variables.

2.3 i-Vectors

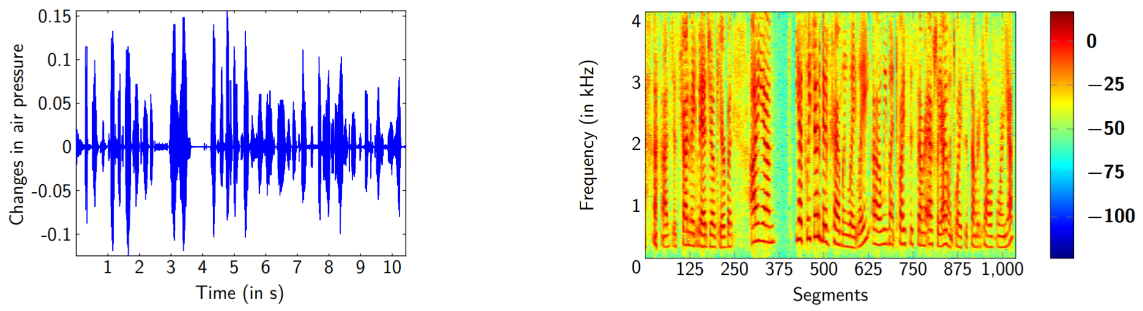


Figure 1: Speech sample after segment-wise Fourier transformation [2]

The characteristics about a speaker’s voice can be obtained by extracting speech signal features [2]. Features are dependent on the speaker and the text which was spoken. Speech signals are extracted by feeding speech segments to a system which uses Fourier transformation and the extraction of Mel-Frequency Cepstrum Coefficients (Figure 1).

Different speakers are identified by comparing resulting patterns with each other. Those speech signal features are called identity-vectors or i-Vectors and this approach is used in state-of-the-art speaker recognition systems.

An i-Vector extractor is a system that gathers sequence of vectors from speech utterances and maps them to a fixed dimensional vector [3]. It is achieved by denoting a K-component GMM as UBM and collecting statistics from the speech utterances.

The process starts with feature extraction from utterances and for this approach no distinction is made between speaker and channel features [38]. Before calculating the i-Vector, we need to first find the UBM supervector. This is demonstrated on Equation 1 where M is the supervector, m is the speaker- and channel-independent supervector and V, U and D define the speaker and session subspaces.

$$M = m + V + U + D$$

Equation 1: UBM supervector

The i-Vector is extracted by using Equation 2, where w is the resulting i-Vector and T is a rectangular matrix of low rank. Both parameters m and M are taken from the UBM supervector. The size of T determines the size of the resulting i-Vector.

$$w = \frac{M - m}{T}$$

Equation 2: i-Vector extraction

For each speech utterance, an i-Vector is obtained and by using a large collection of data, an exhaustive i-Vector for each speaker can be computed. In the paper “Front-End Factor Analysis For Speaker Verification” the authors experimented with i-Vectors which had 100 to 400 dimensions [38]. In that scenario generally higher dimensional i-Vectors produced better results.

Extracted i-Vectors are used in both supervised and unsupervised learning and the vectors are fed directly to the deep neural network (DNN). The DNN used in this thesis is trained with i-Vectors that have 600 dimensions.

2.4 Speaker diarization

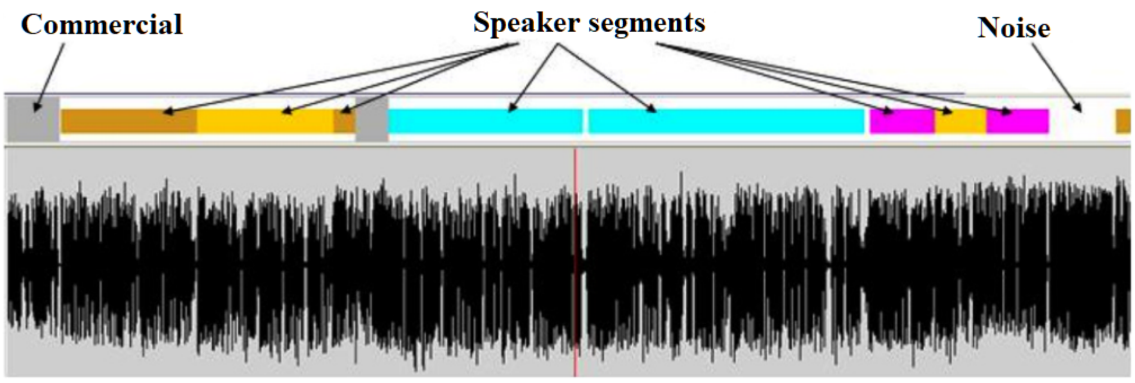


Figure 2: Example of audio diarization on broadcast news [7]

A necessary step in speaker identification is clustering the input audio source into speech segments. Generally an audio source – a news broadcast or an interview – consists of multiple speakers who speak in different times and maybe even in different conditions. Also the footage might contain some music, commercials or some general background noise – see Figure 2. All of those categories are not in the focus the of a general speaker identification system.

For segmenting audio sources by speakers a technique called speaker diarization is used [7]. In its simplest form only speech and non-speech segments are created. In this case non-speech contains all music, silence, commercials and all the different background noises. In a more detailed diarization all speaker changes would be marked and all the different speaker segments coming from the same speaker would be categorised under the same tag. This is also referred as "who spoke when" or speaker segmentation.

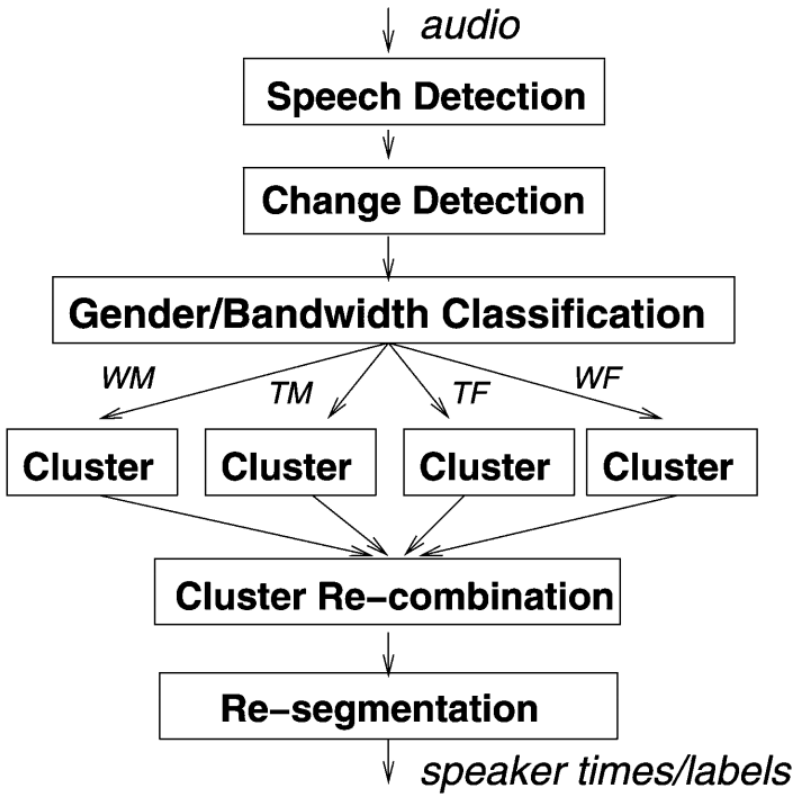


Figure 3 Prototypical diarization system subtasks and the flow between them [7]

A general speaker diarization task has a set of 6 subtasks (Figure 3):

1. Speech detection – the goal of this step is to detect the regions of speech in the audio source. Generally for these tasks a maximum-likelihood GMM is used, which is trained on labelled data.

2. Change detection – the goal of this step is to detect the points in the audio source where most likely the speaker has changed. For this the two adjacent frames of data are compared to each other and the distance metric between them is calculated. To achieve this either Bayesian information criterion technique or Gaussian method is used.

3. Gender and bandwidth detection – the goal of this step is to group the speaker segments by gender and by bandwidth. The reason for this step is to reduce the load of the clustering step. Similarly to the first step, also maximum-likelihood GMMs are used for this.

4. Clustering – the goal of this step is to group speech segments coming from the same speaker together. Ideally one cluster per speaker is produced, but this might not be achieved if the same speaker has speech segments in different conditions, for example in studio and in the street.

5. Re-combination of clusters – the goal of this step is to refine the original clustering outcome and combine similar clusters into one. To do this, clustering is run to under-cluster the audio, generally with clusters with speech over 30 seconds. Different clusters are compared with each other and if a predefined threshold is reached, the clusters are combined. With this the speakers who perform in different background conditions could be grouped under the same tag.

6. Re-segmentation – the goal of this step is to refine the original cluster boundaries and also re-organize short segments that might have been removed in the first step.

These steps could be performed in sequence as shown in Figure 3, but some of the steps could be performed at the same time. For example speech and gender detection could be done in parallel.

With the decreasing cost and increasing performance of general computational devices, speaker diarization can be done in more and more general areas and even in real-time tasks.

2.5 Deep neural networks

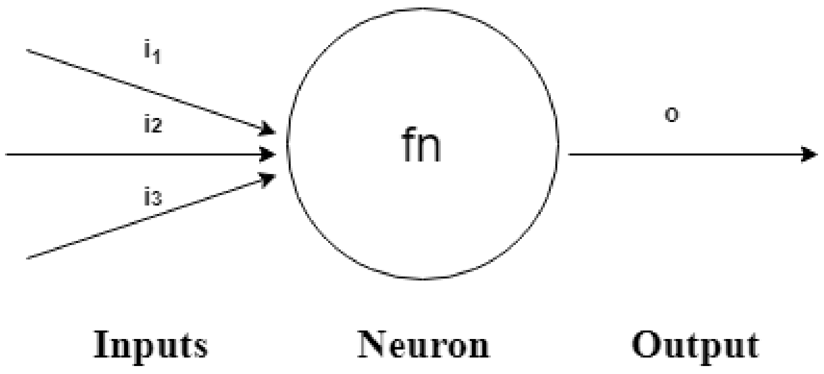


Figure 4: Simplified model of an artificial neuron

A deep neural network (DNN) is a subset of artificial neural networks, with more complexity and multiple special layers called hidden layers between the input and output layers [10]. A regular neural network consists of many interconnected nodes called neurons. Neurons in nature are simple – they have usually several inputs, an activation function and they produce a real-valued output upon activation (Figure 4). The neurons on the input layer get activated by data from sensors, other neurons may be activated by being connected to previously activated neurons. Each of those activation functions has a specific weight to it which determines when the neuron gets activated. Learning in neural networks is all about fine-tuning those weights for accomplishing complicated tasks. Depending on the task, learning requires several iterations and might take a long time to finish. Deep learning (DL) aims to optimize learning across these iterations.

Before the popularity of DL, mostly shallow ANNs were used which usally just consisted of at most two layers of non-linear feature transformations [11]. Examples of those architectures are regular GMMs, maximum entropy models and support vector machines. Shallow architectures have shown good performance with simple and well-constructed tasks, but they struggle with more complicated problems like speech recognition and computer vision. Deep learning focuses on learning complex feature hierarchies where higher level features are formed by lower features [9]. Having a hierarchy means that when improving a lower level sub feature, the knowledge automatically propagates to the connected knowledge. This means that learning does not need specific raw sensory data, but it could be done through abstractions and more complex models could be trained.

The depth in DNNs refers to the number of layers that the architecture consists that are used for learning. As shallow ANNs only consist of a few layers, DNNs could be designed to have many more. Although some of the methods for DNNs were researched and developed in the 1960s, they were not used upon recently, due to worse results than traditional ANNs and the lack of computational power. A breakthrough happened in 2006 when the Deep Belief Network was introduced [12]. It used a greedy learning algorithm, that trained one layer at the time using unsupervised learning techniques for each layer. Shortly after that DNNs were introduced for different classification, natural language processing and robotics tasks. The increased research in ML and the drastically improved general-purpose graphical processing units performance has paved the way for the popularity for DL [11].

The main qualities of DNNs are [9]:

1. Ability to learn complex, highly-varying tasks where the number of different outcomes is greatly larger than the number of training examples.
2. Ability to learn high-level abstractions from low level features.
3. Ability to learn from a large set of training data.
4. Ability to learn from mostly unlabelled data.
5. Ability to use multi-task learning techniques.
6. Ability to perform well in unsupervised learning.

By task DNNs could be divided into three categories [11]:

1. Deep networks for unsupervised or generative learning where most of the learning data is unlabelled.
2. Deep networks for supervised learning where the data has always labels.

3. Hybrid deep networks which tries to combine both unsupervised and supervised learning.

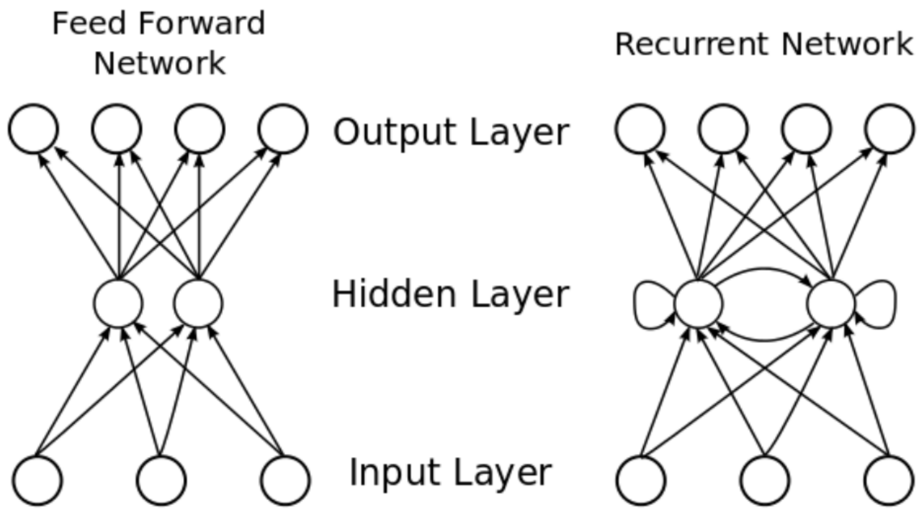


Figure 5: Feed-forward and recurrent neural networks [13]

By architectural design DNNs could be divided into two general categories – feed-forward and recurrent neural networks [13]. Recurrent networks have recurrent connections between the nodes, which creates more inputs that is required for more sequential or time-series data. Feed-forward networks are used commonly for classification or regression tasks, that use a single input. Mostly back-propagation is used with feed-forward networks. At first the network is initialized with random weights in neurons and depending on the task it is adjusted after each training cycle.

2.6 Weakly supervised training for speaker identification

Traditionally speaker identification models are trained with data where all the speech segments are annotated [17]. This could be done by hand using a computer program called Transcriber[1] as seen in Figure 6, but it is very time consuming. One alternative method to that is to identify speakers from speech transcripts. In this method the speech is automatically transcribed and name entities are fetched from that and later matched with corresponding speech segments. Another way, if working with for example TV broadcasts, is to search name entities from the video. Both of these methods could be used, but with introducing another layer of abstraction to the speaker model training.

1 http://trans.sourceforge.net

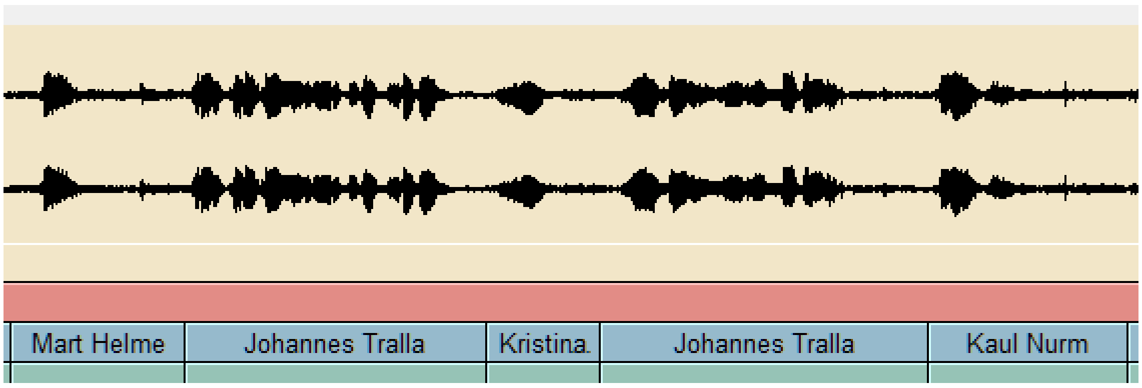


Figure 6: Annotating speakers in Transcriber

Luckily there is a large collection of audio and video materials available online, which could be used for training. These documents often come with a lot of metadata attached to them, one of those properties might be the speakers name. In this case where document has names attached to them, speaker models could be trained using weakly supervised method.

The method works by using recording level labels, instead of traditional segment level labels. If this kind of data is available, the technique is executed in three steps:

1. Firstly, speaker diarization is applied to the training data. This will remove all non-speech parts and also clusters the data, so segments coming from the same speaker would be in the same group.
2. Secondly, i-Vectors are computed for each speaker.
3. Lastly, i-Vectors are used to train feed-forward DNN.

A thing to be noted is that not all speakers in the recording have to be mentioned. The DNN is trained using an objective function at the recording level, because the recording metadata is used for predicting similar i-Vectors. The average distribution over speakers for each recording is calculated using formula described in Equation 3.

$$\tilde{p}_n(y_i)=\left\{\frac{1}{|X_n|}, \text{if speaker is in metadata, otherwise } 0\right\}$$

Equation 3: Average distribution for speakers [17]

The DNNs cost function is the Kullback-Leibler divergence between the previously found expected average distribution and model's expected average conditional distribution (Equation 4).

$$l = \sum_{y} D\left(\tilde{p}_n \| p_{0_s}\right)$$

Equation 4: DNN cost function

All the i-Vectors which are not modelled, will be added under the same label called *unk.* This method has shown a result of up too 95% recall with 98% precision. It was found that in order to identify confidently speakers with models created with this method, about 15 appearances in the training data was required, as shown in Figure 7.

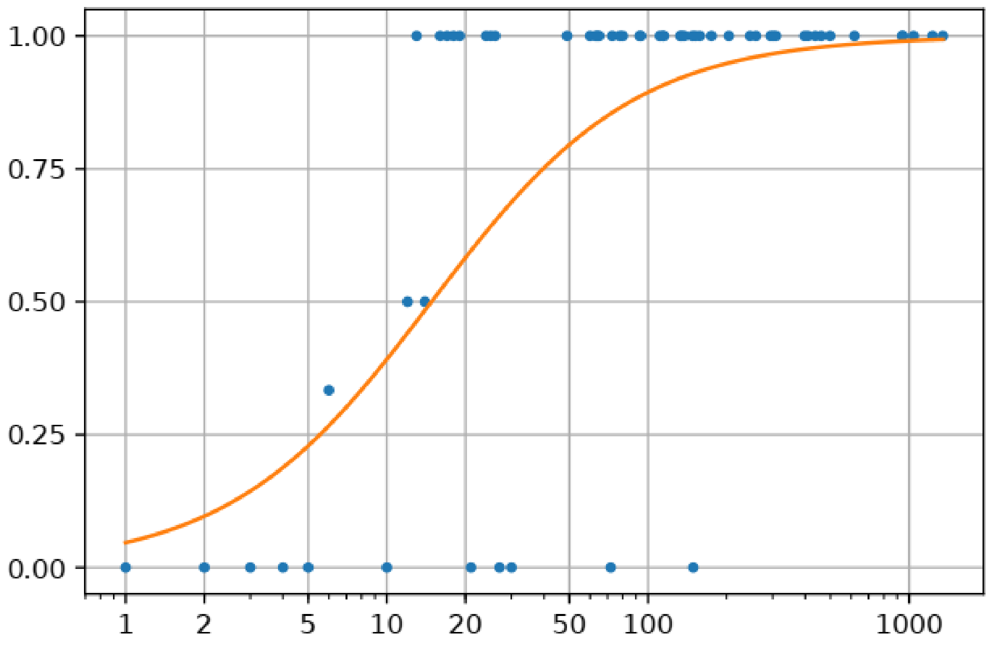


Figure 7: Number of speaker occurrences in training data [8]

DNN trained with weakly supervised method is used in this thesis.

3 Related work

The problem of using weak supervision to create speaker models for election candidates and to analyse their exposure in broadcast media has not been investigated to the best of the Author's knowledge. Lately the focus has been on the affects of social media for the elections while broadcast media has been on the side line [36] [37].

This chapter gives a brief overview of a few studies that are related to this thesis in some way or another.

3.1 Weakly supervised training of speaker models from VoxCeleb dataset

VoxCeleb is an audio-visual dataset, which contains many short clips of human speech, that are extracted from interview videos uploaded to YouTube [14]. The dataset has speakers from a wide range of different accents and ages. In [14] the dataset was used to create speaker models with the help of an existing face recognition dataset by identifying the speaker in the video. By that the speech segment could be extracted which was used for training the speaker models. This method showed about 80% accuracy.

The main drawbacks of that method were that there needed to be an existing face recognition dataset and speakers whose face didn't appear weren't modelled. In the paper by Karu and Alumäe, using methods of weakly supervised training yielded better results with about 95% accuracy compared to previous 80% [8].

The work by Karu and Alumäe shows that using weakly supervised training when recording level annotations are available, will result better accuracy than similar automated methods. This thesis uses the similar weakly supervised method for training speaker models.

3.2 Large-scale speaker identification

Large-scale speaker identification deals with the problem with vast amounts of data where the speaker identification must work. In this paper, data from YouTube[1] is in the focus and the authors propose a method for speeding up the search of speakers [15]. As mostly the data in YouTube is unlabelled the method works in an unsupervised setting.

The proposed method combines i-Vector based approach with locality sensitive hashing, an algorithm for fast nearest neighbour search in high dimensions. The results show a drastic improvement in speed while only sacrificing a minor fraction in precision. For example for 20 second utterances, the proposed algorithm worked 150 times faster, while being about 95% as precise.

Although this method was tested in a controlled unsupervised environment, it shows that data on YouTube's platform is with a necessary quality to train precise speaker identification models. This thesis uses data from YouTube's archives and combines it with other sources to train speaker models.

3.3 Leveling the playing field: How campaign advertising can help non-dominant parties

In this paper, the authors set an hypothesis, that political advertising affects mainly uninformed voters and has bigger affect for the non-dominant party [16]. The authors conducts observations on the 2009 and 2012 Mexico's federal legislative elections, mainly focusing on radio advertisements. The results showed clear increase of the non-dominant party's results, when focusing more on radio advertisements.

This study focuses on the same broader objective if this thesis – finding the correlation between media appearances and election results. Although this thesis focuses on the individual results, not the party's, the methods for making these assessments could be transferred.

1 https://www.youtube.com/

4 Input data

This chapter describes the techniques how the data was gathered to perform the speaker identification experiment with 2019 Estonian Parliament candidates. The data consists of both the audio recordings and the metadata related to them.

4.1 Audio files

One of the most important parts of ML tasks is the collection of good quality training data samples. It is particularly important for supervised ML tasks, because the accuracy of the resulting model depends on the training data quality [18]. Although for weakly supervised ML tasks the data does not have to be as pure as for supervised tasks, the quality and variations of it will play a big role in the results.

For training the models, the essential input data is the audio sample, where the person in interest speaks. The whole task starts with collecting those samples. As the training is language independent, it does not matter what the spoken words mean. This widens our possible sample size considerably.

4.2 Data requirements

The goal for data aggregation in this thesis is to gather a good sample size for 1084 speakers. The complete list of the speakers for whom the data aggregation was done is available in “Appendix 1 – List of 2019 Estonian Parliament election candidates”.

For successfully training a speaker model with good quality, we would need at least 10 training samples for each speaker, as it was demonstrated in the experiment described in chapter 2.6. Requirements for the training samples are the following:

- Metadata for each data sample with the following parameters:

    - Speaker’s name
    - Name of the training sample
    - Name of the origin of the training sample
    - Training sample’s Uniform Resource Identifier (URI)
    - Name of the downloaded file

- Wav audio sample with a unique identifier as the name. This way if multiple sources point towards the same data sample, then they are merged. The file name must correspond to the same file name in the previous metadata requirement. The wav format requirements grants, that the audio is with source quality, since wav does not apply any compression [19].

All the entries must be saved into a Comma-separated values (CSV) file and the audio samples need to be downloaded to the drive.

4.3 Data aggregator

4.3.1 Requirements

In order to gather the data that is needed to train the speaker models, an aggregator must be developed. As the ML task in this thesis is essentially a variation of a multi-instance multi-label learning problem, the learning phase needs to cycle through a lot of data [20]. In order to do that, the aggregator should compile training data from several different sources. Therefore the following functional requirements are set for the application:

- Must be able to gather data from several sources.
- Must be able to gather data for a collection of speakers.
- Adding new sources must be possible.

The following non-functional requirements are also set:

- Must generate output by following the rules set in paragraph 4.2.
- Must be executable from a Linux command line.

4.3.2 Design

In order to comply with the set requirements, the author decided to follow object oriented programming patterns, as it provides more flexibility for future improvements and allows reusing the code [21]. The main software design principles that were followed are polymorphism and factory method patterns. The describing class diagram of the developed application is shown on Figure 8.

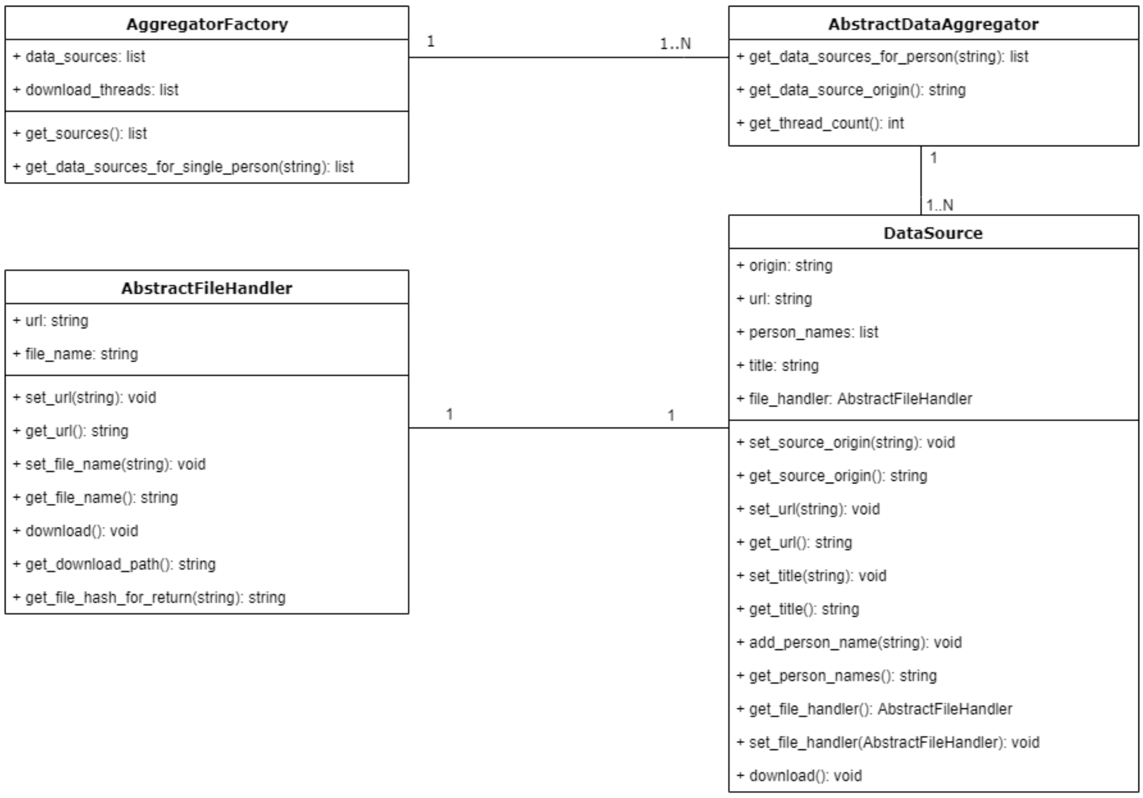

AggregatorFactory
+ data_sources: list
+ download_threads: list
+ get_sources(): list
+ get_data_sources_for_single_person(string): list
1
1..N
AbstractDataAggregator
+ get_data_sources_for_person(string): list
+ get_data_source_origin(): string
+ get_thread_count(): int
1
1..N
DataSource
+ origin: string
+ url: string
+ person_names: list
+ title: string
+ file_handler: AbstractFileHandler
+ set_source_origin(string): void
+ get_source_origin(): string
+ set_url(string): void
+ get_url(): string
+ set_title(string): void
+ get_title(): string
+ add_person_name(string): void
+ get_person_names(): string
+ get_file_handler(): AbstractFileHandler
+ set_file_handler(AbstractFileHandler): void
+ download(): void
AbstractFileHandler
+ url: string
+ file_name: string
+ set_url(string): void
+ get_url(): string
+ set_file_name(string): void
+ get_file_name(): string
+ download(): void
+ get_download_path(): string
+ get_file_hash_for_return(string): string
1
1

Figure 8: Data aggregation application’s class diagram

The interaction of different classes can be seen in Figure 9.

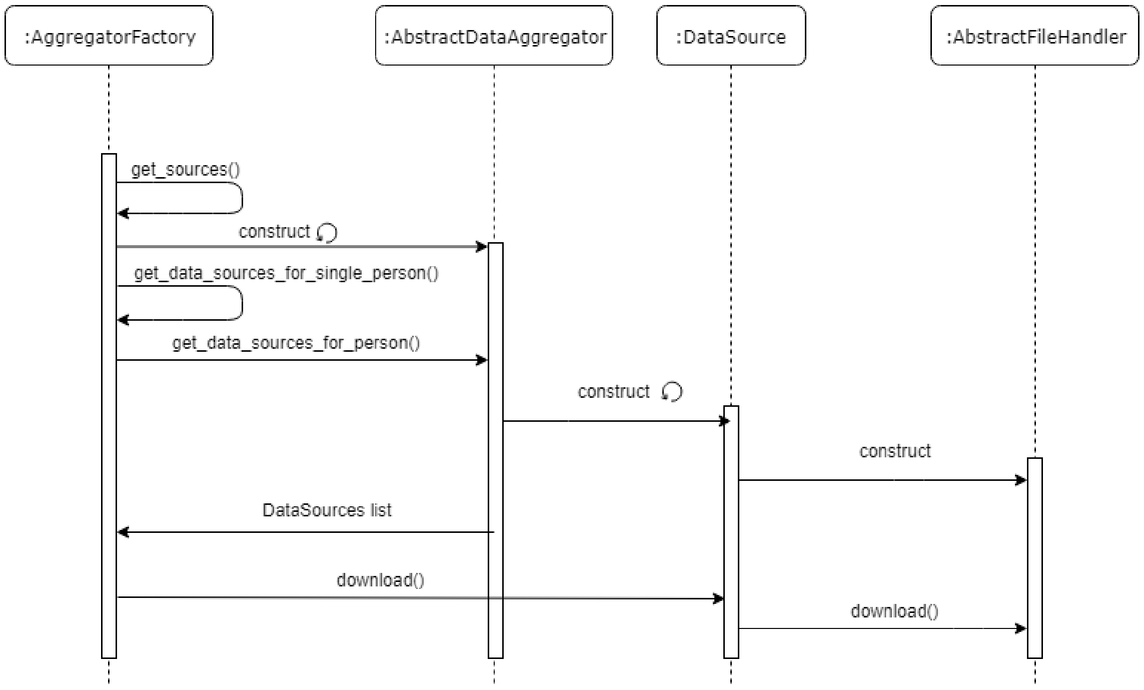


Figure 9: Data aggregation application's interaction diagram

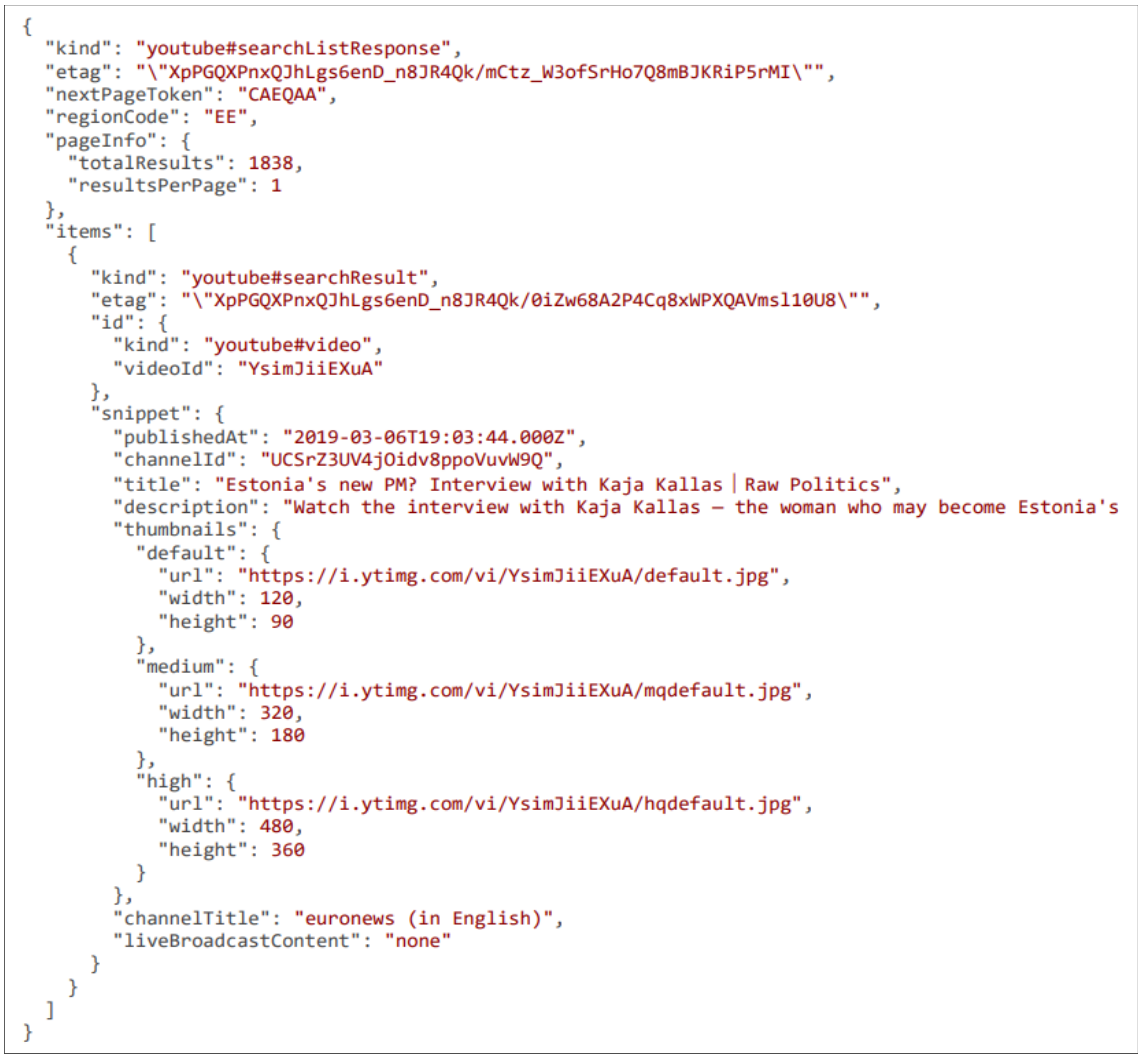
4.3.3 Developed application

The application was developed by following the requirements set in chapter 4.3.1 and by implementing the design proposed in chapter 4.3.2. Python 3.7[1] programming language was used for developing it. Python was chosen because the ease of use in a command line environment and because it supports all the main paradigms of object oriented programming [22]. The finished application can be downloaded from a public GitHub repository [23]. The basic user manual is also provided in the repository.

As one of the functional requirements was to have multiple data sources, two different sources were implemented – Estonian Public Broadcasting (ERR) archive[2] and YouTube[3]. Both of them needed to extend the `AbstractDataAggregator` class. Because the nature of the design, the aggregators need to perform a search step for each person in interest. With YouTube it was a trivial task, because of the available YouTube application programming interface's (API) software development kit (SDK) for Python[4]. With that, a search could be done, to find all the recordings for a person. In

1 https://www.python.org/downloads/
2 https://arhiiv.err.ee/
3 https://www.youtube.com/
4 https://developers.google.com/youtube/v3/quickstart/python

order to maintain data quality, the name of the person needed to be in either the recordings name or in the metadata. Example of the YouTube API's response is visible in Figure 10.



```
{
  "kind": "youtube#searchListResponse",
  "etag": "\"XpPGQXPnxQJhLgs6enD_n8JR4Qk/mCtz_W3ofSrHo7Q8mBJKRiP5rMI\"",
  "nextPageToken": "CAEQAA",
  "regionCode": "EE",
  "pageInfo": {
    "totalResults": 1838,
    "resultsPerPage": 1
  },
  "items": [
    {
      "kind": "youtube#searchResult",
      "etag": "\"XpPGQXPnxQJhLgs6enD_n8JR4Qk/0iZw68A2P4Cq8xWPXQAVmsl10U8\"",
      "id": {
        "kind": "youtube#video",
        "videoId": "YsimJiiEXuA"
      },
      "snippet": {
        "publishedAt": "2019-03-06T19:03:44.000Z",
        "channelId": "UCSrZ3UV4jOidv8ppoVuvW9Q",
        "title": "Estonia's new PM? Interview with Kaja Kallas | Raw Politics",
        "description": "Watch the interview with Kaja Kallas – the woman who may become Estonia's
        "thumbnails": {
          "default": {
            "url": "https://i.ytimg.com/vi/YsimJiiEXuA/default.jpg",
            "width": 120,
            "height": 90
          },
          "medium": {
            "url": "https://i.ytimg.com/vi/YsimJiiEXuA/mqdefault.jpg",
            "width": 320,
            "height": 180
          },
          "high": {
            "url": "https://i.ytimg.com/vi/YsimJiiEXuA/hqdefault.jpg",
            "width": 480,
            "height": 360
          }
        },
        "channelTitle": "euronews (in English)",
        "liveBroadcastContent": "none"
      }
    }
  ]
}
```

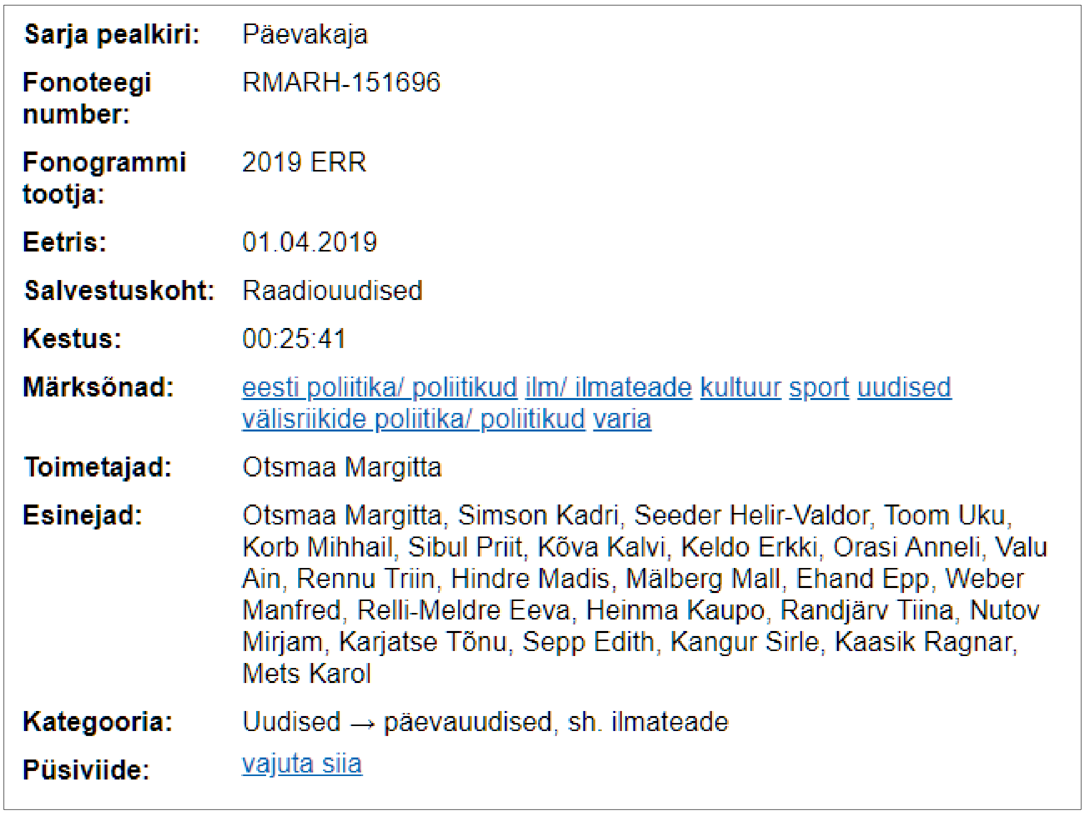
Figure 10: YouTube API example response

After that the source will be downloaded using pytube[1] which is lightweight, dependency-free Python library for downloading YouTube videos. For speeding up the process of downloading data, YouTube aggregation is ran in several threads at once.

As ERR archive does not have a public API, methods of web scraping must be applied [24]. Luckily doing this is rather simple with Python's Pandas[2] library, which is an

1 https://github.com/nficano/pytube
2 https://pandas.pydata.org/

open-source library for data analysis. ERR has provided sufficient metadata with all the recordings (Figure 11), so the data could be gathered for each speaker.



Sarja pealkiri:	Päevakaja
Fonoteegi number:	RMARH-151696
Fonogrammi tootja:	2019 ERR
Eetris:	01.04.2019
Salvestuskoht:	Raadiouudised
Kestus:	00:25:41
Märksõnad:	eesti poliitika/ poliitikud ilm/ ilmateade kultuur sport uudised välisriikide poliitika/ poliitikud varia
Toimetajad:	Otsmaa Margitta
Esinejad:	Otsmaa Margitta, Simson Kadri, Seeder Helir-Valdor, Toom Uku, Korb Mihhail, Sibul Priit, Kõva Kalvi, Keldo Erkki, Orasi Anneli, Valu Ain, Rennu Triin, Hindre Madis, Mälberg Mall, Ehand Epp, Weber Manfred, Relli-Meldre Eeva, Heinma Kaupo, Randjärv Tiina, Nutov Mirjam, Karjatse Tõnu, Sepp Edith, Kangur Sirle, Kaasik Ragnar, Mets Karol
Kategooria:	Uudised → päevauudised, sh. ilmateade
Püsiviide:	vajuta siia

Figure 11: Example of the ERR metadata

ERR aggregator firstly creates an indexed map of all the recordings by using metadata available. This is done by looping through a range of pages in the archive and collecting all the data that is presented in the metadata section of that page. The data range can be defined in the implementation of ERR's aggregator. This information is stored in a cache which is saved into CSV file. After the cache has been created, it can be used for searching all the archive entries for a single person and to download their training samples.

Currently only YouTube and ERR archive have been implemented as possible sources, but new ones could be added easily by extending the `AbstractDataAggregator` and the `AbstractFileHandler` classes. After that the newly created source should be added to `AggregatorFactory#get_sources` method.

4.4 Aggregation results

Table 1: Aggregation results

	Unique recordings	**Unique speakers**
ERR archive	6619	545
YouTube	4526	733
Total	11145	810

The overall results of the aggregation is visible in Table 1. Data aggregation application was ran for 1084 speakers which resulted in over 550 gigabytes of training data with 11145 unique data sources. The data was collected in January 2019. The goal was to create speaker models for all target speakers, but in reality not all of them had enough data for creating a speaker model which would give good results (Figure 7). As it is visible in Table 1, data was found for 810 persons out of 1084. This means, that with the current collection of speakers, we were able to aggregate training data for 75% of them (Figure 12). Furthermore, when setting a minimum of 10 recordings per person, then it resulted in 317 unique speakers, which meant that with this collection of speakers, we were able to create models for about 30% of them.

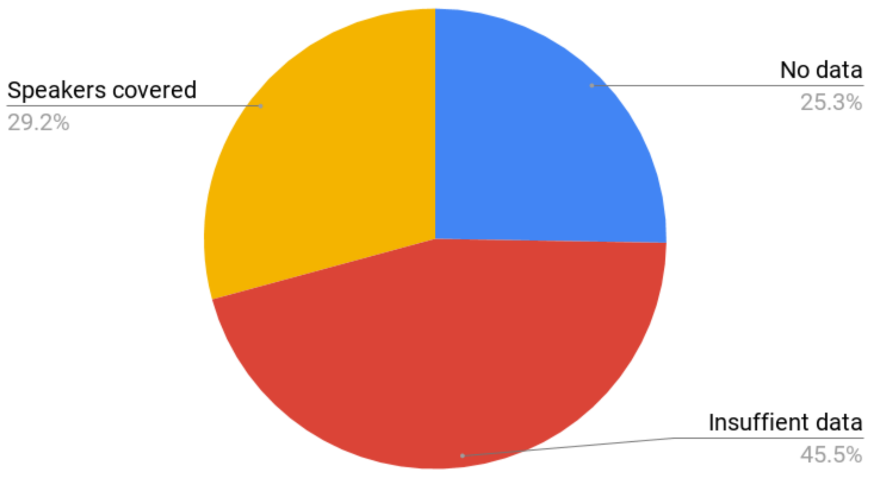


Figure 12: Speakers coverage

The results of the aggregation revealed that the most prominent candidates have gotten the most training data samples. Table 2 shows the ten most covered speakers, along with the party they belong to.

Table 2: Ten most covered speakers

Speaker name	**Data samples count**	**Party**
TAAVI RÕIVAS	374	Eesti Reformierakond
JÜRI RATAS	342	Eesti Keskerakond
KADRI SIMSON	325	Eesti Keskerakond
HANNO PEVKUR	322	Eesti Reformierakond
JÜRGEN LIGI	318	Eesti Reformierakond
TARMO TAMM	315	Sotsiaaldemokraatlik Erakond
URMAS PAET	303	Eesti Reformierakond
SVEN MIKSER	293	Sotsiaaldemokraatlik Erakond
TÕNU OJA	282	Sotsiaaldemokraatlik Erakond
SIIM KALLAS	258	Eesti Reformierakond

Here we can also observe a region specific trend. On the 6th place in Table 2 is Tarmo Tamm, who is a member of Eesti Keskerakond and the Minister of Rural Affairs in the 49th Government of Estonia [25]. Although there might have been a lot of training samples aggregated for him, the fact that "Tarmo" is the 27th most popular first name and "Tamm" is the 2nd most popular last name resulted in a lot of false positives for that speaker [26], [27]. Also there are 3 persons with the name of "Tarmo Tamm" in our collection as it can be seen in "Appendix 1 – List of 2019 Estonian Parliament election candidates". With this set on persons, "Tarmo Tamm" is the only one with non-unique name. A combined chart of all the candidates who received training samples is visible on Figure 12.

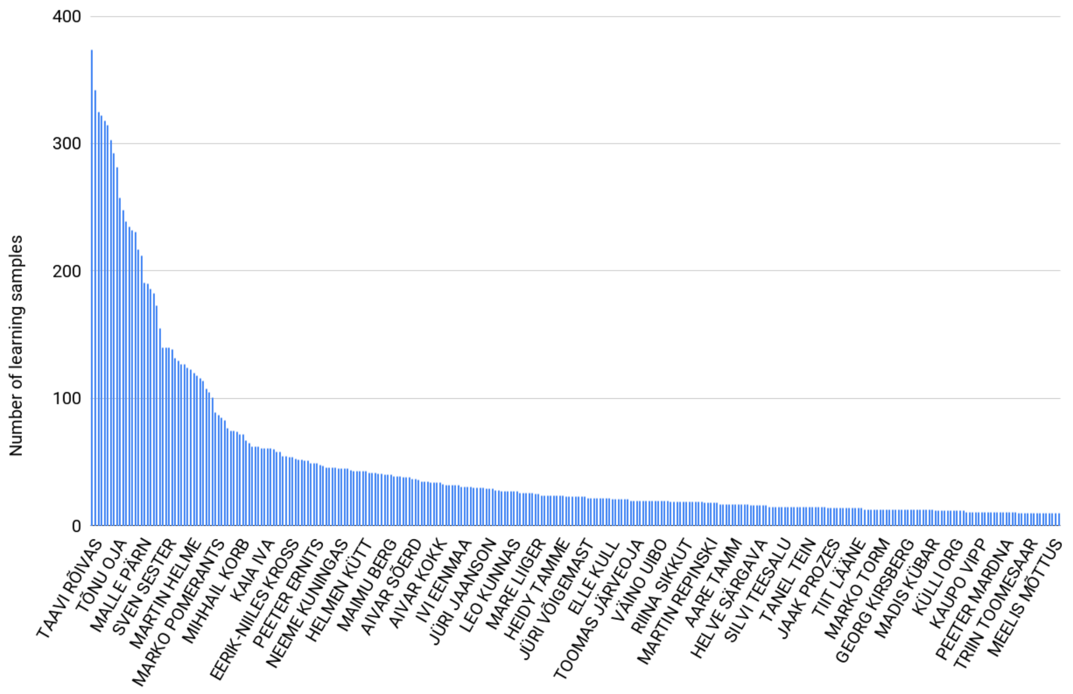


Figure 13: Training samples count by all candidates

When the data is grouped by each candidate's party, then the sum of all training data is represented on Figure 14.

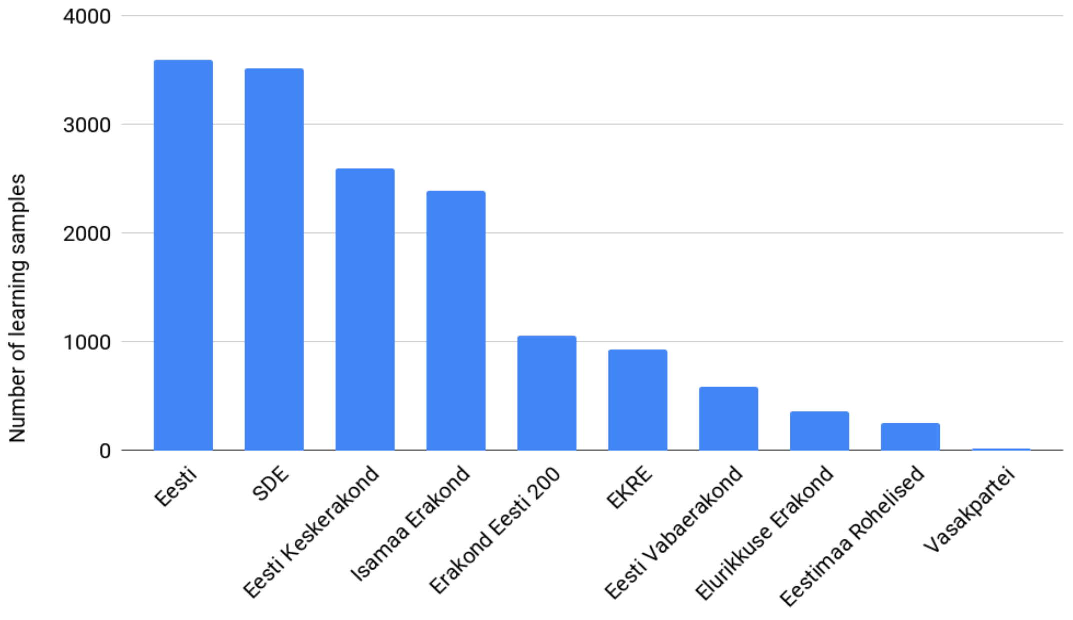


Figure 14: Sum of training data samples grouped by parties

As the aggregation is automatic with some built in false positive detection, the process will not produce perfect results – some of the training samples will have wrong labels added to them. It is caused by the fact that metadata might have facts that are not actually true in the context of gathering training data for speaker identification. For example in the case of YouTube, the aggregator cannot differentiate between person A speaking and when someone else is speaking about person A. For both these cases the person A's name is present in the metadata.

The aggregator developed in this thesis was able to produce data with 88% precision, as it can be seen from the 50 randomly selected samples in "Appendix 2 – Random 50 samples from training dataset". The data from ERR archive is 100% precise and data from YouTube has 73% as the precision rating.

5 Training speaker identification models

This chapter explains how the speaker identification DNN for the thesis was developed. It includes all the processes that are required for training the DNN and later validating its results.

5.1 Training process

The training process uses similar flow as proposed in the experiment that was conducted in chapter 3.1. The process is explained in Table 3 and visualized on Figure 15.

Table 3: Algorithm for training the DNN

Input data	
1. Set of names $N = n_i$, where $i = 1 \dots K$ 2. Aggregator finds for each name n in N a set of audio files a, where $a \subseteq A$ 3. Names n_i where the found audio files $\sum a < 10$ are excluded 4. Unique elements in A are grouped, $A = a_j$, where $j = 1 \dots M$ 5. For each a_j, a set of speakers is found, $S = s_{jki}$, where ki is the number of speakers found and $S \subseteq N$	
Algorithm	
Action	**Result**
1. Applying speaker diarization for audio samples in set A	Audio cluster $D = c_{jli}$ are created for each a_j, where li is are the clusters found for show $a_{j.}$
2. i-Vector extraction	i-Vectors $I{=}c_{jli}$ for each show a_j are found
3. Speaker identification DNN with d inputs and c outputs: d = length of the extracted i-Vectors, (600)	

c = Number of names in set N	
4. Train DNN for E epochs *for j = 1 … M:* *update DNN using label regularization* $l = D(p_j \| p_{\Theta j})$ *Where:* $p_j(speakers) = \left\{ \frac{1}{\|M_j\|}, \text{if speaker is in show, otherwise } 0 \right\}$ p_Θ = *value of the DNN using its current parameters* Θ	

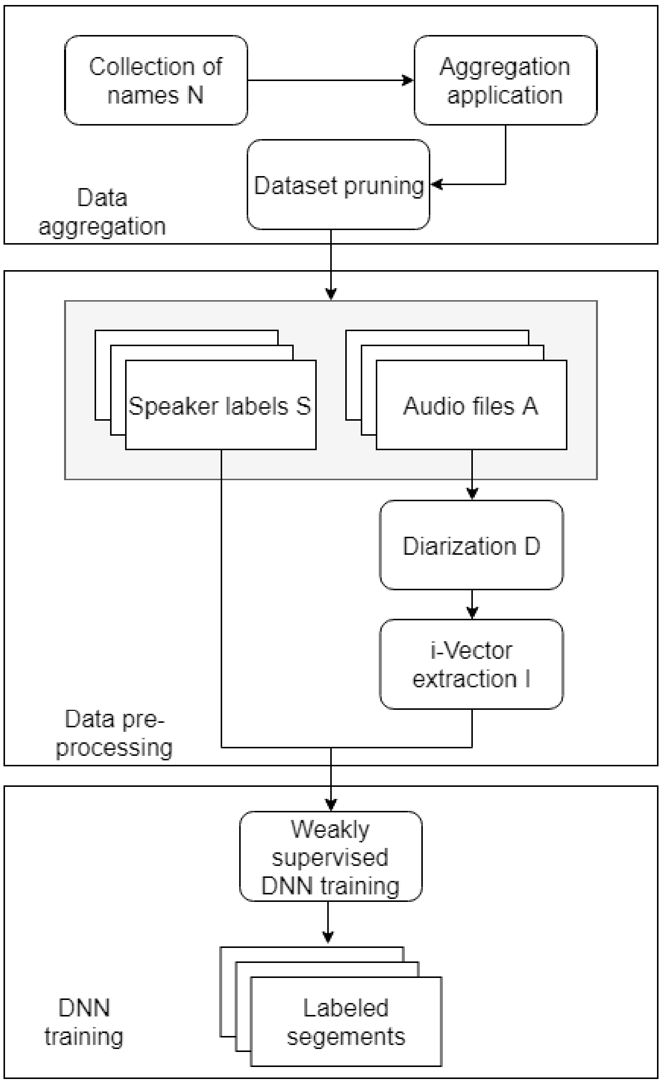


Figure 15: Training process

5.1.1 DNN architecture

The DNN that is used for training the models consists of six layers:

1. Input layer, which receives an i-Vector with 600 dimensions.
2. Dense ReLU layers, which are fully connected layers with ReLU activation function [32].
3. Dropout layers, which will help to prevent overfitting [32].
4. Output layer with Dense Softmax function, which is a fully connected layer and will convert all the outputs to positive percentage values. This way the DNN will output the probabilities for each speaker.

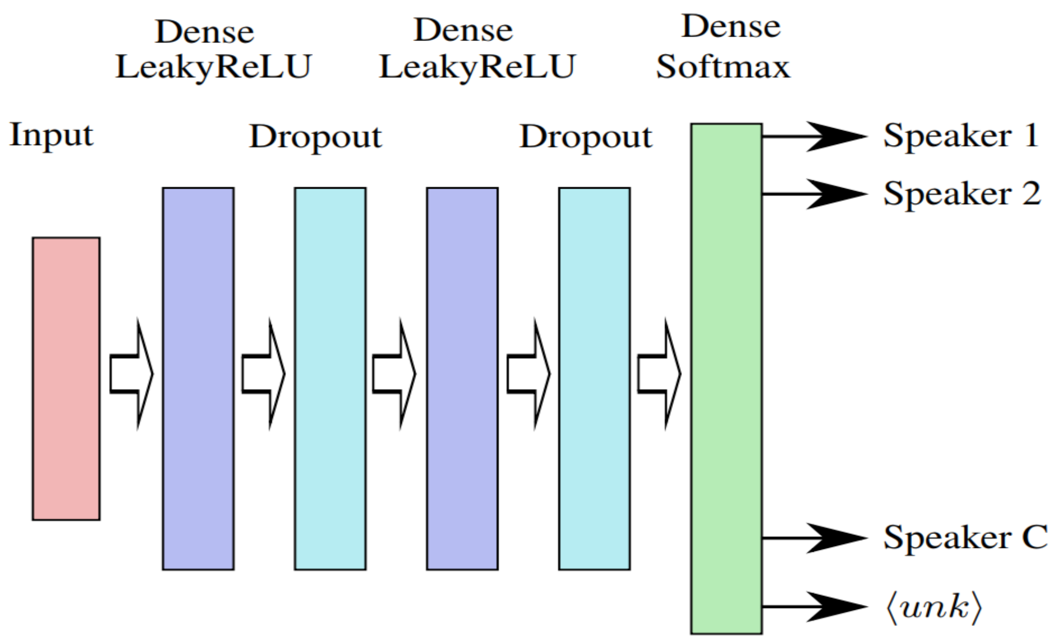


Figure 16: DNN's architecture [8]

The DNN is shown on Figure 16 and its core structure originates from the work done by Karu and Alumäe in [8].

5.1.2 Data aggregation

Data aggregation is the first step in our training process. It is done by feeding a collection of names to the aggregator described in chapter 4.3. Afterwards the resulting dataset is pruned, by removing all persons from the training set who had less than 10 training samples. This is done due to having too low chance to confidently identify speakers with this few samples (Figure 7).

5.1.3 Data pre-processing

First step in pre-processing is applying speaker diarization. It was done by using LIUM SpkDiarization toolkit. It removes all non-speech segments and groups clusters of the same person together [28]. Before diarization, the entire audio sample was one single cluster. After diarization, it is converted into S segments and C clusters, where S is the number of different speakers and C is the number of speech segments. This step will answer the question “who spoke when?”

Secondly, i-Vectors for each cluster will be calculated. For that Kaldi, an open source speech recognition toolkit was used. The calculated i-Vectors are the actual input for training the DNN.

5.1.4 DNN training

For training the DNN, both extracted i-Vectors and recording level speaker labels are used. In one training cycle, the i-Vectors are feed into the system, the DNN evaluates itself and by the information gained from the recording level labels, via back-propagation the weights of the DNN are adjusted. It is illustrated on Figure 17.

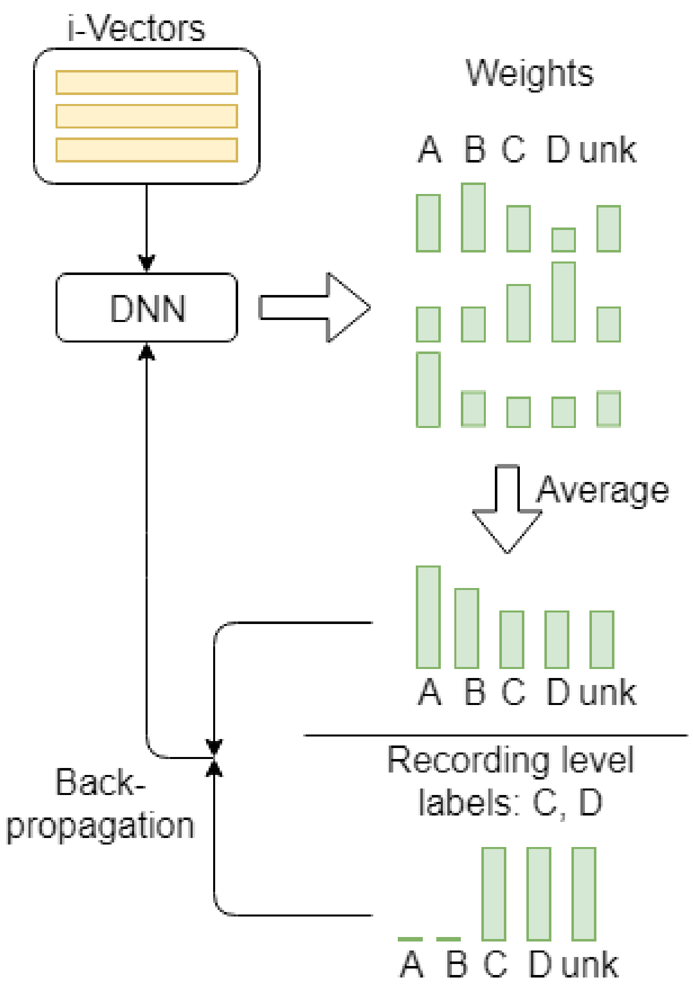


Figure 17: Weakly supervised training cycle

The DNN in this thesis was trained using 9818 unique data samples and it was done for 317 unique speakers. The Python code that was used for training the model is visible in "Appendix 5 – Python Code for Training the Model".

5.2 Validation

This chapter describes how it was made sure, that the trained model was accurate and the labels were attached correctly. A random set of recordings were extracted and the before mentioned speaker identification process was applied. Then the results were compared against hand-annotated data.

5.2.1 Validation targets

The validation targets were two metrics:

1. Recall – the percentage of audio segments that have gotten a label.
2. Precision – the percentage of labels that are correct.

The goal for recall is 70% and for precision 90%. Based on these two metrics, it could evaluated, if a DNN produces confident results.

The formula for calculating recall is as follows:

$$\frac{TP}{TP+FN}, where\, TP = true\, positives\, ; FN = false\, negatives$$

Equation 5: Recall formula

The formula for calculating precision is as follows:

$$\frac{TP}{TP+FP}, where\, TP = true\, positives\, ; FP = false\, positives$$

Equation 6: Precision formula

5.2.2 Using trained DNN for speaker identification

The process of identifying speakers differs from training the DNN. The input data does not contain any metadata about the speakers. An additional parameter T is given, which is the threshold for the identification cycle. Also the system now gives labels for the i-Vectors, if the given threshold is reached. The process is described in Table 4 and visualized on Figure 18.

Table 4: Algorithm for speaker identification

Input data	
1. Audio files $A = a_i$, where $i = 1 \dots N$ 2. Threshold T – confidence score for classifying the i-Vectors	
Algorithm	
Action	**Result**
1. Applying speaker diarization for audio samples in set A	Audio clusters $D = c_{jli}$ are created for each a_j, where li is the clusters found for show $a_{j.}$
2. i-Vector extraction	i-Vectors I=c_{jli} for each show a_j are found
3. Classification of i-Vector V by using the trained DNN	1. S = name of speaker corresponding to the out-put 2. P = posterior probability of the speaker S to be the owner of the i-Vector
4. Discard predictions where posterior probability lower than T	Pruned set of probable speakers for each audio file
5. Return a list of confidently identified speakers S	Collection of probable speakers for each audio file

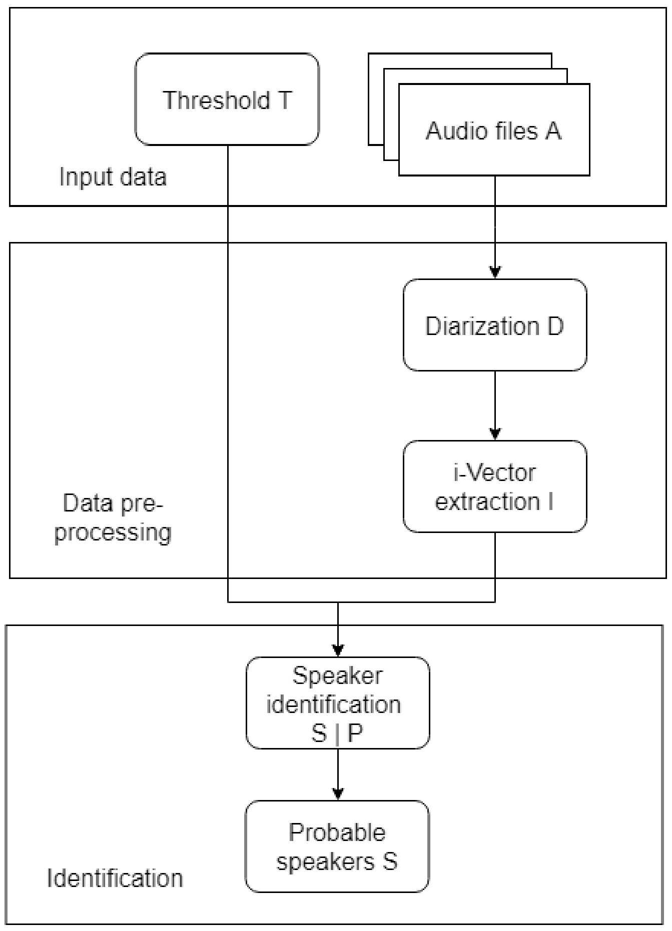


Figure 18: Identification process

5.2.3 Validation process

A set of recordings that were not in the training collection were used for validation. These were 4 shows of Estonian Public Broadcasting's "Valimisstuudio" series. The training data was gathered in January 2019 and the validation dataset is from February and March of 2019.

At first the shows were manually labelled, using Transcriber[1] and then they were fed into the pre-trained DNN, which automatically assigned labels to it. The process of labelling is described in chapter 5.2.2. The complete results are displayed in "Appendix 3 – Validation data". The shows used are the following:

- https://vikerraadio.err.ee/903253/valimisstuudio-majandus-ja-keskkond

1 http://trans.sourceforge.net

- https://vikerraadio.err.ee/903990/valimisstuudio-loimumine-ja-uhiskonna-sidusus
- http://arhiiv.err.ee/guid/20190207023136201000300112290E2BA238B440000004188B00000D0F020382
- http://arhiiv.err.ee/guid/20190304170112601000300112290E2BA238B440000004188B00000D0F030314

All together, validation dataset consisted of 5 hours and 32 minutes of audio recordings and total of 26 persons in interest.

5.2.4 Validation results

From the 26 different persons for whom we wanted to calculate recall and precision metrics, a model was trained for 21 of them. That means that 21 persons in the validation dataset met the minimum requirement of 10 training samples per person rule that was set in chapter 5.1.2. This means our maximum recall value can be as high as 78%. In the context of training and validating speaker models, those speakers are actually not in the interest, due to the fact that they were not included to the training phase. For validation, those 5 speakers were discarded and they were not included to the recall and precision calculation.

The system returned false negative result for 2 speakers, resulting in a 90% recall and all the attached labels were correct which means the precision is at 100%. Both of the false positives were created due to the lack of training samples. One of the speakers had 13 and the other had 10 samples, which in both cases is near the minimum where the system can confidently still attach labels to speech segments (Figure 7).

These results met the goal, which stated 70% recall and 90% for precision. Even if not excluding the 5 non-modelled speakers, the recall is still at 73%, which also fulfils the goal.

6 Experiment

This chapter describes the speaker identification experiment that was conducted using data aggregation method proposed in chapter 4 and by training the DNN using methods of weak supervision. The purpose is to validate this kind of system for real-world speaker identification tasks. The secondary objective was to compare the statistics found from speaker identification with the actual election results and make meaningful observations if possible.

6.1 Test dataset

The experiment was conducted between February 2019 and March 2019, when the Estonian Parliament election campaign was active. The goal was to first train speaker models for all the election candidates and then use the models to identify speakers from a set of hand picked television and radio shows. The total candidates amount was 1084 and the list of them is visible in "Appendix 1 – List of 2019 Estonian Parliament election candidates". As covered in paragraph 4.4, the initial dataset pruning left us with 317 speaker models.

After validating the trained DNN, the test data was chosen. It contained hand picked episodes of different election debates and interviews, that were held in the pre-election period. The shows were downloaded manually from each channels website and were not obtained automatically by using the aggregator described in chapter 4.3, but given the fact that all of those broadcasts included some sort of metadata about the speakers, it could have been done with the proposed application. After downloading, the data was converted to wav file format by using Ffmpeg, which is a cross-platform solution to record, convert and stream audio and video [19], [30]. Detailed overview of test data is visible in "Appendix 4 – The metadata for test dataset".

6.1.1 Statistics

The data sources were ERR's television channels ETV[1] and ETV+[2] and radio channels Vikerraadio[3], Raadio 4[4], Raadio 2[5] and a private radio channel called Kuku Raadio[6]. ERR released a procedure how 2019 election campaign is going to be covered, which included all the channels and their coverage amounts [31]. All together the test dataset consisted of 55 different shows with nearly 55 hours of data to be processed. The shows had, not including the presenters, 210 speakers, from whom 123 were unique.

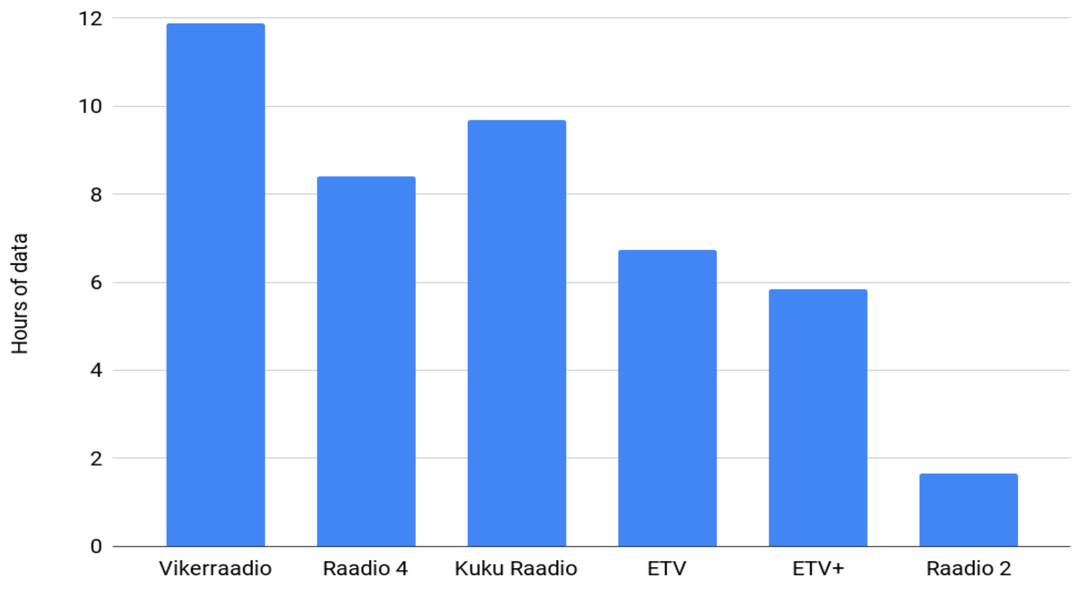


Figure 19: Total speech time by channels

When comparing the results by grouping the shows by channels, then Vikerraadio had the highest duration of any other channels, as it can be seen on Figure 19. A better overview of the proportions is visible on Figure 20.

1 https://etv.err.ee/
2 https://etvpluss.err.ee/
3 https://vikerraadio.err.ee/
4 https://r4.err.ee/
5 https://r2.err.ee/
6 http://kuku.postimees.ee/

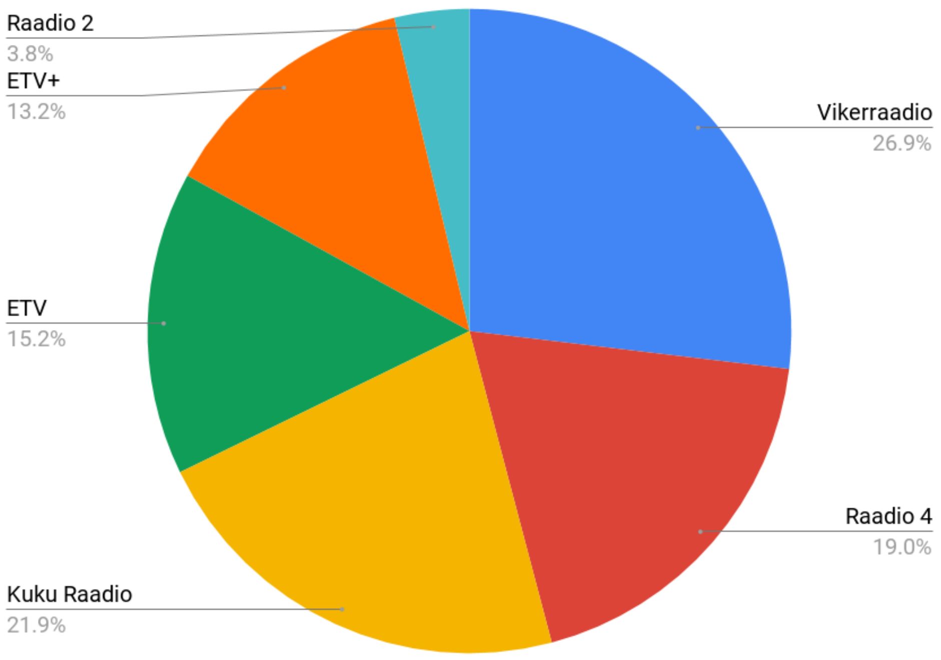


Figure 20: Proportions of total speech time by channels

To view that statistics on how many speakers had the chance to participate a show, then Raadio 4 has the highest number due to having almost double the amount of different shows. Total amount of speakers and different shows is compared on Figure 21.

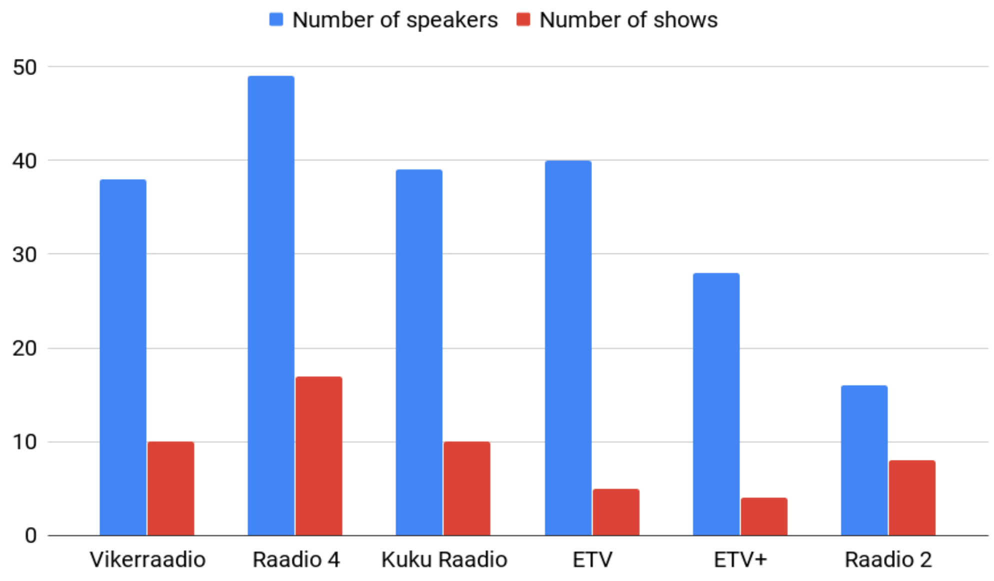


Figure 21: Speakers amount by channels

Another notable value is the duration that a single persons on average had the time to speak. If by simply dividing the total duration of show by the total amount of speakers, we would get some sense of those figures, but as in reality all the shows have a host, or in some cases even multiple, those numbers would not represent the actual real average. During validation phase, which was covered in chapter 5.2, also the speech segments when the presenter spoke were annotated. On average 22% of all the speech belonged to the hosts of the shows. When assuming that that this figure holds true in other shows as well, we can calculate the numbers after removing the duration that it takes for the host to speak. It reveals the results which are displayed on Figure 22 and it could be said, that on Vikerraadio's broadcasts the speakers had more time to speak. But when comparing the results by taking into account the length of the show, then by proportion speakers in Raadio 2 broadcast had the most time per speaker.

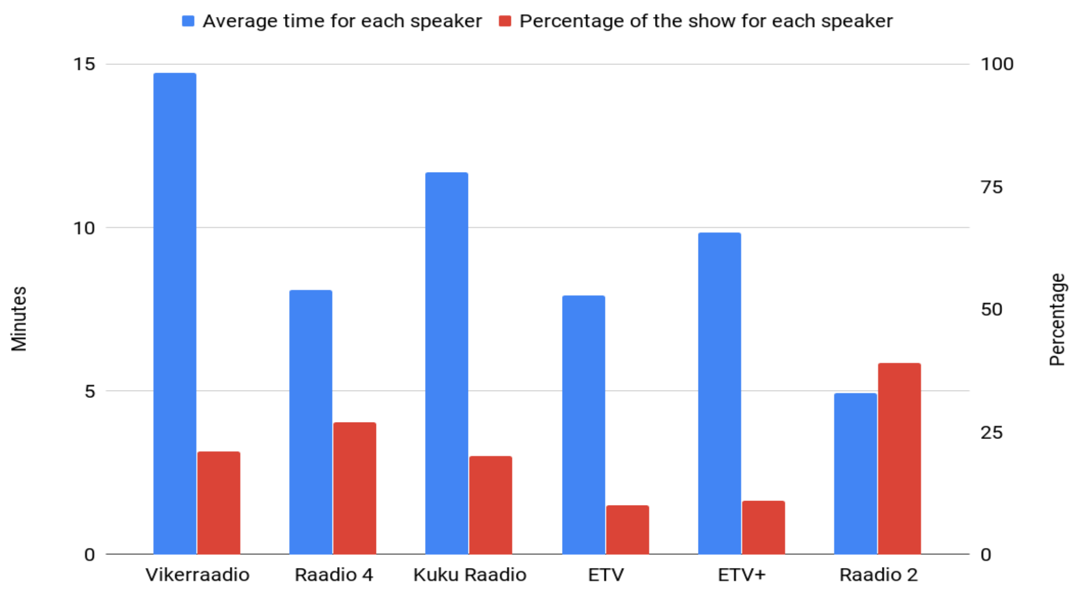


Figure 22: Average speech time for single person by channels

6.2 Results

The test data was fed into the speaker identification system the same way as validation data, which was described in chapter 5.2.2. As the recordings in test dataset contained a list of speakers for each broadcast, it was possible to compare the speaker identification results with those. In this case the comparison is done on a recording level basis.

The hand labelled data revealed that the test dataset contained 123 unique speakers who are in our persons list, but for 54 of those, an identification model was not generated due to lack of training samples. In the perspective of speaker identification we should observe both the entire speakers list and the list of speakers for which a model was generated. Another aspect that should be taken into account are repeated errors with the same speakers. For example if the test reveals that there have been X amount of false negatives, but 100% of those occasions were for speaker Y, then only one speaker model does not return correct results. Grouping by the observable speakers will results in different outcome. This categorization leaves us with the following results:

Table 5: Recall and precision for test dataset

Evaluation method	**Recall**	**Precision**
All speakers and grouped by recordings	58%	99%
All speakers and grouped by speakers	83%	99%
Only speakers with models and grouped by recordings	65%	99%
Only speakers with models and grouped by speakers	88%	99%

As stated in Table 5, the worst result for recall is 58% and the best is 88%. The worst results represent a broader picture, because also those candidates are counted in who did not have enough data for creating a speaker model, whilst the best result only contains speakers for who a model was created. Precision in the other hand is over the four different evaluation methods constantly at 99%. By these results it could be said the in the context of the entire experiment the recall goal was not met, but if looking it in the context of speaker identification the recall goal was achieved, because the lower results were due to not having enough training samples. Precision in the other hand greatly exceeds the objective.

6.3 Analysis

In this chapter both the speaker identification and experiment outcomes are analysed.

6.3.1 Speaker identification analysis

The false negatives of the test are displayed in Table 6 and false positives in Table 7. Both of these values affect the results of recall and precision metrics as it was displayed in chapter 5.2.1.

Table 6: Speakers which created false negatives

Person	Number of training samples
ALEKSANDER LAANE	17
ANDRES METSOJA	46
HANNO PEVKUR	322
HELIR-VALDOR SEEDER	173
JAANUS OJANGU	16
JEVGENI OSSINOVSKI	235
JULIA SOMMER	15
JÜRI RATAS	342
LAURI HUSSAR	232
LAURI TÕNSPOEG	15
OLIVER LOODE	13
RAIMOND KALJULAID	24
RUUBEN KAALEP	21
SIIM VALMAR KIISLER	42
ZÜLEYXA IZMAILOVA	24
TIIA SIHVER	13
ULLA PREEDEN	10

Table 7: Speakers which created false positives

Person	Number of training samples
TOIVO JÜRGENSON	39

With false negatives, the reason why Julia Sommer was not detected is the fact that the training data samples were not valid. A singer in Germany has the same name and that is why this is creating false negatives in the current test. The other false negatives are caused by not having reached the threshold that was set in the DNN.

False positives result on the other hand is very good with only one falsely labelled speaker which resulted in an overall precision of 99%. The one false positive was created on a show[1] where Marko Kaasik and Toivo Jürgenson were mixed up because of two facts: their speech properties are really similar and no speaker model was created for Marko Kaasik due to the lack of training samples.

6.3.2 Analysis of election results and appearances in shows

The objective for this chapter is to compare the speaker identification findings and the election results. The speech duration observations are all achieved with the help of the speaker identification system, so the recall and precision metrics that are displayed in Table 5 should be taken into account.

In the paper, which was covered in chapter "3.3 Leveling the playing field: How campaign advertising can help non-dominant parties" the authors found that non-dominant parties gained the most from radio appearances. By the same notion we could observe the total duration of each candidate's appearances and their election results [34]. With this we can construct the chart which is visible on Figure 23.

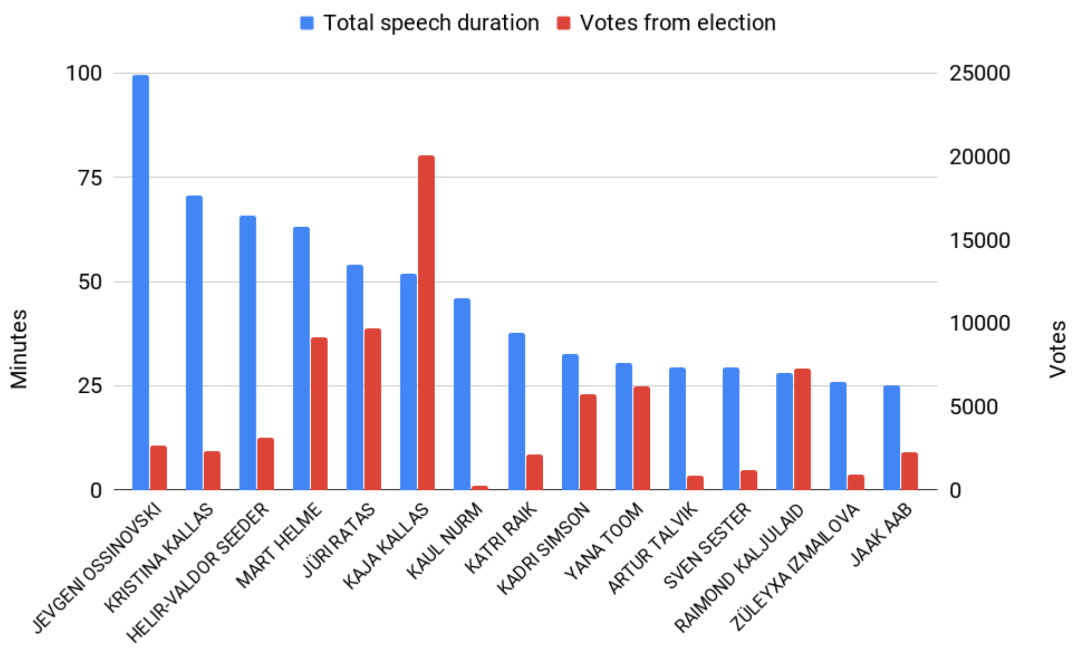


Figure 23: Most spoken candidates and their election results

1 Kuku Raadio, Valimisstuudio 2019-02-25, http://media.kuku.ee/valimisstuudio/valimisstuudio20190225.mp3

On that figure 15 candidates with the most speech segments found are displayed and their corresponding election results as well. Kantor Emor's party support ratings reveal that in January 2019 "Isamaa", "Eesti 200" and "Sotsiaaldemokraatlik Erakond" (SDE) all had 7-11% support, by which they could definitely be regarded as non-dominant parties [33]. By contrast the same survey found that the most popular parties are "Reformierakond", "Keskerakond" and "Eesti Konservatiivne Rahvaerakond" (EKRE) with 25%, 24% and 19% as the support ratings. The findings in this experiment do not support the conclusion which was made in the paper covered in chapter 3.3, that appearing more in the media will benefit smaller parties. Opposite to the latter paper, the 3 candidates – Jevgeni Ossinovski, Kristina Kallas and Helir-Valdor Seeder – who had the highest speech duration and were all members of the aforementioned non-dominant parties got worse results than some candidates who did not participate in any of the shows that were observed in this experiment, as it can be seen from Figure 23 and Table 8.

Table 8: Five candidates who did not appear in any test dataset shows and their election results

Candidate name	**Election result**
Henn Põlluaas	7390
Kristen Michal	6347
Heidy Purga	4763
Siim Pohlak	4152
Taavi Aas	3925

From the same observation its visible that fourth, fifth and sixth spots on the ranking went to the candidates who represented dominant parties in the pre-election period. Those same parties eventually won the elections with "Reformierakond" in the first, "Keskerakond" in the second and "EKRE" in the third place [34]. This means the ranking between dominant parties stayed the same as in the pre-election survey done by Kantor Emor and for non-dominant only "Isamaa" and "SDE" changed places [33].

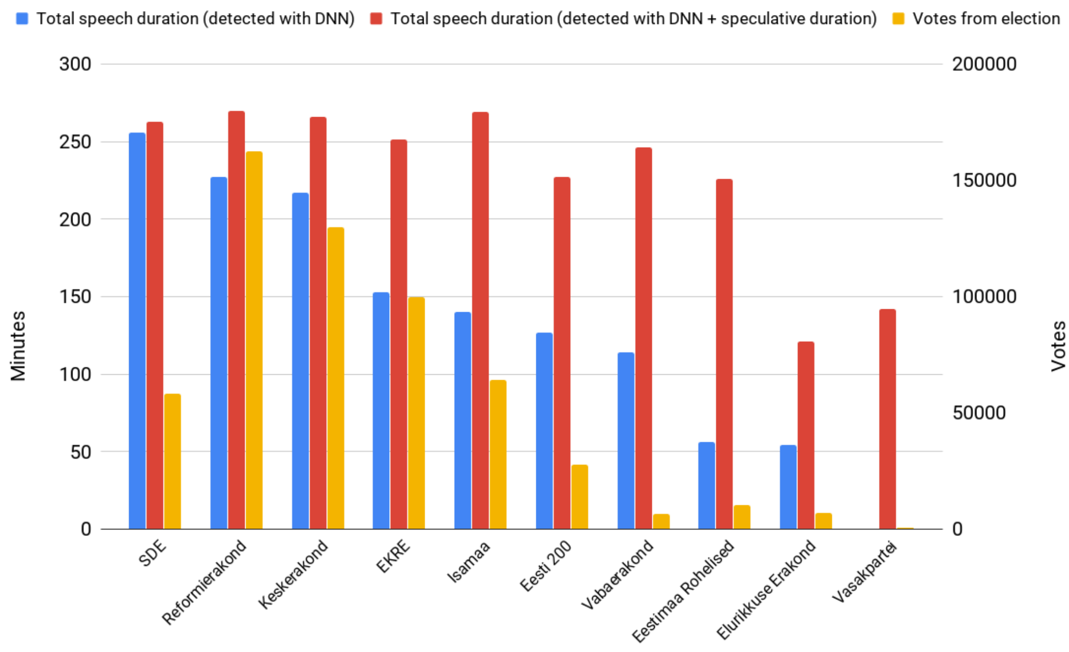


Figure 24: Parties ranking by total speech segments duration

Similar behaviour can be seen on Figure 24 where “SDE” had the highest place in the speech duration ranking, but performed eventually like the pre-election survey indicated.

On Figure 24 also a speculative observation has been done by setting the average speech duration detected with the DNN for those candidates who were not detected and produced false negatives. With this technique we have a speculative overview of the performance of the DNN. It could be said, that with SDE candidates, the DNN detection was the best, as the difference between duration detected with the DNN and with the speculative add-on was the smallest. Besides that, it is visible that for all those parties who submitted a list of full candidates for the elections, the difference between total speech durations is relatively small, which indicates that all had equal opportunities for exposure.

Overall, the data observed indicates that, at least in the setting of this experiment, there is no direct benefit from appearing heavily in broadcast media compared to moderate or even no occurrences. The ranking that was found in the pre-election survey remained at large the same. From Figure 23 it could be said that the non-dominant party candidates invested more time into appearances but as the results showed, it had little effect [34].

7 Applications and future work

In this chapter possible other applications for the speaker identification system are suggested and also improvements for future work.

7.1 Applications

The next chapters suggest alternative uses for this system.

7.1.1 Supplement for a pre-existing speaker identification system

Renewing traditional speaker identification systems where the models were trained on segment-level data can be difficult and time consuming. Using the proposed method for data aggregation and weakly supervised training can make upgrading these systems with new speaker models significantly less challenging.

7.1.2 Improving previously hand-annotated metadata for media broadcasts

If an existing database of annotated broadcasts already exists, then it could be validated or improved, by running a test on that dataset using the proposed speaker identification system.

7.1.3 Toolkit for speaker identification experiments

As the proposed system consists of both the methods for aggregating data for the training samples and the techniques for training speaker identification models, it could be used as a toolkit for running experiments with little overhead from the DNN side. This could make possible for more widespread use of this kind of technology.

7.2 Future work

As found in chapter 6.3.1, a few of the false positives and false negatives of the speaker identification system were caused due to invalid data for training. Also as mentioned in chapter 4.4, data aggregator struggled when a very popular name was used. To prevent both of theses errors, more dimensionality for gathering training samples should be introduced. Using just a name of the person for automatic data collection will not be sufficient for high quality data samples. Other properties of the person like their age, nationality or even picture should be used. Although this is a necessity when it comes to the previously covered issues, it could easily lead to a growth of false negatives, as making the system stricter might reject some samples with insufficient metadata, but which are otherwise correct.

Another area that could need some future improvements is the general amount of different data sources for the developed aggregator. Currently only ERR archive and YouTube were implemented and due to that only 30% of the persons in the current experiment could be covered with speaker models (Figure 12). It is difficult to say how much this metric could have been improved by adding additional sources, but overall this would make the application generally more usable in other areas.

With speaker identification, some issues were with the data pre-processing step (Figure 18) where a cluster for the shows theme music was also created. This is an error with speaker diarization, which should only cluster parts of speech. Current system uses LIUM SpkDiarization toolkit for that, which is widely used for theses tasks [28]. For improving the results, newer versions of the existing toolkit could be used or even alternative solutions like Google's UIS-RNN library [35].

Finally, the entire system which was used, should be packaged together. Currently the data aggregation application is separate from the speaker identification system [23]. For better usability the two system should be merged together for improving future usages.

8 Summary

This chapter analyses whether the problem which was defined in chapter 1.1 was solved and the goals set in 1.2 were achieved. The thesis investigated automatic speaker identification in the context of candidates exposure to broadcast media during the election campaign. The primary goal was to create an effective speaker identification system by using large amounts of impure data which had only recording level labels. The secondary objective was to make observations on candidates exposure to broadcast media and their election results.

For achieving the primary objective, an automatic data aggregation application was developed, which was able to collect data from two different sources – ERR archive and YouTube. The developed application successfully collected over 11 thousand recordings, which were later used for the training phase. After pruning the aggregated data, we had enough data to create models for 317 unique speakers in our target list. This means that with the proposed method for data aggregation, we were able to create models for about 30% of the election candidates. The overall precision for the developed data aggregator was 88%. This means that on average one in ten training samples did not contain the target speaker.

The DNN was trained with weak supervision using the previously aggregated recordings and their metadata. Label regularization was used as the cost function and after each training cycle the weights were adjusted accordingly by the knowledge from metadata related with each recording. When identifying persons from new recordings, a list of speakers and their respective posterior probabilities were returned. With this method on the validation dataset the DNN yielded 73% for recall and 100% for precision. That dataset contained new recordings which were not included in the training dataset. As the goals given in chapter 1.2 stated 70% for recall and 90% for precision, it could be said that these objectives were achieved.

In the experiment phase another set of recordings was chosen. Similarly to the validation dataset, these recordings also were not in the training set. As all of them contained metadata about the speakers, we could again compare the hand annotated results to those found by the speaker identification system. Depending on the evaluation method, the results were 58% and 88% for the worst and best recall and 99% for precision. Although the results were worse than those achieved on the validation dataset, this set contained about 10 times more recordings and covered a larger range of speakers. In the contexts of the current problem, having fewer false positives is more important that fewer false negatives. The current experiment only had one false positive label and the results also achieved the goals for both recall and precision in the context of speaker identification.

By analysing the results from the aforementioned experiment, we could not draw the same conclusions that the authors of the paper [16] came to, which suggested that non-dominant parties would benefit more from the exposure to media, as the findings of this thesis did not observe this kind of behaviour. Oppositely, the SDE party had the highest number of speech segments, but eventually had one of the worse overall results. The results of the pre-election survey at large matched with the results of the actual election. From the observations made, it could be said that the non-dominant party candidates invested more time into appearances but as the results showed, it had little effect.

This thesis proves that it is possible to use methods of weak supervision to create large amount of speaker models by using several data sources with impure quality training samples. The combination of the proposed data aggregation method and weakly supervised training can produce results with high precision. Also by using this kind of system, it was possible to analyse a large scale public event, which in this case was the 2019 Estonian Parliament elections. The observations did not directly indicate towards the improvement of election result by having more exposure to broadcast media.

References

[1] Reynolds, D. A.. Speaker identification and verification using Gaussian mixture speaker models. - *Speech Communication,* 1995, 17, pp. 91-108.

[2] Nautsch, A. Speaker verification using i-vectors, Evaluation of text independent speaker verification systems based on identity-vectors in short and variant duration scenarios. Darmstadt, Hochschule Darmstadt, University of Applied Science, 2014.

[3] Garcia-Romero, D. Espy-Wilson, C. Y. Analysis of I-vector Length Normalization in Speaker Recognition Systems. - *INTERSPEECH 2011,* 2011, pp 249-252.

[4] Reynolds, D. A. Gaussian mixture models. - *Encyclopedia of biometrics*, 2015, pp. 827-832.

[5] Reynolds, D. A. Universal background models. - *Encyclopedia of biometrics*, 2009, pp. 1349-1352.

[6] Beal, M. J., Ghahramani, Z., Rasmussen, C. E., The infinite hidden Markov model. - *In Advances in neural information processing systems,* 2002, pp. 577-584.

[7] Tranter, S. E., Reynolds, D. A. An overview of automatic speaker diarization systems. - *IEEE Transactions on audio, speech, and language processing*, 2006, 14(5), pp. 1557-1565.

[8] Karu, M., Alumäe, T. Weakly supervised training of speaker identification models. - *Odyssey 2018 The Speaker and Language Recognition Workshop : 26-29 June 2018, Les Sables d'Olonne, France, Proceedings,* 2018, International Speech Communication Association, pp. 24–30.

[9] Bengio, Y. Learning deep architectures for AI. - *Foundations and trends® in Machine Learning,* 2009, 2(1), pp. 1-127.

[10] Schmidhuber, J. Deep learning in neural networks: An overview. - *Neural networks,* 2015, 61, pp. 85-117.

[11] Deng, L., Yu, D. Deep learning: methods and applications. - *Foundations and Trends® in Signal Processing,* 2014, 7(3–4), pp. 197-387.

[12] Hinton, GE., Osindero, S., Teh, YW. A fast learning algorithm for deep belief nets. - *Neural computation,* 2006, 18(7), pp. 1527-54.

[13] Hughes, D., Correll, N. Distributed machine learning in materials that couple sensing, actuation, computation and communication. 2016.

[14] Nagrani, A., Chung, JS., Zisserman, A. VoxCeleb: A large-scale speaker identification dataset. - *Interspeech*, 2017, pp. 2616–2620.

[15] Schmidt, L., Sharifi, M., Moreno, IL. Large-scale speaker identification. - *2014 IEEE International conference on acoustics, speech and signal processing (ICASSP),* 2014, pp. 1650-1654.

[16] Larreguy, HA., Marshall, J., Snyder, JM. Leveling the playing field: How campaign advertising can help non-dominant parties. - *Journal of the European Economic Association,* 2018, 16(6), pp. 1812-49.

[17] Karu, M. Weakly supervised training of speaker identification models: master's thesis. Tallinn, Tallinn University of Technology, 2017.

[18] Wissner-Gross, A. Datasets Over Algorithms. - *Edge.com* [WWW] https://www.edge.org/response-detail/26587 (08.04.2019).

[19] Whibley, S., Day M., May P., Pennock M. WAV Format Preservation Assessment. - *Technical Report. British Library*. [WWW] http://wiki.dpconline.org/images/4/46/WAV_Assessment_v1.0.pdf (08.04.2019).

[20] Zhou, Z. H., Zhang, M. L. Multi-instance multi-label learning with application to scene classification. - *In Advances in neural information processing systems*, 2007, pp. 1609-1616.

[21] Martin, R. C. Clean code: a handbook of agile software craftsmanship. Pearson Education, 2009.

[22] Lutz, M. Learning Python: Powerful Object-Oriented Programming. - *O'Reilly Media*, 2013.

[23] "GitHub – siimtalts/speaker-model-data-aggregator: Speaker model data aggregator" [WWW] https://github.com/siimtalts/speaker-model-data-aggregator (08.04.2019).

[24] Mitchell, R. Web Scraping with Python: Collecting More Data from the Modern Web. - O'Reilly Media, 2018.

[25] "Government 23.11.2016- ...". [WWW] https://www.valitsus.ee/en/valitsus/varasemad-valitsused/id/49 (09.04.2019).

[26] "Populaarsed eesnimed". [WWW] https://www.stat.ee/public/apps/nimed/TOP (09.04.2019).

[27] "Levinuimad perekonnanimed". [WWW] https://www.stat.ee/public/apps/nimed/pere/TOP (09.04.2019).

[28] Meignier S., Merlin T. LIUM SpkDiarization: an open source toolkit for diarization. - *CMU SPUD Workshop*, 2010.

[29] "GitHub – kaldi-asr/kaldi: Kaldi Speech Recognition Toolkit". [WWW] https://github.com/kaldi-asr/kaldi (09.04.2019).

[30] "FFmpeg Documentation". [WWW] https://ffmpeg.org/ffmpeg.html (10.04.2019).

[31] "Riigikogu valimiskampaania kajastamise kord Eesti Rahvusringhäälingus 2019. aastal". [WWW] http://info.err.ee/v/avalikteave/valimistekajastamine/26dee461-a3c8-43e8-a75d-f9f7500599bc/riigikogu-valimiskampaania-kajastamise-kord-eesti-rahvusringhaalingus-2019-aastal (10.04.2019).

[32] "PyTorch Documentation". [WWW] https://pytorch.org/docs (12.04.2019).

[33] “Erakondade toetusreitingud”. [WWW] https://www.emor.ee/erakondade-toetusreitingud (13.04.2019).

[34] “2019 Riigikogu valimised (03.03): Detailne hääletamistulemus”. [WWW] https://rk2019.valimised.ee/et/voting-result/voting-result-main.html (13.04.2019).

[35] “GitHub – google/uis-rnn: UIS-RNN library”. [WWW] https://github.com/google/uis-rnn (13.04.2019).

[36] Van Aelst, P., Van Erkel, P., D’heer, E., Harder, R.A. Who is leading the campaign charts? Comparing individual popularity on old and new media. - *Information, communication & society*, 2017, 20(5), pp. 715-732.

[37] Owen, D. New media and political campaigns. - *The Oxford handbook of political communication*, 2018.

[38] Dehak, N., Kenny, P.J., Dehak, R., Dumouchel, P., Ouellet, P. Front-end factor analysis for speaker verification. - *IEEE Transactions on Audio, Speech, and Language Processing*, 2011, 19(4), pp.788-798.

Appendix 1 – List of 2019 Estonian Parliament election candidates

Party	Candidates count
Eesti Vabaerakond	125
Eesti Reformierakond	125
Erakond Eestimaa Rohelised	125
Sotsiaaldemokraatlik Erakond	125
Elurikkuse Erakond	73
Eesti Konservatiivne Rahvaerakond	125
Erakond Eesti 200	125
Eesti Keskerakond	125
Eestimaa Ühendatud Vasakpartei	11
Isamaa Erakond	125
Total	**1084**

Party	Candidates
Eesti Keskerakond	JÜRI RATAS, KADRI SIMSON, MIHHAIL KÕLVART, MAILIS REPS, JAANUS KARILAID, YANA TOOM, KERSTI SARAPUU, ENN EESMAA, MIHHAIL KORB, TANEL KIIK, TOOMAS VITSUT, JAAN MÄNNIK, KALEV KALLO, ANNELI OTT, ERKI SAVISAAR, AADU MUST, MIHHAIL STALNUHHIN, NATALIA MALLEUS, MARIKA TUUS-LAUL, ANDREI KOROBEINIK, IMRE SOOÄÄR, TIIT TERIK, JAAN TOOTS, VALERI KORB, KALLE KLANDORF, MARKO ŠORIN, ESTER KARUSE, VIKTOR TRAS-BERG, TARMO TAMM, SIRET KOTKA-REPINSKI, JAAK AAB, JANEK MÄGGI, RAIMOND KALJULAID, KRISTIINA HEINMETS-AIGRO, KAIDO HÖÖVELSON, AARE OLGO, EEVI PAASMÄE, TOOMAS VÄINASTE, VIKTOR VASSILJEV, DMITRI DMITRIJEV, MÄRT SULTS, KERSTIN-OUDEKKI LOONE, HEIMAR LENK, HELMUT HALLEMAA, VLADIMIR VELMAN, IGOR KRAVTŠENKO, EHA VÕRK, MIHKEL UNDREST, AIVAR RIISALU, ANDRA VEIDEMANN, HANS LIIBEK, HILLAR TALVIK, HELLE KALDA, ANDREI NOVIKOV, MARIA JUFEREVA-SKURATOVSKI, MAREK JÜRGENSON, VADIM BELOBROVTSEV, VLADIMIR ŠOKMAN, MARINA RIISALU, PEETER RAHNEL, ANTTI LEIGRI, TÕNIS MÖLDER, TARMO TAMMISTE, KERSTI MÄNNIK, AIVAR ARU, LAURI LAATS, TRIIN VAREK, ANTS LOPSIK, JANIKA USIN, MERIKE MARTINSON, VLADIMIR SVET, GRETE ŠILLIS, KARL ÕMBLUS, JAANUS KALEV, ÜL-LAR NEEMRAND, MAKSIM BUTŠENKOV, MARTIN REPINSKI, ARDO AASA, ANDRES VÄÄN, JÜRI VÕIGEMAST, ANDRE LAINE, MONICA RAND, KRISTA NÕMM, EDUARD TOMAN, OLEG TŠUBAROV, MAIRE KÄÄRID, JELENA FJODOROVA, RAIVO MATSINA, PEETER LAASIK, IR-INA EMBRICH, JELENA FRUNZE, JAANUS RAIDAL, ÜLLE JUHT, ÜLO TULIK, MAKSIM VOLKOV, AIVAR ROOP, REA RAUS, AADO OHERD, ANDRES ANNAST, JAAN ÕUNAPUU, IRINA PANOVA, JÜRI SAAR,

Party	Candidates
	TARMO TOMSON, ILJA BAN, NIKOLAI PÕDRAMÄGI, MAX KAUR, JAANUS RIIBE, VLADIMIR ARHIPOV, GETTER KLAAS, MIKK PIKKMETS, INDREK UUEMAA, MADIS LEPAJÕE, ERKI TAMMLEHT, PEEP PÕDDER, VIIVE ROSENBERG, SERGEI PTŠJOLKIN, JÜRI ENNET, HEIDY TAMME, EVE EAST, PEETER MARDNA, HANS KRUUSAMÄGI, JOSEF KATS, JÜRI KUUSKEMAA, AVO KEEL, TAAVI AAS
Eesti Konservatiivne Rahvaerakond	MART HELME, MARTIN HELME, HENN PÕLLUAAS, JAAK MADISON, PAUL PUUSTUSMAA, MERRY AART, LEO KUNNAS, HELLE-MOONIKA HELME, SIIM POHLAK, RUUBEN KAALEP, RENE KOKK, JAAK VALGE, RIHO BREIVEL, PEETER ERNITS, ANTI POOLAMETS, KERT KINGO, KAI RIMMEL, URMAS ESPENBERG, ARNO SILD, ALAR LANEMAN, TIIT KALA, URMAS REITELMANN, UNO KASKPEIT, REIN SUURKASK, HEL-DUR PAULSON, KADRI VILBA, KERSTI KRACHT, ENDEL OJA, KALLE GRÜNTHAL, MAIT TALU, RAIVO PÕLDARU, INDREK SÄRG, KAIRET REMMAK-GRASSMANN, WESSE ALLIK, JAEN TEÄR, MARTI KUUSIK, EDMOND PENU, TARMO HINTS, MALLE PÄRN, HARDI REHKALT, MEELIS MALK, ARGO LUUDE, AARNE MÄE, RIHO NÜÜD, ARVO AL-LER, MAREK LEINEMANN, ÜLLE ROSIN, MART KALLAS, HELLE KULLERKUPP, KARL OLAF RÄÄK, INDREK HIRV, AGNES PULK, UR-MAS SIMON, EVE PÄRNASTE, AIVO LILL, MART JÄRVIK, INDREK KÄO, MEDI MEIKAR, ROBERT KIVISELG, KALEV PRITS, RAUL ÖPIK, MAREH KALDA, TOIVO TASA, RAIVO PÕHJAKIVI, TÕNU URVA, ILLE PALUS-ALU, SIIMO LOPSIK, UNO TRUMM, KARINE SIIAK, TÕNIS TÄHE, GAIUS GIL, ENN SARV, ANTS MILLER, JAANUS HÄRMS, INDREK KALDA, HASSO UUETOA, KAIDI SUVISTE, AIVAR ÕUN, ANDRE PERE, ROLAND TÕNISSON, EVELIN POOLAMETS, TANEL TARKMEES, ÕNNELI MATT, ANTS TAUL, VEIKO HARTWIG VALGEPEA, ALAR VALGE, HELI KOIT, TOOMAS VALLIMÄE, AARE TAMM, ANDRES PIIBELEHT, RIHO RAUSMA, VILJO TAMM, MEELIS ROSENFELD, KALMAR LUISK, JÜRI BÖHM, AIN EHARI, MARE LIIGER, EDA RÜÜTEL, PEETER KRALL, AIGAR PÕDER, KRISTIAN SUVE, JUHAN KOBIN, RANNO POOL, MAR-TIN JUDANOV, MEELIS AIANURM, KÜLLI REMSU, MERIKE LUMI, PIRET KOLLO, TIIT RAUD, TÕNU KAUR, AIN PEDAJAS, ANDO TULVIK, URMAS SIEMER, RINGO RIIVES, RUDOLF JEESER, DANIEL MEREÄÄR, HEIKI VILEP, JÜRI KAVER, GEORG KIRSBERG, AARE RÜÜTEL, KATRIN LINDE, LEO BERGSTRÖM, MARI-LIIS LILLEMETS, MIHKEL SARETOK, PRIIT JÜRJENS
Eesti Reformi-erakond	KAJA KALLAS, JÜRGEN LIGI, KRISTINA ŠMIGUN-VÄHI, HANNO PEVKUR, URMAS PAET, KRISTEN MICHAL, AIVAR SÕERD, MARIS LAURI, ARTO AAS, ANTS LAANEOTS, ANDRES SUTT, URMAS KLAAS, KEIT PENTUS-ROSIMANNUS, EERIK-NIILES KROSS, KALLE LAANET, LIINA KERSNA, URMAS KRUUSE, URVE TIIDUS, YOKO ALENDER, MARKO MIHKELSON, KALLE PALLING, VILJA TOOMAST, LAINE RANDJÄRV, JOHANNES KERT, SIGNE KIVI, TOOMAS KIVIMÄGI, AN-DRUS SEEME, JÜRI JAANSON, VALDO RANDPERE, HEIKI KRANICH, ERKKI KELDO, IVI EENMAA, MATI RAIDMA, MADIS MILLING, HEIDY PURGA, ANNELY AKKERMANN, MARGIT SUTROP, ÕNNE PILLAK, SULEV KANNIMÄE, TIIU ARO, ÜLLE RAJASALU, TOOMAS JÄRVEOJA, JAANUS RATAS, MADIS TIMPSON, ANTS LEEMETS, TOOMAS KRUUSIMÄGI, MART VÕRKLAEV, HELE EVERAUS, ANTI HAUGAS, MERIKE LANG, MARGUS LEPIK, MAIDO RUUSMANN, LIIS KLAAR, AIRIS MEIER, MARIS TOOMEL, RAIT PIHELGAS, PEETER SIBUL, UR-MAS SIIGUR, ÜLO NEEDO, KATRIN KUUSEMÄE, OTT KASURI, KRISTO ENN VAGA, ANDRUS UMBOJA, KAUPO NÕLVAK, SIGNE RIISALO, VLA-DAS RADVILAVIČIUS, RENO LAIDRE, AIVAR SURVA, LEO AADEL, AIVAR SAARELA, TIMO SUSLOV, MARIKA VALTER, TARMO ALT, JÜRI

Party	Candidates
	LEBEDEV, MAIRE FORSEL, JAAN ALVER, MEELI PÄRNA, EERO MER-ILIND, RAIVO MEITUS, STEN LAANSOO, AIVAR VIIDIK, ANDRES ARO, RAIMOND TAMM, JAANUS MÜÜR, JUHAN KANGILASKI, LEMBIT KOLK, TÕNU JUUL, AIVAR ROSENBERG, MARKO TORM, TIIT SOORM, ANDRUS PUNT, MARGIT ELVISTE, GEROL SILKIN, MADIS KOIT, TOOMAS RÕHU, MERILI RANNISTE, INDREK KESKÜLA, MATI SADAM, HELEN MAHMASTOL, ILLARI LÄÄN, KADRI KURM, TÕNU MEIJEL, RAINAR ENDEN, ANDREA EICHE, OLESJA OJAMÄE, ANDRES MÕIS, PRIIT PALMET, TOOMAS NÕMMISTE, SERGEI GORLATŠ, TAIMO TUGI, TIIT VAHENÕMM, ÜLVI NOOL-KÕRE, ART KUUM, KATRIN HELENDI, TAAVI TOPPI, MARKO RAUDLAM, MERIKE PERI, TOOMAS KÄRK, EIN-IKE MÖLDER, TIINA RÜHKA, LIANA NÕGENE, RIHO TELL, MEELIS SOLL, TAAVI RÕIVAS, SIIM KALLAS
Eesti Vabaerakond	KAUL NURM, TIINA KANGRO, ANDRES HERKEL, NEEME KUNINGAS, VAHUR KOLLOM, ELO LUTSEPP, PAVO RAUDSEPP, AIN LUTSEPP, JAANUS OJANGU, MÄRT LÄÄNEMETS, KUIDO NÕMM, JUKU-KALLE RAID, EPP ALATALU, HELI KÜNNAPAS, TIIU KUURME, TIIA VÄLK, JANE SNAITH, RAIVO KOKSER, MÄRT MEESAK, AIN OSTRA, MERIKE VÄRIK, ÕIE-MARI AASMÄE, INNA ROSE, ANNE HANSBERG, KAJA JAKOBSON, SIIRI KÄPA, ARVET LINDSTRÖM, TÕNU PLOOMPUU, HEIKI LILL, MIKKO NÕUKAS, KRISTO KRUMM, KATRIN HAUK, JAAK PROZES, TÕNU TEEVEER, URMAS HEINASTE, LAILI JÕGIAAS, URMAS OTT, ARLET PALMISTE, JÜRI PINO, KAJA VAJAK, JAANIKA KLOPETS, EVE VIIDALEPP, ALEKSANDR DORMIDONTOV, SUIDO SAARMETS, ELGE HÄRMA, PIRET PIHEL, SIRJE PALLO, MARI-LIIS TAHKER, KAIDO VÄLJAOTS, KAIDO KALLAVUS, MARIKA PARV, TOOMAS ALATALU, JAANUS PAASOJA, TAAVI SIMSON, MARGUS RAHA, VAHUR HERM, OLEV VAHER, KOIT KUUSK, KARINA KÜPPAS, JÜRMO SILLASTE, TIIT URVA, KRISTO KARU, ÜLLAR VANA, ELMO JOA, PEETER LIINSOO, EVE OSA, TÕNU ARU, ELDAR TOONVERK, ANNE KIIDER, REIN KOTŠIN, ANDO KUURA, AIN AUKSIMÄE, TAUNO KURE, HERLE GER-ASSIMOVA, ANNIKA TOOTS, FREDY VABRIT, MARIKE LAHT, ANDRES LINTS, VIKTORIA LUKATS, KAIE HERKEL, EGON ERKMANN, MARJE SALVE, MULJE KULLERKUP, HEIKI RIPS, ELAR PLOOMIPUU, ALVAR TIPP, KATI KONGO, ALLAR SAU, SANDER KLAUSEN, AVE LOSSMANN, RIINA KURG, JANUS TUUR, AVE PÜTSEPP, DENISS TIŠTŠENKO, OLEV RAHNU, HEINO RITSMAN, TÕNIS LUKATS, DANDY ROSENBERG, AGE MINKA, SIIM TUUR, JAAK VACKERMANN, JANEK OSTRA, VEANA HEINMETS, ASKO SIRP, HANNO KIRSCHFELDT, MART KIVASTIK, UR-MET KÜLAOTS, ÜLLE ORAV, ALAR TINNUS, REIN REBANE, LIDIA RET-TIJEVA, MARGUS SARDIS, MIHKEL MÄLLO, JÜRGEN POST, IIVI ZAJE-DOVA, AARE PERNIK, KRISTI VARUL, REIN PRINS, ALDO VAINO, PEETER LUKAS, HARLY KIRSPUU, DEIN-TOM TÕNSING, MATI ROOS-NURM, MIHHAIL JALLAJAS, MART MUTSO
Eestimaa Ühendatud Vasakpartei	JULIA SOMMER, ANDREI ORPONEN, VIIVI-HELBE PELJUHHOVSKA, OLEG TESLA, TIINA RADIONOV, DIANA NÕMBERG-KLAUS, VIKTOR MALÕGIN, DIANA KOOR, VJATŠESLAV MAKARONSKI, IGOR ROSEN-FELD, ANDREI ANISSIMOV
Elurikkuse Erakond	MIHKEL KANGUR, AHTO KAASIK, ARTUR TALVIK, KAJA TOIKKA, KAUPO VIPP, LAURI KLEIN, LAURI TÕNSPOEG, MATI KOSE, PEETER LEPISK, RAINER KUUBA, RIINA EIGI, TOOMAS TRAPIDO, AGO SAMO-SON, AIRI HALLIK-KONNULA, AIVAR RUUKEL, ALEXANDER LIN-NAMÄE, ALLAN KOKKOTA, ALLAN PROOSO, ANDRES VESPER, AN-NELI PALO, CORNELIA KOTTO, EDA VEEROJA, EHA METSALLIK, ENN KALJO, ENRI PAHAPILL, EVE RANDVERE, GRETE PERTEL, HELEN

Party	Candidates
	ORAV-KOTTA, HELI PIISANG, JAANUS SAADVE, JAANUS-JUHAN IL-LEND, JANICA SEPP, JÜRI KÕUK ROOST, JÜRI PETERSON, JÜRI-ILLI-MAR REINBERG-ŠESTAKOV, KAJA KARU-ESPENBERG, KALLE VIRKUS, KERSTI REPPO, KERTU NAMSING, KÜLLI ORG, LIIS TRUUBON, LILIAN FREIBERG, MADIS IGANÕMM, MADIS MARK, MARIKA JAHILO, MART JÜSSI, MÄRT RANDOJA, OLIVER LOODE, OTT KÖSTNER, PAAVO EENSALU, PEEP TOBRELUTS, PIRET RÄNI, PRIIT-KALEV PARTS, REET POOM, REIN EINASTO, REIN LEPP, RIHO KIR-MJÕE, RIINA SOONE, RIINU LEPA, SIIRI TIIVITS-PUTTONEN, SILJA REE-MET, SIRKKA PINTMANN, TAUNO JÜRGENSTEIN, TAUNO LAASIK, TEET RANDMA, TEHO PAULUS, THEA PERM, TOOMAS FREY, URMAS EELMÄE, VAL RAJASAAR, ÜLLE KAUKSI, ÜLLE KULDKEPP, ÜLO MÄN-GEL
Erakond Eesti 200	KRISTINA KALLAS, LAURI HUSSAR, MEELIS NIINEPUU, KADRI TALI, TIIA SIHVER, ÜLO PIKKOV, LIINA NORMET, TOOMAS UIBO, TAAVI TOOM, GRIGORE-KALEV STOICESCU, IGOR TARO, TIIT ELENURM, KRISTIINA TÕNNISSON, MAREK REINAAS, EINAR VISNAPUU, MONIKA SALU, NIKITA LUMIJÕE, AHTI PUUR, ANNES NAAN, RUSLAN TROCHYNSKYI, ANDO KIVIBERG, KOIT KELDER, ALEKSANDR ŠIROKOV, TERJE BACHMANN, MART SANDER, JANA PAVLENKOVA, PIRKO KONSA, KADRI HALLER-KIKKATALO, JOEL JESSE, KALLE KAL-LAS, ALEKSEI JAŠIN, PILLE TSOPP-PAGAN, ERIK VEST, AARE LAPÕNIN, JAAK LAINESTE, EVELIN PÄRL, MARKO TEIVA, TOOMAS KASEMAA, JULIA RUSTAMOVA, SIIRI LEHTMETS, SIRJE NIITRA, KRIS-TER KALLAS, JÜRI KÄOSAAR, JÜRI LEHTMETS, SVETLANA SKREB-NEVA, ELMU KOPPELMANN, TOOMAS PAJULA, MERLIS PAJUSTIK, DARJA BERESTOVA, ANGELIKA NARIS, KADI PLOOM, KARIN KAUP-LAPÕNIN, PAVEL STARKOV, MADIS VIISE, JAANIKA ROOSMAN, ARE SELGE, JÜRI-OTT SALM, JANNO KULDKEPP, MARJU MÄGER, ÜLLE PÄRL, LEINO MÄGI, MARIANNA KAAT, MARGUS PÄRSIK, LAURI VAIKSAAR, ANDRES INGERMAN, HENRY SINIVEE, HILJE SA-VOLAINEN, MARGOT ROOSE, LARS UUS, KRISTA SILDOJA, SIGRIT ROOSILEHT, ANU UIBO, TIIT PAPP, KAARE PÕDER, STANISLAV TITUŠKO, TIIA PIIRIMEES, MAREK KARM, MARKO NUMMERT, MIKK MÄESAAR, SANDRA JÄRV, KARIN JÕKS, AVE TALU, MAARIKA KUKK, IVAR SOONE, VALERI IVANOV, ANNE KAHRU, HILLAR JOON, MIH-HAIL DERBNEV, TOOMAS TEPOMES, ÜLLE KASK, PIRET KÜRBIS, ARKO OKK, ISMAIL MIRZOJEV, AHTO NEIDEK, HERKO SUNTS, MAR-TIN VAHIMETS, MATI TIIMUS, HANNO PÜTTSEPP, MAARIKA NEILINN, ALEKSANDR MAIOROV, RAIVO MERERAND, VIKTOR GOLUBJAT-NIKOV, RAUL ALLIKSAAR, VÄINO UIBO, TRIIN SIHVER, MARGIT LAIL, ANDRI SIMO, AHTI KALLASTE, EDUARD FINAGIN, JAAK RAIE, KIRILL AVDEJEV, JEVGENI GORLEVSKI, NADEŽDA GRETŠNEVKINA, KATRIN JÕGEVA, MONIKA SIRULI, KAROLIINA OONA RANNE, TATJANA JÜR-GEN, TARMO VALGEPEA, RONALD LAARMAA, TANEL TEIN, EPP PET-RONE, KEITH SIILATS, TARMO TAMM, PRIIT ALAMÄE, MARGUS TSAHKNA
Erakond Eestimaa Rohelised	ZÜLEYXA IZMAILOVA, KAI KÜNNIS-BERES, ALEKSANDER LAANE, JAKKO VÄLI, INGA RAITAR, KASPAR KURVE, MARKO KAASIK, ANU NATALY SAAGIM-RATIA, MATTIAS TUROVSKI, OLEV-ANDRES TINN, JOONAS LAKS, ANDRES JAADLA, RENE KUULMANN, TIMUR SAGITOV, HANNES PUU, HELINA TILK, HEIKI VALNER, ULVI KOOP, RASMUS LAHTVEE, KÜLLIKE REIMAA, JÜRI GINTER, TARMO ANDRE ELVISTO, TIIU-ANN KALDMA, ZOJA MELLOV, KRISTJAN ROHTLA, STEVE TRUUMETS, MARET MERISAAR, OLGA GUDKOVA, MIHKEL TIGANIK, ÜLLE RINGMÄE, ARVI TAPVER, FIDEELIA-SIGNE ROOTS, KRISTER

Party	Candidates
	KIVI, MARGIT ADORF, AGU KIVIMÄGI, REET HÄRMAT, AUL PEDAJAS, URVE MADAR, ARVI MATVEI, TÕNU TRUBETSKY, DARJA VORONT-SOVA, ALLAN JAAKUS, HENRY LAKS, LAURA KIIROJA, MÄRT PÕDER, ANNE VETIK, JAANUS NURMOJA, KATRIN JÕGISAAR, TIIU ROOSMA, EERO UUSTALU, LAURA LISETE ROOSAAR, OSKAR-ALEKSANDER LESMENT, CHRISTINA RANNIT, RAUL KÜBARSEPP, MARGARITA RANDVIIR, ARVO KUIV, ANDREI GUDIM, HANNES LIITMÄE, MAIGI KÄIGE, ANNE VAHEMÄE, ANDREI KIISLER, INGA PLOOMIPUU, EVELI PERMANSON, VEERA KOSTINA, ANETTE-MARLEN OTT, TIINA RAM-MUL, RIINA PERNAU, JAAN OLARI, HARRI TILK, SVEN ARULAID, MERIKE KURVITS, KADRI SIKK, KARIN PÕLDOJA, MEIDI TIIMUS, KRISSIKA KAST, REIGO LEHTLA, ALEKSEI NEHOROŠKIN, ERKO ELMIK, TAAVO SAVIK, MOONIKA MALMRE, JOHANNES SARAPUU, MARTTI PREEM, ALEKSANDER POTOTSKI, ERROL VARES, KATRIN LE-HTVEER, LEONID MIRONOV, JÜRI-JOONAS KEREM, ALO KAASIK, RUTH TAMMEORG, PIRET KÄIGE, KRISTA LEHARI, KRISTINA SADOH-HINA, ALLAN TALU, ANNE ARTUS, PAVEL KRÕLATOV, ELO-MARIA ROOTS, HELLE KAASIK, EVE ROOTS, MERILY VAKKER, GERT HÄUSLER, ANNALIISA ASVEIT, MAKSIM ERIKSSON, VOOTELE VAHER, ERKO SOOPALU, HANNES METS, MAI MIKK, KALLE OLUMETS, AARE VEEDLA, KRISTEN KANNIK, RIINA ODNENKO, ELEONORA LOGINOVA, HELI KUUSE, ERKI SAKSING, STINA RAMMUL, ANU PALUOJA, MIKK PÄRNITS, LOORE HÄRMAT, IVO-HANNES LEHTSALU, TIIA KÕNNUS-SAAR, ELMER JOANDI, TIINA KILKSON, ROMAN MUTONEN, REA SEP-PING, ANDRES MEESAK, ALEKSEI LOTMAN
Isamaa Erakond	MARKO POMERANTS, KAIA IVA, RAIVO TAMM, MART LUIK, HELEN HÄÄL, TARMO KRUUSIMÄE, TÕNIS LUKAS, MART NUTT, TOOMAS TÕNISTE, LAURI VAHTRE, KARL SANDER KASE, MART MAASTIK, ÜL-LAR SAAREMÄE, MERLE JÄÄGER, MADIS PÄTS, PRIIDU PÄRNA, KALEV VAPPER, MARI-ANN KELAM, TIIT MATSULEVITŠ, EINAR VALLBAUM, KALLE MUULI, ELA TOMSON, TIIT SALVAN, EVA KAMS, ELLE KULL, GEORG LINKOV, TAMARA VAHTRA-AASMETS, MARJA-LIISA VEISER, PRIIT VÄRK, TIIT LÄÄNE, MEELIS MÕTTUS, LAURI LUUR, KRISTIAN TASKA, HEIKI HEPNER, KAIDO KAASIK, MIHKEL JUHKAMI, ANDRES NOORMÄGI, ALAR KARU, EVA-LIISA LUHAMETS, ULLA PREEDEN, SIIM SUURSILD, AVO ÜPRUS, TRIVIMI VELLISTE, TOOMAS SCHMIDT, ALARI KIRT, ARTUR PÕLD, JÜRI ELLRAM, IVAR TEDREMA, ANNELI KANNUS, KURMET MÜÜRSEPP, MEELIS KUKK, LAGLE PAREK, ANDRES ÖPIK, ANDRES LAISK, ANTI TOPLAAN, TÕNIS PRULER, VALENTINA GUROVA, JANIKA GEDVIL, RAIN SANGERNEBO, PRIIT HUMAL, ARNE TILK, KUNO ERKMANN, TIIT MEREN, JÜRI TREI, URMAS MARDI, MADIS KÜBAR, TIIT HARJAK, AIVO TAMM, KASPAR KAARJAS, JAANA PUUR, JAAN VAIKSAAR, KALLE VISTER, JAANUS PÕLDMAA, PILLE LILL, TOIVO JÜRGENSON, ANDRES ERGMA, KAUPO RÄTSEPP, TÕNU MUNK, ARNO STRAUCH, VALTER VAHA, AARE AN-DERSON, FRIEDRICH KAASIK, TOOMAS PIIRMANN, URMAS AAVA, IMRE RAMMUL, ANNE EENPALU, GERLI LEHE, JANEK RAIK, HER-MANN KALMUS, HELDUR LÄÄNE, TARVO SIILABERG, AIN PAJO, AARE ARVA, PEETER VÕSU, JAAK AHELIK, INDREK LUBERG, ANDRES KAARMANN, KASPAR KOKK, MONIKA ROGENBAUM, VOOTELE HANSEN, ILLIMAR LEPPIK VON WIRÉN, PRIIT PÕLDMÄE, JAKO KULL, LEA NILSON, VILLU KARU, HELDUR-VALDEK SEEDER, ANN RÄÄMET, TIIT NABER, MART RANNUT, INGO NORMET, ARVI KAROTAM, LEEMET VAIKMAA, TANEL OTS, HELIR-VALDOR SEEDER, JÜRI LUIK, VIKTORIA LADÕNSKAJA-KUBITS, SVEN SESTER, PRIIT SIBUL, URMAS REINSALU, ANDRES METSOJA, AIVAR KOKK, SIIM VALMAR KIISLER,

Party	Candidates
	MIHHAIL LOTMAN, RIINA SOLMAN, RAIVO AEG
Sotsiaaldemokraatlik Erakond	JAAK ALLIK, AIN BÖCKLER, MATI ÕUNLOO, TÕNU OJA, HARDI VOLMER, SVEN MIKSER, RAINER VAKRA, MADIS KALLAS, KATRI RAIK, HELMEN KÜTT, HELJO PIKHOF, IVARI PADAR, INDREK TARAND, EIKI NESTOR, MARJU LAURISTIN, JEVGENI OSSINOVSKI, MARINA KALJURAND, KALVI KÕVA, RIINA SIKKUT, INDREK SAAR, LAURI LÄÄNEMETS, MONIKA HAUKANÕMM, RENE TAMMIST, ANAS-TASSIA KOVALENKO, TOOMAS JÜRGENSTEIN, REILI RAND, MARI-ANNE MIKKO, JÜRI MOROZOV, REIN RANDVER, HELVE SÄRGAVA, TANEL TALVE, BARBI-JENNY PILVRE-STORGARD, KALEV KALJUSTE, NEEME SUUR, ENE AUGASMÄGI, MAKSIM ILJIN, MARIKA SAAR, JAAK JUSKE, TRIIN TOOMESAAR, MART MERI, KADRI KÕUSAAR, KAIDO SIPELGAS, KELLY KONETSKI-RAMUL, EDUARD ODINETS, SIRET PI-HELGAS, PRIIT LOMP, GEA KANGILASKI, REIN JÄRVELILL, KADRI-AIJA VIIK, SIIM TUISK, MADLE LIPPUS, OLEV REMSU, PILLE PETERSOO, HERGO TASUJA, KAIRIT PIHLAK, ANTON PRATKUNAS, PIRET AUS, MARGUS MÖLDRI, MEELIS LUHT, SIRJE TOBRELUTS, MADIS VESKIMÄGI, ERIKA SCHOLLER, ANDREI HVOSTOV, KÜLLI URB, JÜRI-SAIMON KUUSEMETS, KAIRIT LINDMÄE, EERIK LUMISTE, TATJANA OLESK, JUHAN-MART SALUMÄE, JAAN SÕRRA, LEMMIT KAPLINSKI, INARA LUIGAS, JAAN KRINAL, MARIS SILD, KARL-MAR-TIN SINIJÄRV, OLGA SÕTNIK, TÕNIS BLANK, SILVI TEESALU, MAR-GUS PUNANE, ETTI KAGAROV, KÜLVAR MAND, MATI KEPP, OTT MAIDRE, MAIRE MURUMAA, MERCEDES MERIMAA, JOOSEP VIMM, KRISTA KAMPUS, ANTO LIIVAT, MERIKE METSTAK, HEIKI HANSO, GARRI RAAGMAA, SERGEI SOLOVJOV, TAMBET SOVA, MAANO KOE-METS, ASKO TAMME, TARMO TAMM, REELIKA RÜÜTLI, JEVGENI VAISBEIN, KRISTEL RANNAÄÄRE, JAAK KANGILASKI, ILJA BORODKIN, ANTS KUTTI, KAAREL ORUMÄGI, JARNO LAUR, ANTS JO-HANSON, KAUPO KUTSAR, TATJANA JAANSON, ROY STRIDER, MAIMU BERG, SILVER MEIKAR, OTT VALDMA, PAVEL PROKOPENKO, TONIO TAMRA, TÕNU INTS, ANTI ALLAS, ANET TOMAŠPOLSKAJA, ANDRUS VAARIK, KIRILL KLAUS, MERLI REIDOLF, AIN PINNONEN, GERTRUD KASEMAA, MIHHAIL BELJAJEV, MARK TENIN, ALLAN KAL-JAKIN, MADIS ROODLA

Appendix 2 – Random 50 samples from training dataset

Speaker name	Source	Source URI	Valid
MAIGI KÄIGE	YouTube	https://youtu.be/cmEwga-h0E4	YES
MARTIN REPINSKI	ERR	http://arhiiv.err.ee/guid/148571	YES
TIINA RÜHKA	YouTube	https://youtu.be/W-wEBqJDcNg	YES
MARINA KALJURAND	YouTube	https://youtu.be/RgUz2aM1_fk	NO
JÜRI RATAS	ERR	http://arhiiv.err.ee/guid/124856	YES
TOOMAS KIVIMÄGI	ERR	http://arhiiv.err.ee/guid/82173	YES
TAAVI AAS	ERR	http://arhiiv.err.ee/guid/101949	YES
ANTS JOHANSON	ERR	http://arhiiv.err.ee/guid/107767	YES
RAIVO PÕLDARU	YouTube	https://youtu.be/RLpj99O9RX4	NO
PEETER LAASIK	YouTube	https://youtu.be/WnJ9F7s8fpA	YES
LAURI HUSSAR	ERR	http://arhiiv.err.ee/guid/107373	YES
MIHHAIL STALNUHHIN	ERR	http://arhiiv.err.ee/guid/105432	YES
JAAN OLARI	YouTube	https://youtu.be/i2whG_weevM	NO
LAURI HUSSAR	ERR	http://arhiiv.err.ee/guid/44522	YES
HERMANN KALMUS	YouTube	https://youtu.be/0olk51DnSHU	YES
INDREK SAAR	ERR	http://arhiiv.err.ee/guid/57557	YES
ANTI TOPLAAN	ERR	http://arhiiv.err.ee/guid/125355	YES
URMAS KRUUSE	ERR	http://arhiiv.err.ee/guid/61579	YES
MARGIT SUTROP	ERR	http://arhiiv.err.ee/guid/44092	YES
MARTIN HELME	YouTube	https://youtu.be/2Tetur5k5L8	NO
VIIVE ROSENBERG	ERR	http://arhiiv.err.ee/guid/55573	YES
HANNO PEVKUR	YouTube	https://youtu.be/mGG-DYmZrvw	YES
JEVGENI OSSINOVSKI	ERR	http://arhiiv.err.ee/guid/39648	YES
MARGUS TSAHKNA	ERR	http://arhiiv.err.ee/guid/77893	YES
TAUNO JÜRGENSTEIN	YouTube	https://youtu.be/LKgOGXp04GM	NO
URVE TIIDUS	YouTube	https://youtu.be/__9L9eeSBEA	YES
JAAK ALLIK	ERR	http://arhiiv.err.ee/guid/82026	YES
INDREK SAAR	ERR	http://arhiiv.err.ee/guid/86128	YES
SIIM VALMAR KIISLER	ERR	http://arhiiv.err.ee/guid/80410	YES
VIKTOR VASSILJEV	ERR	http://arhiiv.err.ee/guid/52676	YES
HELIR-VALDOR SEEDER	ERR	http://arhiiv.err.ee/guid/59404	YES
SIIRI KÄPA	YouTube	https://youtu.be/oIAddHYNr7U	YES

MARIKE LAHT	YouTube	https://youtu.be/o93XPglv6_0	YES
HENRY LAKS	YouTube	https://youtu.be/oOYhMTVZDk8	YES
URMAS PAET	ERR	http://arhiiv.err.ee/guid/86669	YES
PRIIT SIBUL	ERR	http://arhiiv.err.ee/guid/108975	YES
KRISTEN MICHAL	ERR	http://arhiiv.err.ee/guid/52390	YES
HEIKI HANSO	YouTube	https://youtu.be/ybnhGj46-8Y	YES
JAAK VALGE	YouTube	https://youtu.be/fdrBHzUO_AM	YES
OLEV REMSU	YouTube	https://youtu.be/y8aKkXnTdmA	YES
LIIS KLAAR	ERR	http://arhiiv.err.ee/guid/78860	YES
SERGEI SOLOVJOV	YouTube	https://youtu.be/QalcElO6rIA	YES
OLGA SÕTNIK	YouTube	https://youtu.be/0YmhM2Y_xkM	YES
SIIM KALLAS	ERR	http://arhiiv.err.ee/guid/51942	YES
SVEN MIKSER	ERR	http://arhiiv.err.ee/guid/118317	YES
VILLU KARU	YouTube	https://youtu.be/mI7bZRfUTh8	NO
KALVI KÕVA	YouTube	https://youtu.be/VqhKtsDgiBc	YES
MATI RAIDMA	ERR	http://arhiiv.err.ee/guid/84214	YES
VILJO TAMM	YouTube	https://youtu.be/SNuaemnDLQs	YES
IGOR TARO	ERR	http://arhiiv.err.ee/guid/100796	YES

Appendix 3 – Validation data

Vikerraadio “Valimisstuudio” - 05.02.2019

6 annotated speakers in metadata: Kadri Simson, Maris Lauri, Inga Raitar, Helen Orav-Kotta, Ulla Preeden, Arp Müller. As Arp Müller is the presenter and not in the collection of persons we would like to model, he is not in interest at the moment.

Manually labelled	**Occurrences in training data**	**Manually labelled segments total duration (sec)**	**Speaker Identification Output**	**Recall**	**Correct**
Kadri Simson	325	784	Kadri Simson	Yes	Yes
Maris Lauri	61	879	Maris Lauri	Yes	Yes
Inga Raitar	10	736	Inga Raitar	Yes	Yes
Helen Orav-Kotta	0	512	-	-	-
Ulla Preeden	10	553	-	-	-

Presenters total hand annotated speech segments duration: 1519,7 seconds, 30% of the total show’s duration.

Vikerraadio “Valimisstuudio” - 07.02.2019

6 annotated speakers in metadata: Raimond Kaljulaid, Karl Sander Kase, Tiina Radionov, Katri Raik, Urmas Reitelmann, Mirko Ojakivi. As Mirko Ojakivi is the

presenter and not in the collection of persons we would like to model, he is not in interest at the moment.

Manually labelled	**Occurrences in training data**	**Manually labelled segments total duration (sec)**	**Speaker Identification Output**	**Recall**	**Correct**
Raimond Kaljulaid	24	1094	Raimond Kaljulaid	Yes	Yes
Karl Sander Kase	2	740	-	-	-
Tiina Radionov	0	235	-	-	-
Katri Raik	49	1163	Katri Raik	Yes	Yes
Urmas Reitelmann	0	499	-	-	-

Presenters total hand annotated speech segments duration: 1284,3 seconds, 25,6% of the total show’s duration.

ETV “Valimisstuudio” - 06.02.2019

10 annotated speakers in metadata: Jaak Aab, Taavi Rõivas, Helmen Kütt, Priit Sibul, Tiina Kangro, Helle Kullerkupp, Tiia Sihver, Aleksander Laane, Andres Kuusk, Johannes Tralla. As Andres Kuusk and Johannes Tralla are the presenters and not in the collection of persons we would like to model, they are not in interest at the moment.

Manually labelled	**Occurrences in training data**	**Manually labelled segments total duration (sec)**	**Speaker Identification Output**	**Recall**	**Correct**
Jaak Aab	83	423	Jaak Aab	Yes	Yes

Taavi Rõivas	374	611	Taavi Rõivas	Yes	Yes
Helmen Kütt	43	611	Helmen Kütt	Yes	Yes
Priit Sibul	46	515	Priit Sibul	Yes	Yes
Tiina Kangro	20	531	Tiina Kangro	Yes	Yes
Helle Kullerkupp	1	414	-	-	-
Tiia Sihver	13	332	-	-	-
Aleksander Laane	17	444	Aleksander Laane	Yes	Yes

Presenters total hand annotated speech segments duration: 799,7 seconds, 17,9% of the total show's duration.

ETV "Valimisstuudio" - 02.03.2019

10 annotated speakers in metadata: Jüri Ratas, Kaja Kallas, Mart Helme, Jevgeni Ossinovski, Helir-Valdor Seeder, Kaul Nurm, Kristina Kallas, Züleyxa Izmailova, Andres Kuusk, Johannes Tralla. As Andres Kuusk and Johannes Tralla are the presenters and not in the collection of persons we would like to model, they are not in interest at the moment.

Manually labelled	**Occurrences in training data**	**Manually labelled segments total duration (sec)**	**Speaker Identification Output**	**Recall**	**Correct**
Jüri Ratas	342	756	Jüri Ratas	Yes	Yes
Kaja Kallas	120	597	Kaja Kallas	Yes	Yes

Mart Helme	140	610	Mart Helme	Yes	Yes
Jevgeni Ossinovski	235	624	Jevgeni Ossinovski	Yes	Yes
Helir-Valdor Seeder	173	483	Helir-Valdor Seeder	Yes	Yes
Kaul Nurm	43	530	Kaul Nurm	Yes	Yes
Kristina Kallas	42	472	Kristina Kallas	Yes	Yes
Züleyxa Izmailova	24	568	Züleyxa Izmailova	Yes	Yes

Presenters total hand annotated speech segments duration: 668,1 seconds, 12,5% of the total show’s duration.

Appendix 4 – The metadata for test dataset

Total of 55 shows and 210 speakers, from whom 123 are unique.

Nikolai Degtjarenko was one of the speakers, but as he is not in the collection of speakers who we are modelling in this thesis, he is ignored.

Source	**Show name**	**Airing date**	**Speakers**
Raadio 4	Пять вопросов кандидату	5.02.2019	Mihhail Korb
Raadio 4	Пять вопросов кандидату	6.02.2019	Siim Kallas
Raadio 4	Пять вопросов кандидату	7.02.2019	Jevgeni Ossinovski
Raadio 4	Пять вопросов кандидату	8.02.2019	Leo Kunnas
Raadio 4	Пять вопросов кандидату	9.02.2019	Timur Sagitov
Raadio 4	Пять вопросов кандидату	11.02.2019	Kristina Kallas
Raadio 4	Пять вопросов кандидату	12.02.2019	Oliver Loode
Raadio 4	Пять вопросов кандидату	13.02.2019	Viktoria Ladõnskaja-Kubits
Raadio 4	Пять вопросов кандидату	15.02.2019	Neeme Kuningas
Raadio 2	Riigieksam	02.2019	Kristo Enn Vaga, Kaja Kallas
Raadio 2	Riigieksam	02.2019	Kristina Kallas, Aleksei Jašin
Raadio 2	Riigieksam	02.2019	Ruuben Kaalep, Mart Helme
Raadio 2	Riigieksam	02.2019	Karl Sander Kase, Helir-Valdor Seeder
Raadio 2	Riigieksam	02.2019	Kaul Nurm, Õie-Mari Aasmäe

Source	**Show name**	**Airing date**	**Speakers**
Raadio 2	Riigieksam	02.2019	Julia Sommer, **Nikolai Degtjarenko***
Raadio 2	Riigieksam	02.2019	Jüri Ratas, Ester Karuse
Raadio 2	Riigieksam	02.2019	Jevgeni Ossinvski, Pavel Prokopenko
ETV+	Предвыборная студия	6.02.2019	Mihhail Kõlvart, Marina Kaljurand, Urmas Heinaste, Viktoria Ladõnskaja-Kubits, Kristina Kallas, Andrei Gudim, Julia Sommer
ETV+	Предвыборная студия	13.02.2019	Marko Mihkelson, Raimond Kaljulaid, Toomas Alatalu, Leo Kunnas, Grigore-Kalev Stoicescu, Oleg Tesla, Oliver Loode
ETV+	Предвыборная студия	20.02.2019	Siim Kallas, Jevgeni Ossinovski, Sven Sester, Raivo Kokser, Kersti Kracht, Olev-Andres Tinn, Lauri Tõnspoeg
ETV+	Предвыборная студия	27.02.2019	Jana Toom, Sergei Gorlatš, Katri Raik, Siim Kiisler, Riho Breivel, Timur Sagitov, Ahti Puur
ETV	Valimisstuudio	2.03.2019	Jüri Ratas, Kaja Kallas, Mart Helme, Jevgeni Ossinovski, Helir-Valdor Seeder, Kaul Nurm, Kristina Kallas, Zyleixa Izmailova
ETV	Valimisstuudio	6.02.2019	Jaak Aab, Taavi Rõivas, Helmen Kütt, Priit Sibul, Tiina Kangro, Helle Kullerkupp, Tiia Sihver, Aleksander Laane
ETV	Valimisstuudio	13.02.2019	Mailis Reps, Maris Lauri, Marju Lauristin, Tõnis Lukas, Tiiu Kuurme, Jaak Valge, Lauri Hussar, Jüri Ginter
ETV	Valimisstuudio	20.02.2019	Jaanus Karilaid, Urmas Paet, Sven Mikser, Jüri Luik, Andres Herkel, Leo Kunnas, Margus Tsahkna, Rasmus Lahtvee
ETV	Valimisstuudio	27.02.2019	Joonas Laks, Rene Tammist, Sven Sester, Kadri Simson, Jürgen Ligi, Martin Helme, Priit Alamäe, Jaanus Ojangu
Kuku Raadio	Valimisstuudio	1.03.2019	Aleksander Laane, Andres Herkel, Sven Sester, Urmas Paet

Source	Show name	Airing date	Speakers
Kuku Raa-dio	Valimisstuudio	18.02.2019	Helir-Valdor Seeder, Kristina Kallas
Kuku Raa-dio	Valimisstuudio	19.02.2019	Züleyxa Izmailova, Kaul Nurm
Kuku Raa-dio	Valimisstuudio	20.02.2019	Mart Helme, Jevgeni Ossinovski
Kuku Raa-dio	Valimisstuudio	21.02.2019	Jüri Ratas, Kaja Kallas
Kuku Raa-dio	Valimisstuudio	22.02.2019	Julia Sommer, Artur Talvik
Kuku Raa-dio	Valimisstuudio	25.02.2019	Mihhail Lotman, Aadu Must, Marko Kaasik, Heljo Pikhof, Kristiina Tõnnis-son, Urmas Klaas, Kuido Nõmm, Indrek Särg
Kuku Raa-dio	Valimisstuudio	26.02.2019	Kadri Simson, Toomas Kivimägi, Mart Helme, Indrek Tarand, Andres Metsoja
Kuku Raa-dio	Valimisstuudio	27.02.2019	Ain Ostra, Jürgen Ligi, Hannes Puu, Hel-men Kütt, Jaak Madison, Ruslan Trochynskyi, Jaak Aab, Helir-Valdor Seeder
Kuku Raa-dio	Valimisstuudio	28.02.2019	Raimond Kaljulaid, Lauri Hussar, Martin Helme, Sven Mikser
Raadio 4	Valimisstuudio	18.02.2019	Lauri Laats, Viktoria Ladõnskaja-Kubits, Urmas Espenberg, Svetlana Skrebneva, Aleksander Laane
Raadio 4	Valimisstuudio	19.02.2019	Yana Toom, Mart Luik, Nikita Lumijõe, Toomas Alatalu, Andrei Gudim
Raadio 4	Valimisstuudio	20.02.2019	Jevgeni Ossinovski, Siim Kiisler, Kersti Kracht, Raivo Kokser, Olev-Andres Tinn
Raadio 4	Valimisstuudio	21.02.2019	Yana Toom, Hanno Pevkur, Külli Remsu, Kristina Kallas, Julia Sommer
Raadio 4	Valimisstuudio	25.02.2019	Jevgeni Ossinovski, Eero Merilind, Jaanika Klopets, Ago Samoson, Igor Rosenfeld
Raadio 4	Valimisstuudio	26.02.2019	Marko Mihkelson, Marianne Mikko, Leo Kunnas, Artur Talvik, Oleg Tesla

Source	**Show name**	**Airing date**	**Speakers**
Raadio 4	Valimisstuudio	27.02.2019	Katri Raik, Viktoria Ladõnskaja-Kubits, Kaja Toikka, Andrei Gudim, Neeme Kuningas
Raadio 4	Valimisstuudio	28.02.2019	Siim Kallas, Raimond Kaljulaid, Jana Pavlenkova, Lauri Tõnspoeg, Andrei Anissimov
Vikerraadio	Valimisstuudio	5.02.2019	Kadri Simson, Maris Lauri, Inga Raitar, Helen Orav-Kotta, Ulla Preeden
Vikerraadio	Valimisstuudio	7.02.2019	Raimond Kaljulaid, Karl Sander Kase, Tiina Radionov, Katri Raik, Urmas Reitelmann
Vikerraadio	Valimisstuudio	12.02.2019	Siret Kotka-Repinski, Jaak Valge, Priit Alamäe, Lauri Tõnspoeg, Rene Tammist
Vikerraadio	Valimisstuudio	14.02.2019	Urmas Kruuse, Marika Parv, Rainer Kuuba, Andres Metsoja, Igor Taro
Vikerraadio	Valimisstuudio	19.02.2019	Züleyxa Izmailova, Kaul Nurm
Vikerraadio	Valimisstuudio	20.02.2019	Kristina Kallas, Helir-Valdor Seeder
Vikerraadio	Valimisstuudio	21.02.2019	Jevgeni Ossinovski, Mart Helme
Vikerraadio	Valimisstuudio	22.02.2019	Kaja Kallas, Jüri Ratas
Vikerraadio	Valimisstuudio	30.01.2019	Viivi-Helbe Peljuhhovska, Hanno Pevkur, Aleksander Laane, Kuido Nõmm, Riina Sikkut
Vikerraadio	Valimisstuudio	31.01.2019	Ruuben Kaalep, Aleksei Jašin, Sander Klausen, Andrei Orponen, Laura Lisete Roosaar

Appendix 5 – Python Code for Training the Model

```
#! /usr/bin/env python3.5

import argparse
import kaldi_io
import random
import numpy
from collections import Counter, OrderedDict
import tempfile
import os
import codecs
import pdb
import torch
from torch.utils.data import Dataset, DataLoader
import torch.nn as nn
from torch.autograd import Variable
import torch.nn.functional as F
from sacred import Experiment

ex = Experiment('valimised_dnn')


@ex.config
def my_config():
    save_model = None
    min_spk_occ = 8
    num_epochs = 30
    dev_minimum_precision = 0.95
    initial_learning_rate = 0.01
    final_learning_rate = 0.001
    hidden_dim = 768
    dropout = 0.2
    train_ivector_file = 'exp/ivectors_100k_2048/train_segmented_combined/
spk_ivector.scp'
    wav2poi_file = 'data/train_segmented_combined/wav2poi'
    wav2spk_file = 'data/train_segmented_combined/wav2spk'
    dev_ivector_file = 'exp/ivectors_100k_2048/dev/spk_ivector.scp'


def label_reg_loss(y_pred, y_true_mean, class_weights, eps=1e-10):
    y_pred = y_pred.clamp(eps, 1)
    y_true_mean = y_true_mean.clamp(eps, 1)
    y_pred_mean = torch.mean(y_pred, dim=0)
    return torch.sum(class_weights * y_true_mean * torch.log(y_true_mean /
y_pred_mean + eps), dim=-1)


def init_weights(m):
    if type(m) == nn.Linear:
        torch.nn.init.xavier_uniform_(m.weight)
        m.bias.data.fill_(0.01)


class SidDnn(nn.Module):

    def __init__(self, feature_dim, output_dim, hidden_dim,
dropout_prob=0.2):
        super(SidDnn, self).__init__()
        self.layers = nn.Sequential(
            nn.Linear(feature_dim, hidden_dim),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout(dropout_prob),
            nn.Linear(hidden_dim, hidden_dim),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout(dropout_prob),
            nn.Linear(hidden_dim, output_dim)
        )

    def forward(self, x):
        probs = F.softmax(self.layers(x), dim=-1)
        return probs


def matches(name1, name2):
    return name1.replace("-", "_") == name2.replace("-", "_")


@ex.automain
def run(save_model, min_spk_occ, num_epochs, dev_minimum_precision,
        initial_learning_rate, final_learning_rate, hidden_dim, dropout,
        train_ivector_file, wav2poi_file, wav2spk_file, dev_ivector_file):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    wav2spk = {}
    spk2wav = {}
    for l in open(wav2spk_file):
        ss = l.split()
        wav2spk[ss[0]] = ss[1:]
        for spk in ss[1:]:
            spk2wav[spk] = ss[0]

    wav2poi = {}
    poi2wav = {}
    for l in open(wav2poi_file):
        ss = l.split()
        wav = ss[0]
        poi = ss[1]
        wav2poi.setdefault(wav, []).append(poi)
        poi2wav.setdefault(poi, []).append(wav)

    # keep names that occur at least min_spk_occ times across all shows
    pruned_poi_set = set([a[0] for a in poi2wav.items() if len(a[1]) >=
min_spk_occ])
    print("%s speakers left after pruning" % len(pruned_poi_set))

    print("Reading dev i-vectors")
    dev_ivecs = []
    dev_ivec_input = 'copy-vector scp:%s ark:- |' % dev_ivector_file
    for key, vec in kaldi_io.read_vec_flt_ark(dev_ivec_input):
        dev_ivecs.append((codecs.decode(bytes(key, "latin1")), vec))

    print("Reading train i-vectors")
    wav2ivecs = {}
    train_ivec_input = 'copy-vector scp:%s ark:- |' % train_ivector_file
    for key, vec in kaldi_io.read_vec_flt_ark(train_ivec_input):
        wav = spk2wav[key]
        if not wav in wav2ivecs:
            wav2ivecs[wav] = numpy.array([vec])
        else:
            wav2ivecs[wav] = numpy.append(wav2ivecs[wav], [vec], axis=0)

    # pruned_poi_list is a list of all names left after pruning, plus <unk>
    # poi_ids is a dict that maps names to their indexes in pruned_poi_list
    pruned_poi_list = []
    pruned_poi_list = ["<unk>"]
    pruned_poi_list.extend(sorted(pruned_poi_set))
    poi_ids = {}
    for poi in pruned_poi_list:
        poi_ids[poi] = len(poi_ids)

    # poi_ids_in_shows is a dict that maps show IDs to sets that contain
    # all name IDs in that show, with a special ID for <unk> for
    # pruned-out speakers
    poi_ids_in_shows = {}
    unk_count_in_shows = Counter()
    for show, pois in wav2poi.items():
        poi_ids_in_show = set()
        for poi in pois:
            if poi in poi_ids:
                poi_ids_in_show.add(poi_ids[poi])
            else:
                unk_count_in_shows[show] += 1
        poi_ids_in_shows[show] = poi_ids_in_show

    poi_id_count = torch.tensor([len(poi2wav.get(pruned_poi_list[i], [])) for
i in range(len(pruned_poi_list))])
    poi_id_count[poi_ids["<unk>"]] = sum([unk_count_in_shows[show] for show
in wav2poi])

    weight_magnitude = 0.3

    poi_learn_weights = ((sum(poi_id_count).float() / (len(poi_ids) *
poi_id_count.float())) ** weight_magnitude).to(
        device)

    def validate(model):
        dev_ivecs_arr = numpy.array([i[1] for i in dev_ivecs])
        dev_predicted_target_probs = model.for-
ward(torch.from_numpy(dev_ivecs_arr).to(device))
        dev_predicted_speaker_ids = dev_predicted_target_probs.argmax(dim=1)
        tp = [0] * 100
        fp = [0] * 100
        tn = [0] * 100
        fn = [0] * 100

        for i in range(0, 100):
            confidence_threshold = i / 100.0
            for (j, speaker_id) in enumerate(dev_predicted_speaker_ids):
                true_name = dev_ivecs[j][0]
                if matches(true_name, pruned_poi_list[speaker_id]) and
dev_predicted_target_probs[
                    j, speaker_id] >= confidence_threshold and
pruned_poi_list[speaker_id] != "<unk>":
                    tp[i] += 1
                elif (not matches(true_name, pruned_poi_list[speaker_id]))
and dev_predicted_target_probs[
                    j, speaker_id] >= confidence_threshold and
pruned_poi_list[speaker_id] != "<unk>":
                    fp[i] += 1
                else:
                    fn[i] += 1

        for i in range(0, 100):
            if tp[i] + fn[i] > 0 and tp[i] + fp[i] > 0:
                recall = 1.0 * tp[i] / (tp[i] + fn[i])
                precision = 1.0 * tp[i] / (tp[i] + fp[i])
                if precision >= dev_minimum_precision:
                    print("Best recall at minimum precision %f (%f),
threshold %f: %f" % (
                        dev_minimum_precision, precision, i / 100.0, recall))
                    return recall

        return 0.0

    # Compute oracle coverage
    num_covered = 0
    for i in dev_ivecs:
        dev_name = i[0]
        if any([matches(dev_name, poi) for poi in pruned_poi_list]):
            num_covered += 1
        else:
            print("%s not in train data" % dev_name)
    coverage = 1.0 * num_covered / len(dev_ivecs)
    print("Oracle name coverage on dev data: %f" % coverage)

    input_dim = list(wav2ivecs.items())[0][1].shape[1]
    model = SidDnn(input_dim, len(poi_ids), hidden_dim, dropout).to(device)
    optimizer = torch.optim.SGD(model.parameters(), lr=initial_learning_rate)
    scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'max',
factor=0.5, verbose=True,

min_lr=final_learning_rate, patience=2)

    best_recall = 0.0
    handle, f_name = tempfile.mkstemp()

    for epoch in range(num_epochs):
        # set training mode
        model.train()
        # Train the DNN, show-by-show
        for show, pois in random.sample(wav2poi.items(), k=len(wav2poi)):
            if show not in wav2ivecs:
                continue

            ivecs_for_show = wav2ivecs[show]
            # Label proportions: uniform over the names in that show
            num_ivecs = len(ivecs_for_show)
            if num_ivecs > 0:
                label_props_for_show = torch.zeros((len(poi_ids))).to(device)
                label_props_for_show[list(poi_ids_in_shows[show])] = \
                    1.0 / num_ivecs
                label_props_for_show[poi_ids["<unk>"]] = \
                    1.0 * (num_ivecs - len(poi_ids_in_shows[show])) /
num_ivecs

                ivecs_for_show_device =
torch.from_numpy(ivecs_for_show).to(device)
                model.zero_grad()
                probs = model.forward(ivecs_for_show_device)
                loss = label_reg_loss(probs, label_props_for_show,
poi_learn_weights)

                loss.backward()
                optimizer.step()

        print("Finished epoch %d" % epoch)
        # set evaluation mode
        model.eval()
        recall = validate(model)
        if recall > best_recall:
            print("New best model, saving it")
            torch.save(model, f_name)
            best_recall = recall

        scheduler.step(recall)

    print("Finished training")

    ex.add_artifact(f_name, name="model.torch")
    os.remove(f_name)
    with open("%s.names" % f_name, "wt", encoding='utf-8') as f:
        for name in pruned_poi_list:
            print(name, file=f)
    ex.add_artifact("%s.names" % f_name, name="model.names")
    os.remove("%s.names" % f_name)
```