TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Jörgen Valk 183404IAPM

USING WEB SCRAPING FOR BUILDING SPOKEN LANGUAGE IDENTIFICATION MODELS

Master's Thesis

Supervisor: Tanel Alumäe
PhD

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Jörgen Valk 183404IAPM

ANDMETE MASSKOGUMINE KÕNELDAVA KEELE TUVASTAMISE MUDELITE LOOMISEKS

Magistritöö

Juhendaja:	Tanel Alumäe
	PhD

Tallinn 2020

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: Jörgen Valk

13.05.2020

Abstract

Spoken language identification has many applications from speech recognition systems to data analysis. The identification systems are commonly created using neural networks that require a lot of training data. There are some existing datasets, specifically created for the task, but most of them have problems like being quite costly, not containing the required languages, etc. This thesis investigates the use of automatically collected web audio data for the task of spoken language identification.

To collect the web audio datasets a data collection pipeline has to be developed. The required speech data is collected from the YouTube platform. Suitable content is found by using a set of pre-generated search phrases that have been created from Wikipedia data. Automatically collected information inevitably contains much noise and false positive results, therefore heavy filtering has to be applied during the data collection process. As a result from the data collection step a large dataset is built, containing data for over 100 languages.

The spoken language identification models are built with neural networks and use the collected datasets. The models follow the state-of-the-art x-vector approach for language identification. Additionally, different noise robust loss functions are used to deal with the noisy labels from web data. Results show, that the proposed solution can be very effectively used for language identification and the models perform well.

The thesis is written in English and contains 61 pages of text, 8 chapters, 30 figures and 6 tables.

Annotatsioon

Andmete masskogumine kõneldava keele tuvastamise mudelite loomiseks

Kõneldava keele tuvastusel on rakendusi mitmes erinevas valdkonnas, alates kõnetuvastussüsteemidest ja lõpetades andmeanalüüsiga. Selliste keeletuvastussüsteemide loomiseks kasutatakse tänapäeval tihti närvivõrke, mille treenimiseks on vaja suurel hulgal andmeid. Keeletuvastuse jaoks on olemas mõned andmekorpused, kuid paljudel neist on probleeme. Näiteks on need liiga kallid, vajaminevad keeleandmed puuduvad või on andmed täiesti eri domeenidest. Antud magistritöö eesmärgiks on uurida automaatselt internetist kogutud massandmete kasutamist kõneldava keele tuvastamise mudelite loomisel.

Andmestike kogumiseks on vaja luua tööriistad, mis võimaldaksid seda automaatselt teha. Kõneandmed kogutakse YouTube'i platvormilt kastades eelnevalt genereeritud otsingufraase. Otsingufraasid luuakse mitmekeelsetest Vikipeedia andmetest. Automaatselt internetist kogutud andmed sisaldavad paratamatult palju müraseid ja soovimatuid tulemusi. Nende välja filtreerimiseks rakendadakse andmekogumisprotsesis mitmeid erinevaid meetmeid, et lõpptulemuses oleks võimalikult sobivad andmed. Lõpuks kogutakse suur andmekorpus, rohkem kui 100-le keelele, mida kasutatakse keeletuvastusmudelite treenimiseks.

Keeletuvastusmudelid luuakse kasutades närvivõrke ja eelnevalt kogutud andmeid. Mudelid on koostatud viimasel ajal keeletuvastuse valdkonnas parimaid tulemusi saavutanud x-vektorite meetodi järgi. Kuna tegemist on osaliselt müraste veebiandmetega, siis rakendatakse ka tuvastusmudelites erinevaid müraga arvestavaid kaofunktsioone, mis peaksid mudeli tulemusi parandama. Lõpptulemused näitavad, et sellisel kujul veebiandmete kasutamine kõneldava keele tuvastamiseks on täiesti võimalik ja mudelite tulemused on head.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 61 leheküljel, 8 peatükki, 30 joonist ja 6 tabelit.

List of abbreviations and terms

AAM	Additive Angular Margin
API	Application Programming Inteface
CE	Cross Entropy
DET	Detection Error Tradeoff
DNN	Deep Neural Network
GMM	Gaussian Mixture Model
GPU	Graphical Processing Unit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
LRE	Language Recognition Evaluation
MAE	Mean Absolute Error
MAP	Maximum Posterior Probability
MFCC	Mel Frequency Cepstral Coefficent
MSE	Mean Squared Error
MUSAN	Music Speech And Noise (corpus)
NIST	National Institute of Standards and Technology
NLL	Negative Log Likelihood
OOS	Out Of Set
OSCAR	Open Super-large Crawled Almanach coRpus
ReLU	Rectified Linear Unit
RIR	Room Impulse Reponse
SVM	Support Vector Machine
TDNN	Time Delay Neural Network
TF-IDF	Term Frequency Inverse Document Frequency
UBM	Universal Background Model
VPN	Virtual Private Network
VPS	Virtual Private Server

WAV	Waveform Audio File
XML	eXtensible Markup Language

Table of contents

List of figures

List of tables

1 Introduction

This thesis investigates the use of automatically collected web data, such as YouTube videos for building spoken language identification models with neural networks. The work can be divided into two main parts. Firstly, the process of collecting the required datasets. Secondly, the use of such datasets for creating the language identification models. In the work a data collection pipeline for YouTube videos is created, a data validation process is conducted and spoken language identification models are built and tested.

1.1 Problem

Nowadays, neural networks are frequently used to solve problems from a wide variety of fields. Such networks can only be built if there is enough annotated data that can be used for training the models. Such datasets are also required for tasks in the language domain. In order to create a language identification model that is able to identify the spoken language from a speech segment, datasets are required for each of the target languages. They have to contain hours of audio clips with speech data. There are some existing datasets created for the task, but they often contain resources for only a few of the required languages or data for the language that the model needs to be able to identify is not present at all. For example, since this thesis is written in Estonia, then the Estonian language is not present in any of such large datasets for language identification. If datasets for smaller languages like Estonian exist, then they usually have to be acquired separately from different sources to use for the identification task. Other drawbacks of the larger available datasets are that they are quite expensive and some of them contain speech data from only one specific domain.

To solve the problem, the use of automatically collected web data for creating spoken language identification models is investigated in this thesis. The problems solved in this work can be divided into two parts - the dataset collection and building the language identification models.

Firstly, in the data collection step, an effective way needs to be developed for automatically collecting the speech data required for the identification models. Since the datasets need to be collected for many different languages, a universal collection process has to be developed. Data for the neural networks has to be annotated, meaning that all of the collected data has to have language labels associated with it. Data that is automatically collected from the internet inevitably contains noise, false language labels and other problems that also need to be addressed in this step.

In the second step an effective way needs to be found to use YouTube data for language identification in neural networks. Some additional techniques and methods have to be applied in order to deal with the potentially noisy and falsely labelled training data.

1.2 Objective

The first objective is to collect large multilingual datasets for the purpose of language identification. During that step an automatic data collection and filtering process needs to be developed. The YouTube platform is investigated as the primary resource for the speech data. A validation experiment will be carried out on the collected datasets to find out the quality of such automatically collected data and to create separate datasets that can be used for validation when creating the identification models.

Secondly, the goal is to effectively use such web data for the spoken language identification models. The models will be validated on data from the previous step and some external datasets. The goal is to achieve similar language identification accuracy to other models trained with already available datasets.

1.3 Outline

The first chapter describes the task of spoken language identification and the problems with existing datasets, gives an overview of the goals and talks about the proposed solution for the problem.

The second chapter gives an overview of previous work that is related to this thesis. The problem of spoken language identification is described in more detail and also background topics like neural networks, i-vectors and x-vectors are shortly discussed.

The third chapter gives an in depth overview of the dataset collection process. The initial data requirements and available data sources are described. The video collection and audio extraction process is shown and finally an overview of the collected datasets is given.

The fourth chapter covers the data validation process that is conducted on the collected datasets. An overview of the validation process is given and then the findings from the results are discussed.

The fifth chapter describes the neural network based language identification models and shows some of the used noise robust loss functions.

The sixth chapter presents the results achieved by the spoken language identification models. The models are evaluated on some of the validated data from previous steps and also some external datasets.

The seventh chapter discusses some of the areas where current work can be improved on further and what are the larger related topics that can be investigated in the future.

The last chapter concludes the work that was done and presents the most important achieved results.

2 Background

The following sections give an overview of some previous work that is related to this thesis. For example, datasets and projects where YouTube data has been used for language identification or other tasks in the same domain will be described. Background theory and methods like i-vectors, x-vectors and neural networks will also be shortly covered.

2.1 Related work

The use of YouTube data is not new for the machine learning community. Datasets have been created for many different tasks using the video and audio resources available on YouTube. For example, a lot of such work has been done in the computer vision and language processing domain. For the specific task of spoken language identification, the use of YouTube data has been less popular. There has been work, where YouTube data was used for language identification, but the amount of collected languages and the dataset sizes were quite modest. Also, in many cases the YouTube data has not been the primary data source for building the models. In the work most similar to what is done in this thesis, YouTube data was used for tuning and testing the neural networks that had previously been trained on TV broadcast data instead.

2.1.1 VoxCeleb

Similar work of using YouTube data in the audio-visual domain has been previously done by the creators of VoxCeleb [32] dataset for speaker verification and identification tasks. Since the data required for speaker identification and verification used to mostly be hand annotated and therefore limited in size, the authors of VoxCeleb proposed an automatic pipeline for collecting data from open-source media like YouTube.

The created pipeline combined techniques from computer vision and spoken language domain. The initial step in the process they used was to collect the celebrity names from various sources. Then the top videos related to each of the celebrity names were automatically downloaded from YouTube. Some extensive face tracking and speaker and face verification tasks followed. Finally, a dataset containing over 100 000 speech utterances for 1251 speakers was formed [32].

VoxCeleb has been widely used in the machine learning community and proven to be very valuable. Aside from its original goal of speaker verification and identification it has been used for many other tasks, like face generation, face synthesis and emotion recognition, making it even more versatile than originally intended.

2.1.2 KALAKA-3

In this thesis the use of YouTube audios for building language identification models can most closely be compared with the KALAKA-3 database. KALAKA-3 is a database based on YouTube data and built for identifying European languages to support the Albayzin 2012 Language Recognition Evaluation (LRE) that was organized by the Spanish Thematic Network on Speech Technologies [44], [43]. In the work by Rodríguez-Fuentes *et al.* TV broadcast speech was combined with automatically collected data from YouTube and used for the purpose of spoken language recognition.

In the data collection stage the goal for building KALAKA-3 was to collect around 300 YouTube audio clips for 21 target languages. The languages themselves were distributed into different groups. There were six main target languages where plenty of training data was available, four empty-training languages for which no training data was available and 11 Out Of Set (OOS) languages.

YouTube data was collected from six predefined categories (Education, News, Entertainment, How-to, Nonprofit and Technology) and a list of videos was created for each of the target languages by using the YouTube API. The videos were queried with keywords formed by using the aspell[1] dictionary for each language, choosing 2000 random words and doing some additional processing. To increase the chance of videos having the expected spoken language, only content with available geographical information was used. Such approach seems like a good solution to finding only the correct videos, but as the

[1] https://ftp.gnu.org/gnu/aspell/dict/0index.html

work in this thesis shows, such hard limits usually decrease the amount of search results drastically because only a small subset of YouTube videos actually contain such metadata and for languages with less or lower grade YouTube video content there may be no results at all.

After creating a list of suitable videos the authors of the paper validated a sample of videos from each of the required language categories to verify their labels. After that the videos were automatically downloaded. Audio was extracted from the videos and some additional filtering and data converting was done. Data was distributed into training, tuning and test datasets to be used for language identification models. In total the dataset contains about 200 hours of audio that is ready to be used.

To benchmark the dataset the authors also proposed multiple language identification models, all following the i-vector approach described in Section 2.5 and using the Kaldi [37] framework. The results of the models were not close to state-of-the-art results, but they still provide a challenging benchmark for the development of spoken language technology. It was also concluded that using language specific data only for tuning the models is not enough and having training data for all target languages is the key for attaining good language identification performance [43].

2.1.3 How2 dataset

Another use for YouTube data has been proposed by the creators of How2 dataset [46]. It is a large-scale dataset that was created for the purpose of multimodal language understanding. Multimodal meaning that the text, speech, images and other means of carrying information are not processed in isolation, but combined together and processed jointly.

The dataset has been put together from instructional videos collected from YouTube, their English subtitles and other metadata, primarily the video description, which they have concluded to be the video summary. Portuguese translations were crowd sourced for the video subtitles so that the dataset would be multilingual. Data has been collected by using an automatic keyword-spider that they have described in more detail in [60]. Total size of the dataset is nearly 80 000 clips and this results in 2000 hours of audio.

Together with the dataset the authors have provided many different applications that the dataset can be used for. For example: automatic speech recognition, machine translation, speech to text translation and content summarization. This shows that such automatically acquired datasets can be applied on a wide variety of tasks with different problem areas.

2.2 Spoken language identification

Spoken language identification is the task of determining the identity of a language from a speech sample [24]. There are many different areas where spoken language identification is used or could be used [24], [48], [5], [31]:

- Automatic speech translation systems (switching between models or systems)
- Multilingual speech recognition (switching between models or systems)
- Spoken document retrieval and categorization (performing sorting and searching on unlabelled audio data)
- Call centers (selecting an operator fluent in the required language)

Nowadays, spoken language identification is also of interest to different intelligence agencies for data analysis and it is starting to be more widely used even in smart vehicles [28]. Since a lot of the systems mentioned work in real time the computational cost for the identification task is often very important [61].

Research has shown that humans are able to identify languages quite successfully even with minimal knowledge of the language and little previous exposure. For unknown languages humans are often able to make subjective guesses, for example "*It sounds like Arabic*" [24], [18]. For machines the task is harder. In more strict terms the problem can be formalized as having a set of acoustic feature vectors $O = \{o_1, o_2, ..., o_T\}$, where o_t is extracted from a waveform at a discrete frame t and there are T such vectors. The set of languages under consideration $\{L_1, L_2, ..., L_N\}$ is equally probable. Identifying a language out of the set of N possible languages involves an assignment of the most likely language label $\hat{L}$ to the acoustic observation O, such that

$$\hat{L} = \underset{l}{\operatorname{argmax}}\ p(O|L_l) \tag{1}$$

which follows a maximum-likelihood criterion [24], [5].

2.3 Methods

Over time different approaches have been used for solving the spoken language identification task, from high level systems focusing on phones and the frequency of the sequences of phones observed in each target language to systems based on spectral characteristics of each language [10]. Many algorithms previously developed for all kinds of speech and speaker recognition tasks can be effectively applied in language identification [5].

Typically the process can be divided into two separate problems: frontend modelling and backend modelling [4], [3]. The frontend modelling part takes care of the feature extraction process, extracting a sequence of features from the input audio data. The purpose of such extraction is to find the most relevant data from the speech waveform and eliminate as much useless data as possible. For language identification a good feature extraction process would filter out all speech properties that are dependant on the speaker or noise and try to assign a higher importance to the speech segments that would be the most useful for differentiating between multiple languages [3]. One of the most used techniques for feature representations are Mel Frequency Cepstral Coefficents (MFCC) which also approximate the nonlinear frequency resolution of the human ear [56], [3]. After the feature extraction process is finished, model training and language identification can be done in the backend step. For that there are also many different methods that have been used over time. In the past GMM-UBM models, Support Vector Machines (SVM) and logistic regression have been very popular [18], [10]. Nowadays, in the state-of-the-art systems neural networks have replaced them [27]. For language identification Deep Neural Network (DNN) the input is a stacked set of spectral features extracted from short segments of speech [42], [7].

2.4 Neural networks

Neural networks are a popular machine learning method that is currently applied in very many different areas, starting from image recognition and computer vision to language technology and medical research [26], [16], [36]. These networks are used as computational models that are able to extract important information out of large datasets and later make predictions based on what they have learned.

As the name “neural” implies, the networks contain small computational units called neurons. In the simplest form a single layer neural network is just a type of neuron called a perceptron, shown in Figure 1. Neurons and perceptrons are lightly inspired by the human brain and act as simple input-output devices. The simplest form of a perceptron will take an input in terms of binary data $x_1, x_2, ..., x_n$ and produce a single binary output. Each input has also a weight $w_1, w_2, ..., w_n$ which describes the importance of that value relative to the output. In the most basic case the single binary output of the network is either 0 or 1, depending on the weighted sum $\sum_j w_j x_j$ and some initially set threshold value. In other types of more complex neurons the output can be any value between 0 and 1 and also the simple threshold is replaced with a value called bias.

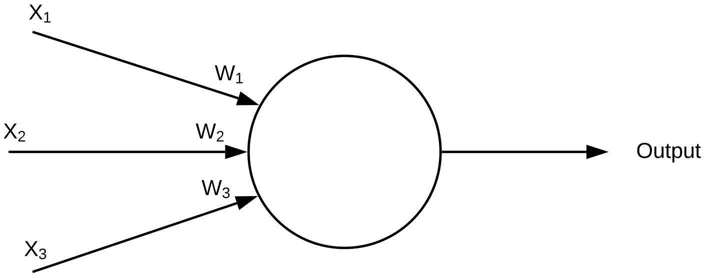


Figure 1. A single perceptron with three inputs.

In larger neural networks, shown in Figure 2, neurons are combined into groups called layers and there is more than just one computational layer. The leftmost layer is called the input and the rightmost the output. The layers in between are referred to as hidden layers, since the computations performed there are not visible to the end user. If a neural network has two or more hidden layers then it is also referred to as a deep neural network. If every neuron in a layer is connected to every neuron in the next layer then the layer is called fully connected or dense.

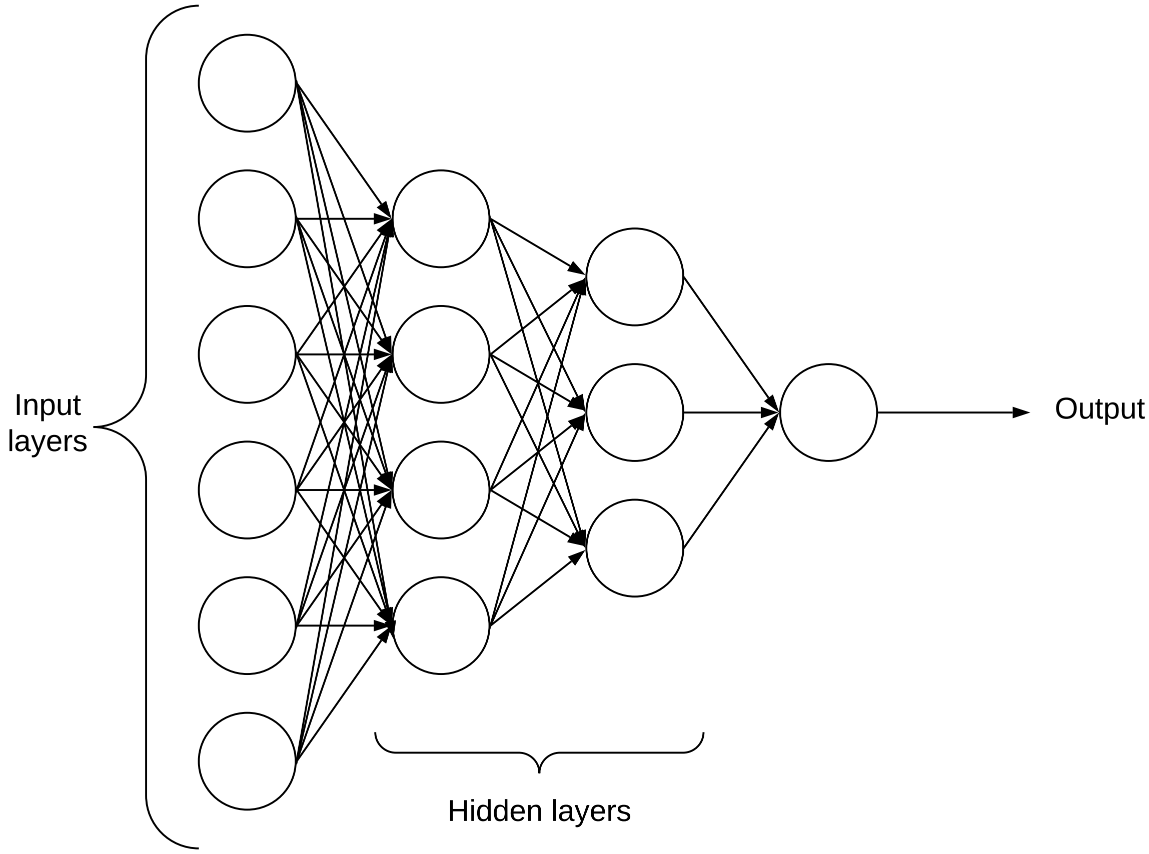


Figure 2. Example of a neural network with four layers.

Overall the process of making neural networks do what is needed is called training them. The training is enabled by the use of weights and biases in the neurons. By feeding input data thorough the network in many steps called epochs we can adjust the weights and biases at the end of each cycle by taking the prediction made by the network and comparing that with a real known value for the particular input. Doing so we can calculate the loss or error in the network and adjust the weights and biases accordingly to try to minimize the loss. Such iterations are done many times in the training stage, until the predefined number of steps is exceeded or some other form of criteria is met. Frequently some kind of an early-stopping criteria is used that will stop the training if for example the loss in the network has not decreased a considerable amount in the last n epochs. The training process itself can be done in multiple ways. In supervised training the data has labels that are known to be valid and the training process relies on the fact that they are correct. In unsupervised training there are no labels or the labels are called weak and the model tries to learn the structure of the data on its own. In this work, supervised learning is used, although the labels associated with the audio data are quite noisy and may not be true.

2.4.1 Loss functions

In general the loss value in the network describes how close the predictions are to the target towards which the training is done [35]. Choosing the right loss function for the application at hand is quite critical, because a good choice for the loss can often increase the networks accuracy noticeably. One of the simplest loss functions that is used is the quadratic loss, also known as Mean Squared Error (MSE), defined in Equation 2

$$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2}$$

where n is the total number of training samples, Y_i is the expected output vector of the network and $\hat{Y}$ is a vector of the network predictions [33], [6], [35]. When the network predictions are equal with the expected values then the loss value decreases to zero and when the Euclidean distance between the network targets and predictions grows then also the loss value increases.

Different types of loss functions are used depending on whether the network has a single binary output or the predictions have multiple possible classes. For binary classification the previously shown MSE loss can be used. For multiclass predictions a softmax final layer is commonly used in the neural network architectures which outputs a probability distribution and requires a different type of loss function [33]. In such cases one of the most commonly used losses is the Cross Entropy (CE) loss

$$L_{CE} = -\sum_{i=1}^{n} y_i \log(o_i) \tag{3}$$

where y is the vector of outputs for the n mutually exclusive classes and o is the vector of output probabilities [1].

There are many more loss combinations and types that can be used based on specific needs, for example some may be better suited for applications in the computer vision field and others can be used quite well in all areas. Some of such loss functions that are well suited for current work in the language domain and that are able to deal with noisy or falsely labelled data will be described in Section 5.3.

2.4.2 Data augmentation

When training neural networks, then usually the intent is for them to generalize as well as possible and to reduce overfitting [6], [1]. To do that as much training data as possible is needed, but in many cases there may just not be enough labelled data available and there is no way to collect more. In these cases (and sometimes even if there is enough data) data augmentation can help. Data augmentation is the process of artificially creating more data. It must be applied carefully, by knowing the limitations of the task at hand to not ruin the results. One area where data augmentation is widely applied is in computer vision and image recognition. An existing image can be flipped, rotated, warped, cropped, scaled and transformed in many different ways to create new representations of the same input image. Usually such transformations do not take much computing power, so they do not need to be pre-generated and can be created during the training process [1].

Related to current thesis, data augmentation can be well applied in the spoken language domain also. A common solution is to add all kinds of noises to the input audio or speech samples, like music, babble and traffic. Audio files can be be speed perturbed, meaning that the audio is either slowed down by some factor (commonly 0.9) or sped up (commonly by a factor of 1.1). Additionally, time can be shifted or the pitch changed. Some of these augmentation methods will be applied in the language identification models described in Chapter 5.

2.5 I-vectors

In the past most systems built for spoken language identification were based on identity vectors (i-vectors) [9]. Additionally they have been used for tasks like speaker and dialect recognition, speaker diarization, speech recognition and many more [22], [10], [56], [59].

I-vectors are a compact form of the speech data, as a fixed size representation of the speech utterance (frequently 400 to 600 dimensions) [61]. The fixed-size representation means that the sequence of frames for a given utterance can be mapped into a low-dimensional vector space while preserving the total variability of the signal [22], [10], [4], [24]. Thus the low-dimensional space is also called total-variability space.

The process of i-vector extraction is done by mapping a sequence of vectors obtained from a speech utterance to a fixed length vector [15]. An audio segment is first analyzed to find all the relevant information and to extract acoustic features that convey the language information [41]. The i-vector mapping is done by using a Universal Background Model (UBM), which is essentially a language independent Gaussian Mixture Model (GMM). The Baum-Welch statistics from the utterances are collected and a supervector is constructed by appending together the statistics for each mixture component as in Equation 4

$$M = m + Tw \tag{4}$$

where m is the speaker- and channel-independent supervector, T is a rectangular matrix of low rank and w is a low-dimensional random vector having a standard normal distribution $N(0, I)$ [9], [24], [61], [56]. An i-vector is obtained for each utterance as the Maximum Posterior Probability (MAP) point estimate of w [15].

2.6 X-vectors

X-vectors are the newer approach for solving problems in the language domain. They have been indirectly introduced and used in [51] without being called x-vectors yet. The method has been used for speaker recognition and diarization tasks with excellent results [52], [54], [12]. Recently, it has been adapted for the language identification task and has outperformed many existing systems in that area, being currently the state-of-the-art approach for the task [53], [38].

The x-vector system is built with a feed-forward DNN and the input to the network is a sequence of T speech frames (raw audio in the form of MFCC-s) [53]. The network can be divided into four major components. The first part processes the data at frame level and is essentially a Time Delay Neural Network (TDNN) [39]. The second component is the statistics pooling part, where the mean and standard deviation of all the vectors is calculated and concatenated. Then the fully connected layers follow and finally a softmax classifier layer. The x-vectors themselves can be extracted after the first fully connected layer, that is the segment 6 in Table 1. Although for short utterance evaluations it has been reported that better results can be achieved by doing the extraction one layer later, after the seventh segment [21].

One of the good features of x-vectors is that after extracting they can be used like i-vectors. That means that all of the technology that has been developed for i-vectors over many years can be utilized. The x-vector DNN can also be used directly for classification tasks without the need to extract the vectors.

Table 1. Standard x-vector DNN architecture [53].

Layer	**Layer context**	**Total context**	**In × out**
frame 1	$\{t-2, t+2\}$	5	$5F \times 512$
frame 2	$\{t-2, t, t+2\}$	9	1536×512
frame 3	$\{t-3, t, t+3\}$	15	1536×512
frame 4	$\{t\}$	15	512×512
frame 5	$\{t\}$	15	512×1500
stat. pooling	$[0, T)$	T	$1500T \times 3000$
segment 6	$\{0\}$	T	3000×512
segment 7	$\{0\}$	T	512×512
softmax	$\{0\}$	T	$512 \times L$

The authors of x-vectors have reported that in order to receive the best results with language identification, data augmentation should be used. In their work a random augmentation method from the following options was used to augment the speech utterance at hand [53]:

- speed perturbation
- music - a random sample from the Music Speech And Noise corpus (MUSAN) [50] is added to the input
- noise (MUSAN noises)
- reverberation - recording is reverberated with simulated Room Impulse Responses (RIR) [23]

The spoken language identification models in this work also use the x-vector approach, but the implementation is based on different tools and has some additional components.

3 Data collection

This section covers the process and tools that were used for collecting the speech data that is required to build the language identification models. Additionally, the used data sources and data acquisition methods are described and an overview of the collected datasets is given.

3.1 Data requirements

In this work the methods used for the task of spoken language identification are neural networks. In order to create the models and train the networks much data in audio form is needed. In the case of language identification the data needs to be speech data, as the goal is to recognize the language from audio clips that contain speech. As with many neural networks training tasks the datasets need to be annotated. This means that for each audio clip in the dataset there needs to be a corresponding label that refers to the spoken language in that clip. To automatically collect such annotated data, two different data sources are needed in this work. One for multilingual text data and the second one for audio data.

3.1.1 Text data

Before starting any major work on the data collection process, some requirements were set for both types of data sources that will be used. The text data is the first step in the speech data collection process. It should be available for as many different languages as possible, because the final amount of speech data collected depends on how much text data can be used in the first step. The text dataset for each language should not be very small in size and the content should cover a wide variety of categories. This greatly helps to improve the results in the audio collection step and ultimately increases the language identification models' accuracy.

3.1.2 Audio and video data

For the second audio data collection step six initial requirements were also set that would give a general goal to aim for.

- Initial subset of 8 languages
- Long term goal for 100+ languages
- For each language 100–150 hours of audio
- Maximum audio clip duration is 1 hour
- Data from different categories, speakers and recording conditions
- Metadata for each audio clip

Before doing any experiments with the data collection process and not knowing the limits of the used methods and the quality of the collected data it was not reasonable to aim for a very large number of languages at first. Initially a goal of eight languages was set that is shown in Table 2. The collected data quality for these languages could be quickly and easily evaluated by a few people and if the results were satisfactory then the larger target of 100+ languages would be tackled next.

Table 2. Languages in the initial target set of eight languages.

Languages	English, Estonian, Russian, Urdu, Latvian, German, Finnish, Spanish

For the amount of data collected for each language it was decided that 100–150 hours of speech should be enough for most cases. Doing some experiments showed that the data post-processing steps may filter out a quite large portion of the collected data, so the 100–150 hour limit was chosen with a reasonable buffer. This ensures that even after all of the filtering steps, there is still enough data remaining for the training of the neural networks. In reality, for some languages it was not possible to collect that much data, but this is still acceptable since even less than 50 hours of data can be used for creating the models.

After doing some experiments it was concluded that the maximum duration of an audio clip should be limited. It was decided that one hour is a good value that works well for the majority of languages. The reason for such limits is that the automatic data collection process used for finding the audio data, often comes across videos that are much longer. For example, when dealing with data that was in Estonian the Parliament meetings were often encountered and their durations are frequently many hours long. If many of such videos would be added to a language's dataset then the entire dataset would be made of only a few very long videos which is not desired.

The requirement for videos to be from many different categories, speakers and recording conditions is also important. It helps the trained language identification models recognize the language from different types of recordings, speakers with varying ages, speech with specific domain content etc.

3.2 Available data sources

Nowadays, there are many different platforms available that freely distribute all kinds of open-source media and audio-visual information. In this work, for the purpose of language identification, audio data extracted from videos works as well as any other source. One of the most well known and popular video sharing platforms around the world is YouTube. To take advantage of that popularity and the vast amount of data that can be collected from there, YouTube is an excellent choice to use for the speech data.

YouTube being a large platform with billions of videos, there needs to be an effective way to find the required content. Of course the main part of finding the required data is the YouTube's search engine, but for that some form of keywords, search phrases or query strings are needed for all of the target languages. Such search phrases can be generated from text data by using different text mining and analysis techniques, provided that there is a large enough text corpus for each of the target languages. Since the text datasets have to be multilingual and also contain content about different categories, a perfect data source that fits such requirements is the free encyclopedia – Wikipedia. It does contain articles about a very wide range of topics and also has multilingual versions in more than 309 languages [57]. Figure 3 shows the dataset sizes for the 10 largest Wikipedias. It can be seen that the largest Wikipedias contain millions of articles, which is more than enough for this work. By collecting data from Wikipedia for many different languages a large multilingual dataset can be put together that provides us the data needed for generating the search phrases for each of the target languages. The phrases can then be used to find relevant content from YouTube.

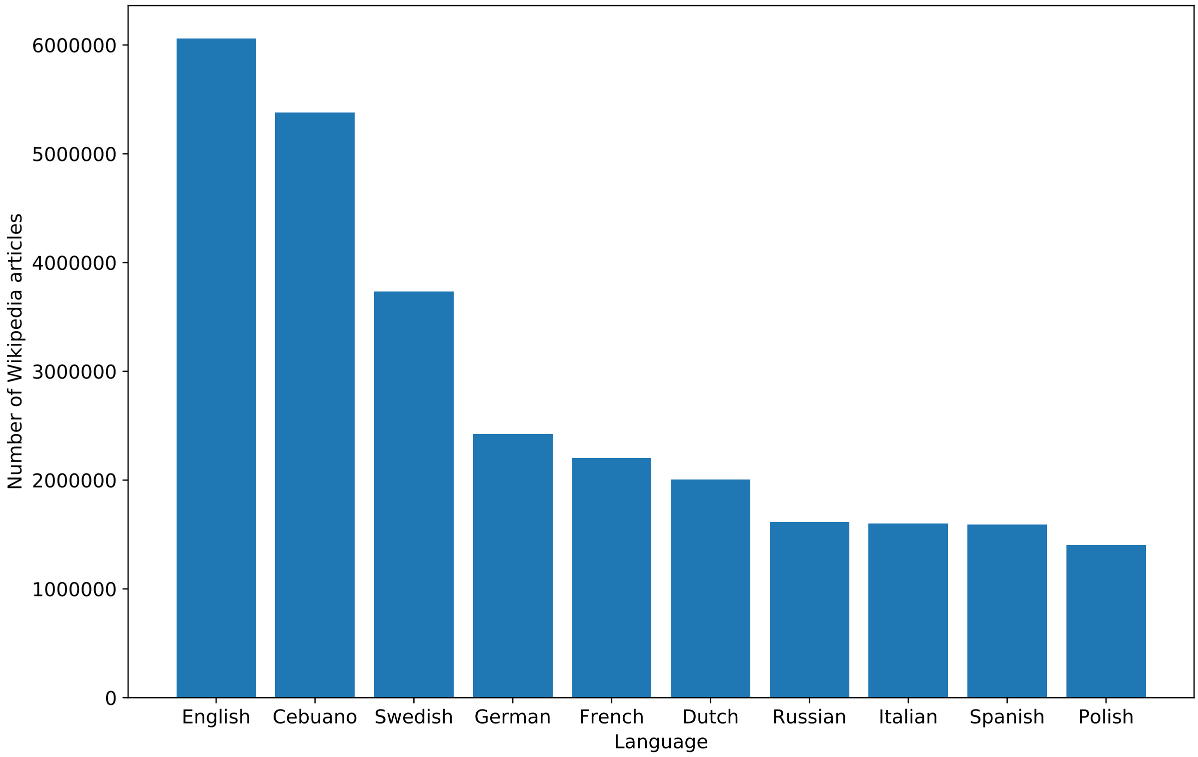


Figure 3. Top 10 languages in Wikipedia with the most articles.

As an alternative to Wikipedia, another interesting multilingual dataset that has been recently released and could also potentially be used is OSCAR [55]. It is a large collection of multilingual text corpora that have been assembled by using the CommonCrawl[2] dataset, which contains automatically collected web-crawl data. This data has been filtered and clustered by the authors of OSCAR and is now distributed as a large collection of datasets for many different languages.

In this thesis the OSCAR dataset was experimented with initially, but since the Wikipedia

[2] https://commoncrawl.org/

resources have currently been enough and in some ways easier to work with, the potential of OSCAR has not been utilized. One of the reasons why OSCAR is a bit harder to use in this thesis is the fact that the data is not categorized in terms of content in any way. There are pieces of news articles, advertisements, random website texts, etc which makes the search phrase generation results not as good as when using data from Wikipedia. This could potentially be solved by using different methods for generating the phrases or by trying to automatically categorize the data. In terms of the dataset size OSCAR in many cases has considerably more data than Wikipedia. For example, Figure 4 shows the size differences between the 10 largest Wikipedia's and OSCAR. In the case of English OSCAR has more than 100 times more (1.2 TB deduplicated) data than Wikipedia. On the other hand for some languages required in this work OSCAR did not have any available datasets at all.

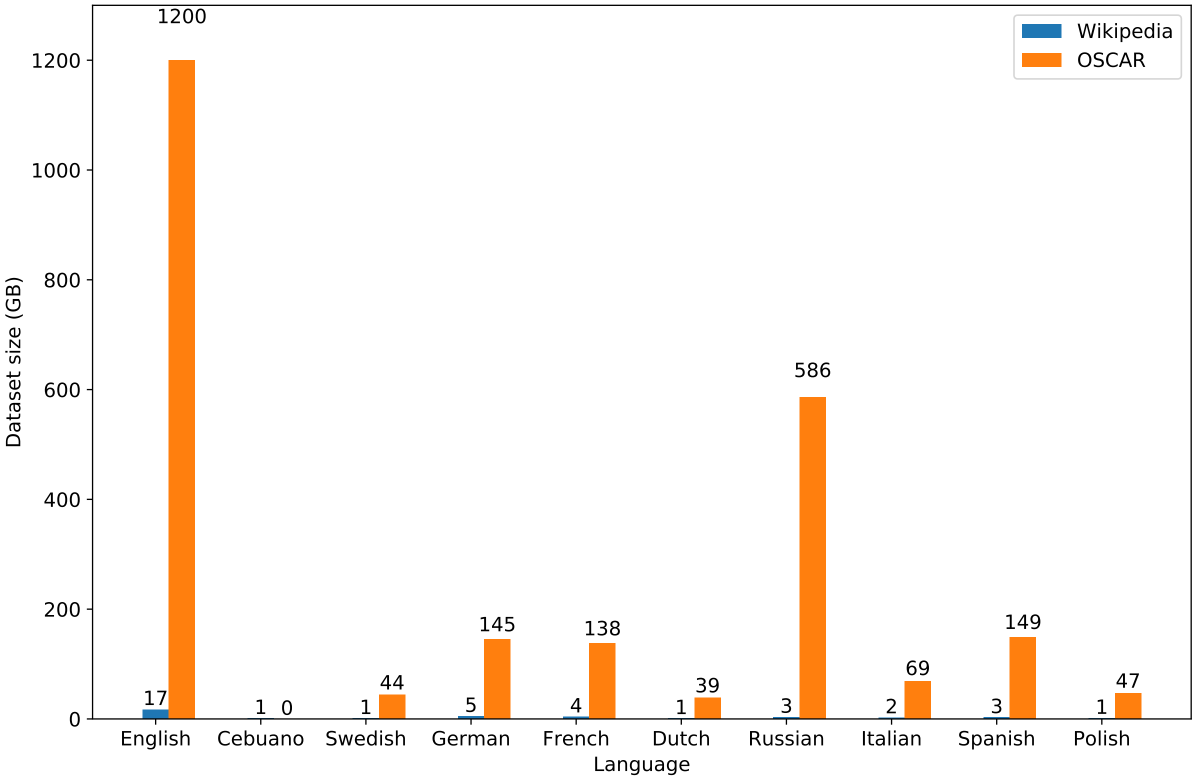


Figure 4. Dataset size difference between Wikipedia and OSCAR.

3.3 Collection process

Overall the data collection process can be divided into multiple steps. A diagram describing the process and steps is shown in Figure 5. First, the Wikipedia data needs to be downloaded, then the search phrases can be generated, videos collected and finally downloaded. The following sections describe the process in more detail.

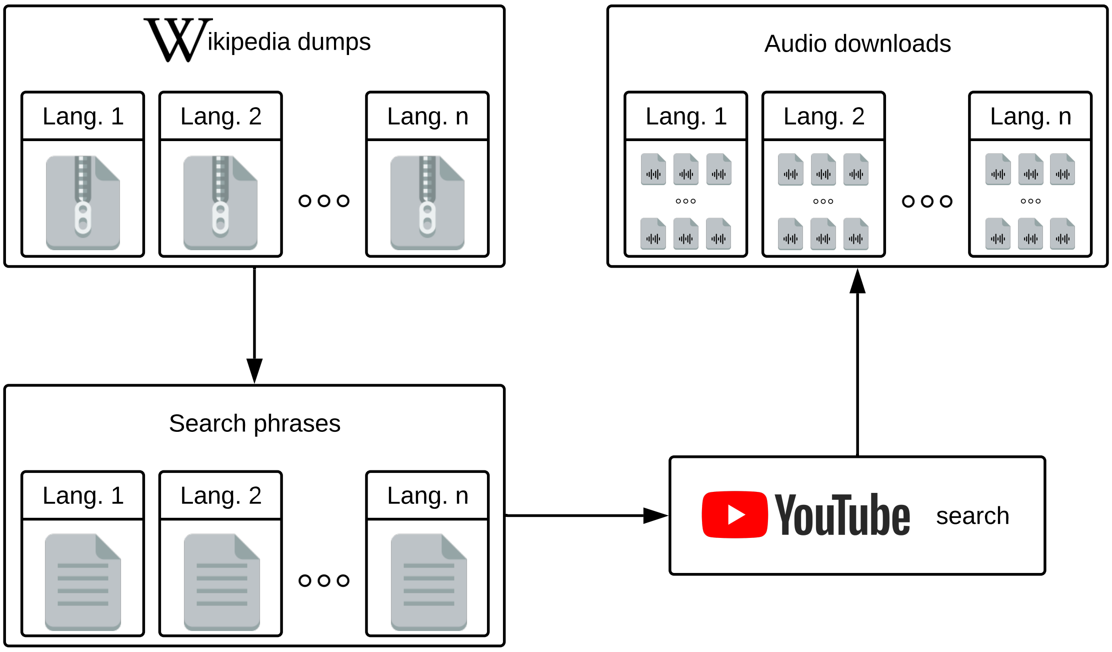


Figure 5. High level overview of the data collection process.

3.3.1 Wikipedia data

To collect the required audio data from the YouTube platform search phrases for querying the videos have to be generated. For the phrases Wikipedia data is used as described before. Luckily, Wikipedia contents are quite well distributed and readily available, there is no need to carry out a separate web crawling process to collect data from Wikipedia pages. The encyclopedia's data is distributed in terms of dumps[3] that are created frequently (roughly once a week) by dumping the entire Wikipedia contents into compressed eXtensible Markup Language (XML) format. The dumps provide different types of data

[3] https://dumps.wikimedia.org/

to choose from: database backups, static HTML pages, analytics data, etc. For generating the search phrases database backup dumps are the most useful, they contain Wikipedia pages and articles.

The amount of data and articles count for each language's Wikipedia is quite different. For some smaller or less active communities there may be less than 1000 articles. On the other hand larger languages have hundreds of thousands of pages and articles. For the initial set of eight languages that were used in this work, the article counts are shown in Table 3. After doing some experiments with the eight languages and some additional less active or smaller languages it was decided that only languages that have more than 10 000 articles can be used the most effectively in this thesis. That is not a problem, since out of the 307+ languages available in Wikipedia 149 of them have more than 10 000 articles. In some cases the limit of 10 000 may not be enough because in some languages a huge amount of articles are automatically generated or translated and the contents are about all kinds of locations or geographical information which is not suitable.

Table 3. Number of Wikipedia articles for the initial eight target languages [57].

Language	**Number of articles**
English	6 054 821
German	2 418 874
Russian	1 612 983
Spanish	1 589 671
Finnish	482 388
Estonian	207 555
Urdu	152 736
Latvian	101 007

Data from Wikipedia was collected by creating a script that would automatically download the correct Wikipedia dumps for a predefined list of languages. Then the collected dumps would be uncompressed and converted into more easily readable and processable JavaScript Object Notation (JSON) format. This resulted in folders of Wikipedia's JSON data for each of the target languages. A segment from a random article in the dumps is shown in Figure 6. To make processing the data easier all of the JSON files were concatenated into one long file per language. Articles shorter than 3000 characters and the ones containing just a title were filtered out. This improved the results in the next phrase generation step.

```
{
    "id": "291260",
    "url": "https://en.wikipedia.org/wiki?curid=291260",
    "title": "Underwater photography",
    "text": "Underwater photography\n\nUnderwater photography is
             the process of taking photographs while under water.

             ...

             allowing for tens of new records and even new species.
             \n\n\n\n"
}
```

Figure 6. Segment of a random article from English Wikipedia dump.

3.3.2 Search phrases

After the Wikipedia datasets for each language had been downloaded and ready to use, the search phrases could be generated. The idea behind the phrases is that if the dataset from English Wikipedia is taken and some text mining or analysis methods applied then we can generate some words or phrases that are also in English. If the source dataset is in another language then the generated phrases will also be in that other language. If some language's phrases are used to query the content from YouTube, then usually there is a quite high probability that the video results (or at least the top results) are in that same language that the search phrase was. That is just how some part of the YouTube search engine works and how it can be utilized in this work. Such automatic data collection process will undeniably not be 100% accurate and there will be false positive video results that have to be dealt with.

To generate the search phrases there are quite many approaches that could be taken. In this work the quality of the search phrases is not too critical, therefore there is no need to use very complex or advanced methods. The method that will be used is Term Frequency Inverse Document Frequency (TF-IDF). TF-IDF is a well known method that helps to evaluate how important a word in a document is. If a word occurs many times in a document, then its relevance should be increased and it should have more value than other words in the same document. At the same time if a word occurs many times in many documents, then it might just be a frequent word that should not have a high relevance. We can apply TF-IDF on each language's Wikipedia dataset and we will get a long list of the "most important" phrases for each of the languages.

After doing some initial experiments with the collected datasets in different languages, it was concluded that a good and universal phrase length to use in the YouTube search engine is three words. Using shorter phrases resulted in quite vague results that also had very many false positives. Longer phrases on the other hand decreased the amount of results too much or there were no results at all.

Generating the phrases or trigrams for the initial set of eight languages showed that there is still more false positive results than is desired. In this case the false positive results are the search phrases that are in other languages than expected or contain too many numbers, stop-words, etc. To alleviate the main problem of the phrases being in the wrong language an extra step was introduced to the phrase generation process, in terms of a text based language identification model. A few available tools were experimented with and finally the Polyglot[4] Python package was chosen, which supports a wide variety of natural language processing tasks with multilingual data and also had the best balance between supported languages and processing speed.

The text based language identification model could be applied on the generated phrases and all phrases whose language was unknown or not the one that was expected were filtered out. This step removed quite many generated phrases and now the final set of phrases that was achieved was well usable. A randomly picked sample of the generated phrases for the initial eight test languages can be seen in Table 4.

[4] https://polyglot.readthedocs.io

Table 4. Sample of the generated search phrases for the initial eight languages.

Language	**Random search phrase**
English	“the northern territory”
Estonian	“ameerika ühendriikide relvajõud”
Finnish	“hiileen liittynyt hydroksyyliryhmä”
German	“abgesetzten enameloliden zahnkappen”
Latvian	“starptautiskajā šaha turnīrā”
Russian	“совета рабочих депутатов”
Spanish	“administración del estado”
Urdu	“انہ اور فرانس”

3.3.3 Searching for videos

When search phrases for the required languages were collected and filtered, the next step would be collecting the videos from which audio could be extracted. An overview of the process is given in Figure 7. The first task is to use the generated phrases and find videos that match the requirements set before, without downloading them yet. A very convenient way to access YouTube search functionality is through the YouTube Application Programming Inteface (API)[5]. The problem with that approach was that the YouTube API limits were exceeded too quickly. If the video collection process would have been distributed over a longer period of many months then it would not have been a problem, but it was desired to perform the task relatively quickly. Another option to search for videos is the YouTube web interface that is exposed to all users. To use that, another script was created that would go through the generated search phrases and try to find videos through the search functionality on main YouTube site. This approach raised another problem, now the request rate limits were exceeded too quickly, meaning that too many requests were made in a short period of time. A simple solution to the problem was to add a delay between each search query, but it turned out to not be a permanent fix. Some other approaches were experimented with, like using proxy servers, adding

[5] https://developers.google.com/youtube/v3

HyperText Transfer Protocol (HTTP) headers and pre-validated cookies, also Virtual Private Network (VPN) servers were tried. Final approach, that seemed to work in all occasions was to use a single non-public proxy server. This allowed to continuously search for videos without any delay and no restrictions. In the end it seemed like there could have been some restrictions on the particular machine or network that the requests were initially made from, because when doing some experiments with random Virtual Private Servers (VPS) with no proxies there were no problems also.

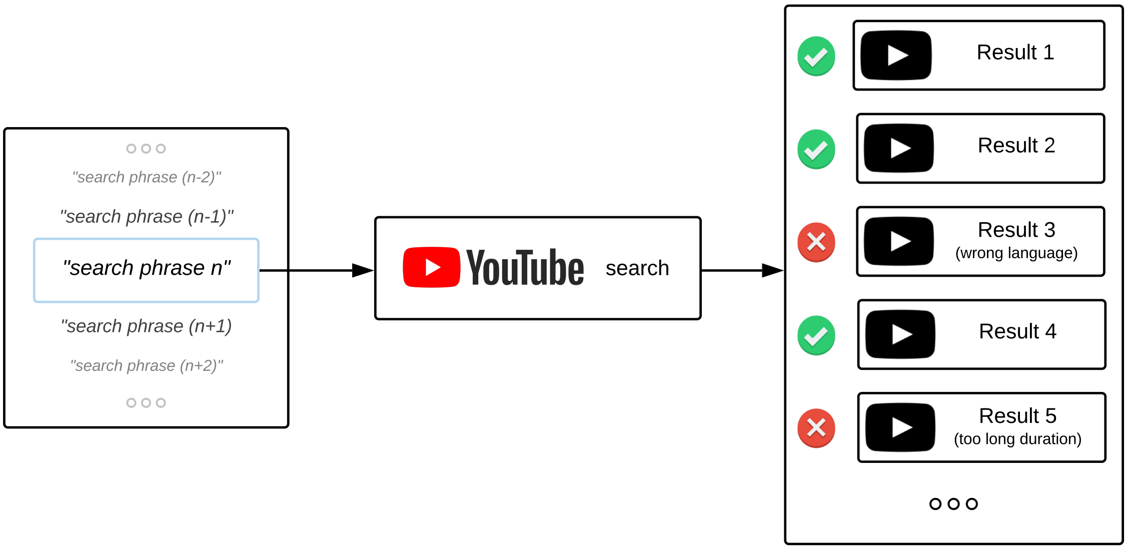


Figure 7. Overview of the process of searching for videos.

Overcoming the obstacles on the technical side enabled to start the real video collection process. All of the previously generated phrases were used one by one to find matching videos. Even though the phrases were heavily processed and filtered this still resulted in many false positive video results. In this case the search phrase was in the expected language but the videos were in another. In rare cases even the video title would be in the expected language but the content would not. To decrease the number of false positive video results it was decided that the same text based language identification model could be re-applied in this step. Now it would be applied on each of the video results. The model could be applied on the title, description (which usually reflects the video content and language most closely) and other metadata if available. After doing a search with a phrase the top n results would be fed through the language identification model and all results with unknown or not expected language were filtered out, similarly to the search phrase generation process before. This decreased the amount of false positive results noticeably. Overall the process of searching for videos was the most time consuming. For the initial target set of eight languages, collecting the required amount of video id's took a few days. In later stages, when a dataset of 107 languages was collected then the process took many weeks of continuous searching.

3.3.4 Downloading audio

In the previous step videos were not yet downloaded. The id's of suitable videos were stored to be used in this step. This allowed to separate the time-wise long task of searching for videos and also to run the two tasks simultaneously if needed. Now valid video id's were processed and videos downloaded, audio extracted and saved into datasets. In addition to the extracted audio also video metadata was downloaded and stored. In the downloading process the youtube-dl[6] tool was used.

3.4 Collected data

As described before an initial set of eight languages was first used to experiment with the process. An overview of data collected for the eight initial target languages is shown in Table 5.

[6] https://github.com/ytdl-org/youtube-dl

Table 5. Overview of the dataset that contains the initial eight languages.

Language	Number of files	Duration (h)	Size (GB)
English	759	180.5	19.4
Estonian	561	150.3	16.1
Finnish	573	122.0	13.1
German	516	136.0	14.6
Latvian	686	100.0	16.2
Russian	581	181.0	19.4
Spanish	479	131.0	14.0
Urdu	709	138.5	14.9

In later stages data was collected in total for 107 languages. The full list of languages in the final dataset and the data properties are shown in the table in Appendix 2. In total audio was collected from close to 78 thousand videos, resulting in a dataset with a size of over 1 terabyte and 14 044 hours of audio content.

When looking at the video categories in the large dataset of 107 languages in Figure 8. It can be seen that there is data from many different topics. The category with the highest number of videos ("People & Blogs") could actually have subcategories of it's own, because the contents of such videos have a wide range of topics.

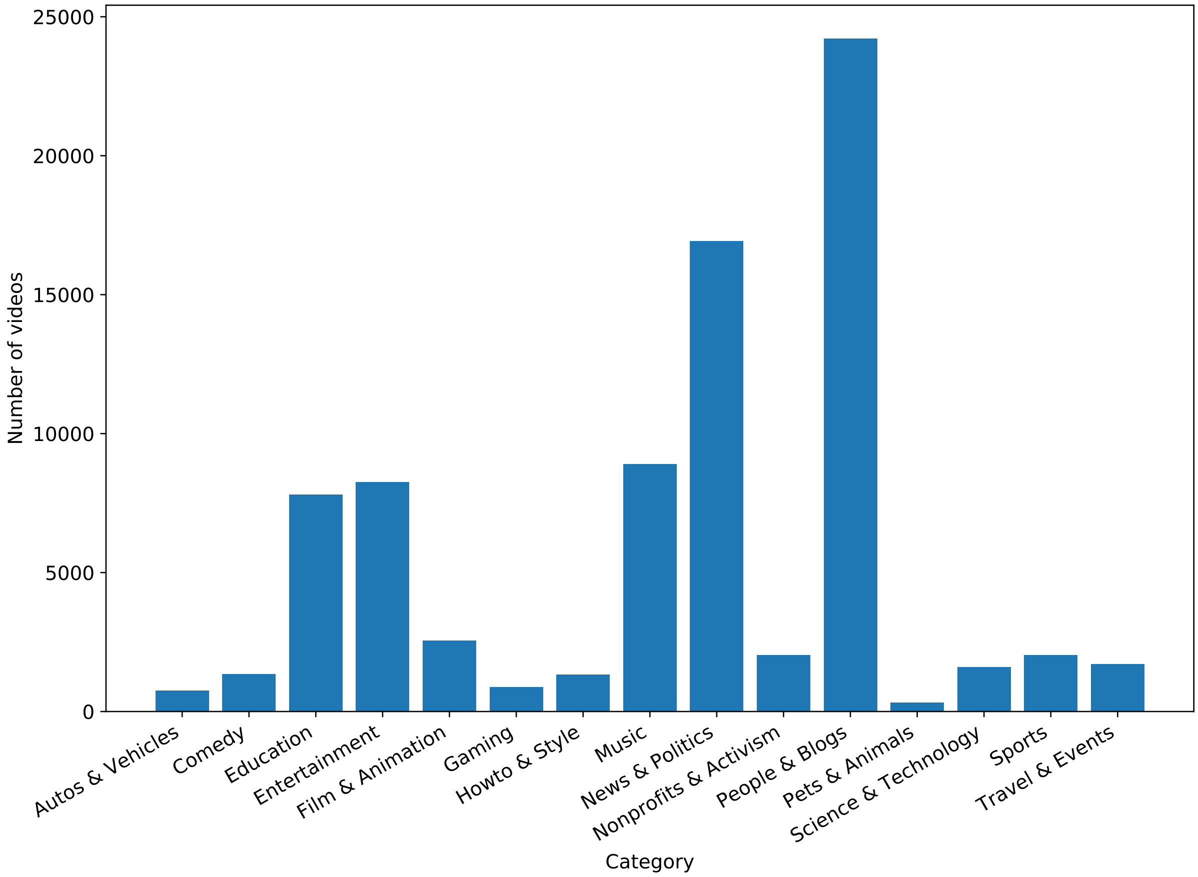


Figure 8. Video category distribution for the large collected dataset of 107 languages.

All of the collected videos have a maximum duration of one hour. When looking at all the durations in Figure 9 it can be seen that the majority of the dataset consist of relatively short clips between 1–10 minutes. This is actually good, because then there are many clips with different speakers, acoustic conditions, topics, etc, as the requirements asked for.

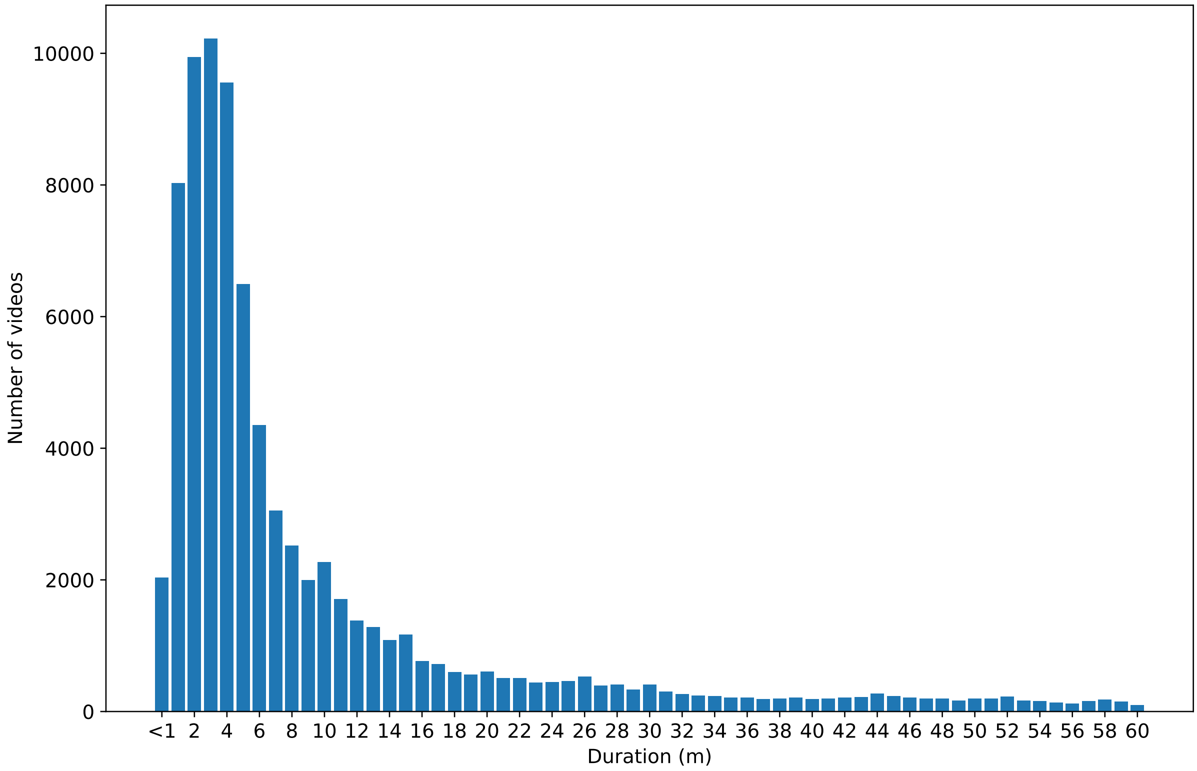


Figure 9. Video durations in the large collected dataset of 107 languages.

3.5 Audio processing

After downloading the audio datasets, two main steps were remaining. Firstly, the downloaded audio files need to be processed. Secondly, a validation experiment needs to be carried out. Audio extracted from YouTube comes in various different formats like *opus*, *m4a*, *ogg*. It is much simpler to deal with the data if every audio file is in same format with similar properties. To accomplish that, all of the audio files were converted into single channel 16k bit rate WAV files which can also be directly used by the language identification models in next steps.

All of the filtering that was done during data collection process removed most of the audio files with inappropriate content, but still some audios remained that contain only music, non-speech content or just silence and that should not be in the dataset. Also audio files with a maximum duration of one hour are quite long to use in the validation and model building step and should be segmented into shorter pieces. To solve both of these problems, speaker diarization process was applied on the dataset by using the LIUM

SpkDiarization tool [45]. The process partitions all of the audio files into shorter segments with a maximum duration of 20 seconds, grouped by the speaker identity. As much of the music and non-speech or silent parts as possible will be removed. Figure 10 shows the distribution of segmented audio clips in terms of their length in seconds. There are a few short segments with a duration of 2-3 seconds and the majority of the clips have a duration between 4 to 10 seconds.

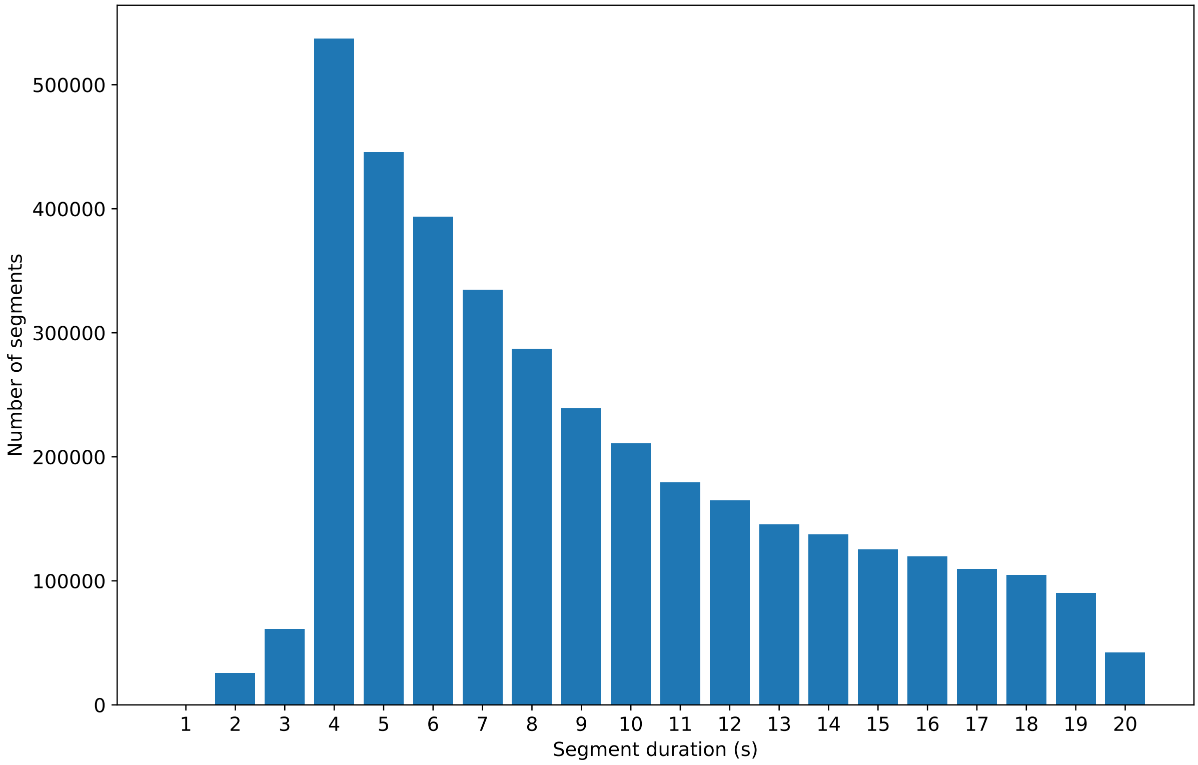


Figure 10. Audio clip lengths after speaker diarization and segmentation.

4 Data validation

As seen from the previous sections, data collected by using an almost automatic pipeline for many different languages inevitably contains false positive results. Even if many filtering steps have been applied during the process. To use such collected datasets it is important to first validate them. Validation means that a small subset of collected audio clips is processed by a human and a true label indicating the spoken language in the clip is attached to each of them. Validation results can then be used for multiple purposes. First, we can get an overview of "label quality" in the collected data. Meaning that if the expected labels from the automatic data collection process match the true labels given by real people then dataset labels are accurate and there are no false positives or true negatives. Secondly, by using validated audio clips, a separate dataset can be built that contains only validated audio clips and can be used for development and validation when training the neural networks. The final language identification models can also be evaluated on such human annotated datasets.

Validation part was carried out in two stages. After collecting data for the initial eight target languages they were validated by a small group of people. Later, after data for a larger collection of 107 languages had been acquired, they were also added to the validation process and more people were offered the opportunity to participate in the validation process.

4.1 Validation application

To conduct the data validation experiment a custom web application was built that would allow to share the validation task to a wide group of people who could have skills in different languages. The application was built with Python and Flask[7] framework as the backend. On frontend Vue.js[8] was used. Since the application logic is not very complex and there is not too much data that needs to be stored, all of the validation results and other information was stored in a SQLite database.

[7] https://flask.palletsprojects.com/

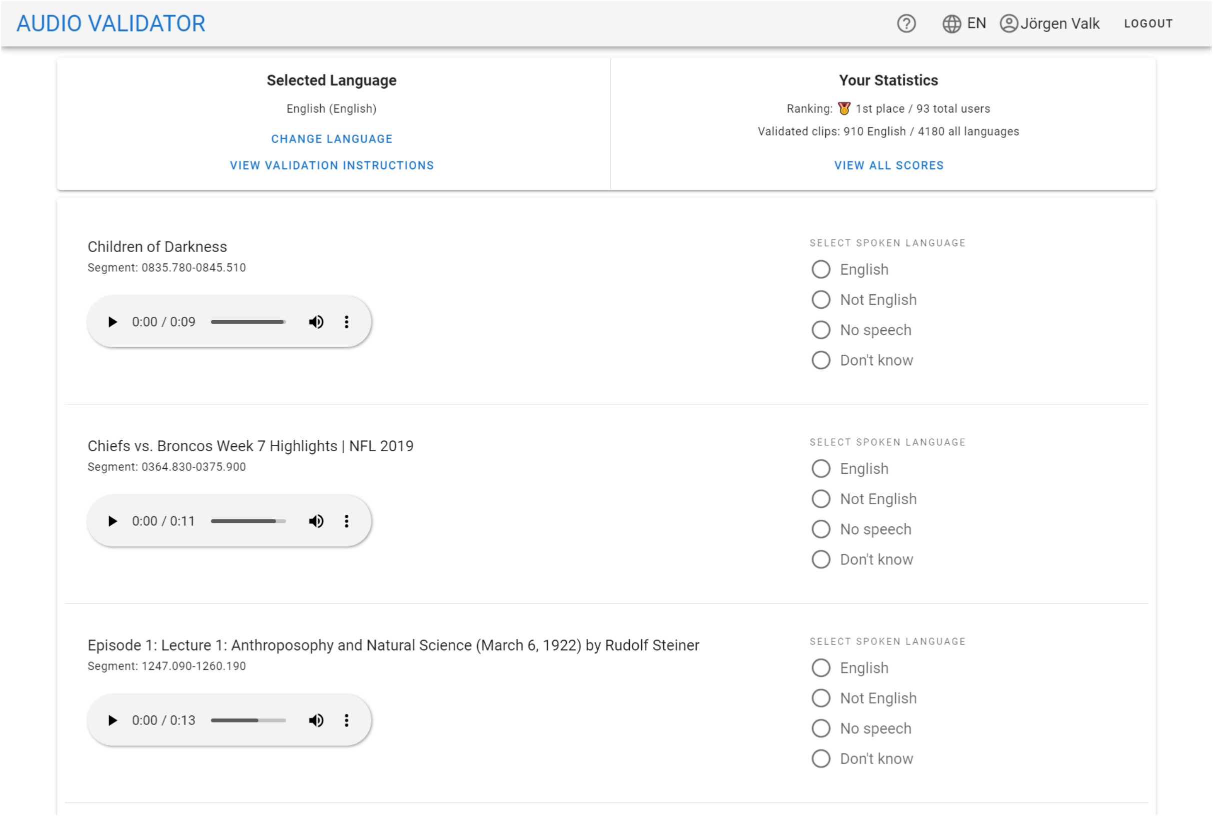
[8] https://vuejs.org/

People who were willing to contribute some time to the validation process could sign in through their Google accounts. Authentication was added to reduce the possibility of luring in spammers and other misbehaving persons. A list of all collected languages was displayed to the users and they could pick a language to validate. Before starting work on a language for the first time users' language proficiency on a scale of 1-5 was also asked. One representing users with no skill at all in the language and five for native speakers. This data could be used later to filter or group the validated audio clips if needed. After selecting a language to validate a random selection of 10 audio clips was given to the user to work on as shown in Figure 11. If possible, then some of the 10 clips were selected from ones that had already been annotated once by another user. This would also allow to filter out some unexpected results later.

The expected process was for the user to listen to an audio clip with a maximum length of 20 seconds and then choose one of the following labels:

- Is `<given language>`
- Is not `<given language>`
- No speech
- Do not know

The `<given language>` part in the first two answers would be replaced by the expected language name. The "*No speech*" option was needed because even after the speaker diarization process and other filtering steps there still exists data that does not contain speech. In some clips it is really hard to differentiate the spoken language or in some cases there are multiple languages spoken simultaneously. That is why the final "*Do not know*" option was also needed.



Selected Language

English (English)

Your Statistics

Ranking: 1st place / 93 total users

Validated clips: 910 English / 4180 all languages

Children of Darkness

Segment: 0835.780-0845.510

Chiefs vs. Broncos Week 7 Highlights | NFL 2019

Segment: 0364.830-0375.900

Episode 1: Lecture 1: Anthroposophy and Natural Science (March 6, 1922) by Rudolf Steiner

Segment: 1247.090-1260.190

Figure 11. Screenshot of the main validating page in the audio validating application.

4.2 Validation results

During validation process people validated audio clips in 50 languages out of the total 107 available. 14 178 unique audio segments were processed which results in 18 560 labels for the entire dataset. All 50 validated languages and their general validation information is shown in a table in Appendix 3. For some languages only a few audio clips were validated and that is currently not enough to be effectively used. Out of the 50 validated languages for 31 more than 50 labels were collected (50 to 3810). These 31 languages are shown in Figure 12. In terms of validation result counts top five most validated languages were Estonian, Armenian, English, German and Finnish. They make up about 50% of the validated segments.

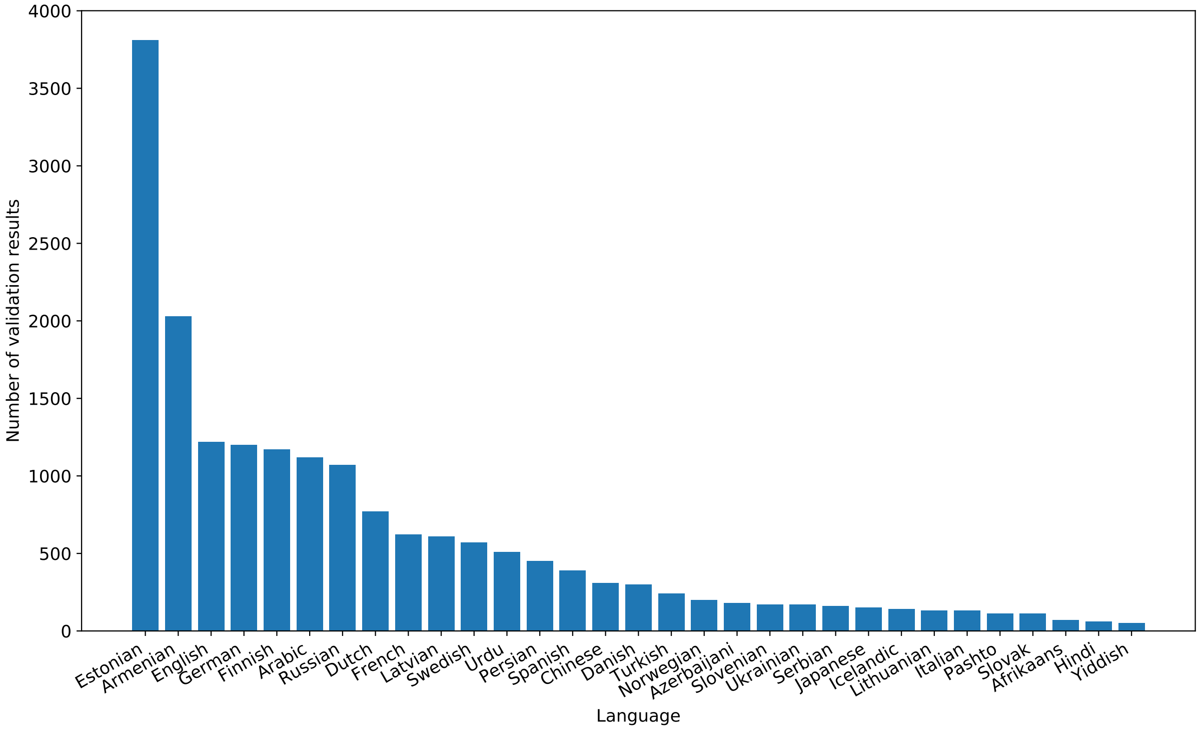


Figure 12. Number of validation results for languages with more than 50 results.

Validation process started initially in November 2019 and the final results in this work reflect the state of April 2020. As can be seen from Figure 13 most of the validation results were collected in the first days in November and secondly when the validation application was distributed more widely in February 2020.

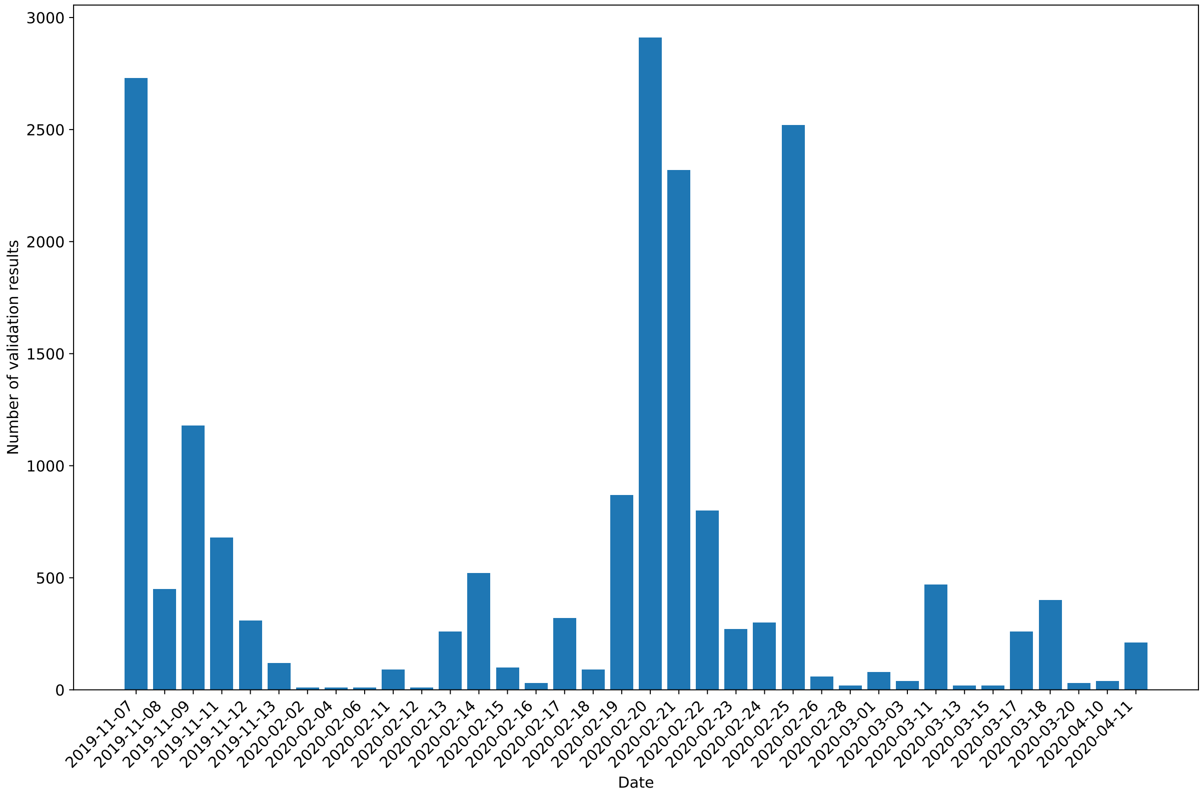


Figure 13. Number of validation results per date.

As validating users were asked for their proficiency in the language that they were annotating it is good to see on Figure 14 that actually the highest percentage of users validating the clips speak the language at hand naively.

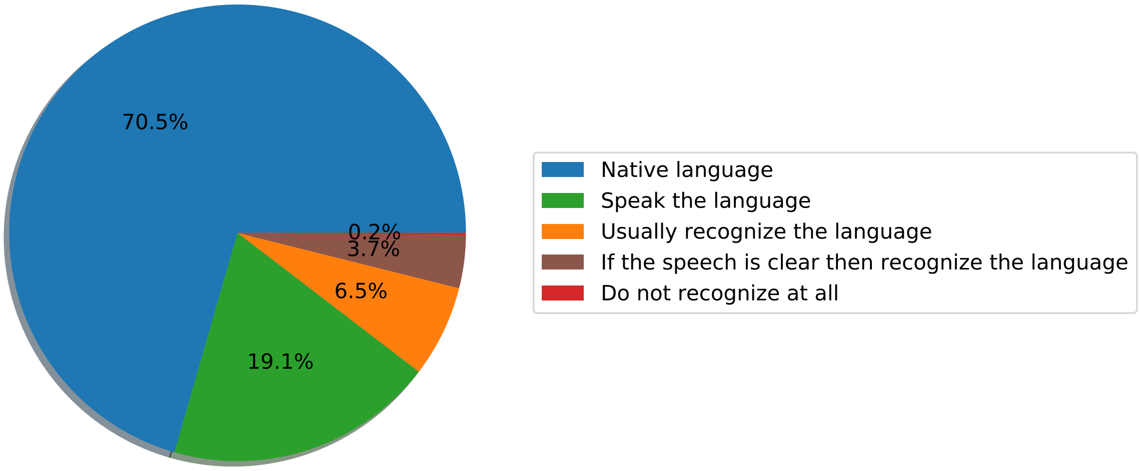


Figure 14. Validation results grouped by the validator’s language proficiency.

One of the most valuable and interesting results from the validation process is how many of the collected videos from YouTube with their expected label actually have the correct label. For all 50 validated languages the results on label quality are shown in Appendix 3. For the 31 languages with a considerable amount of data the results are shown in Figure 15.

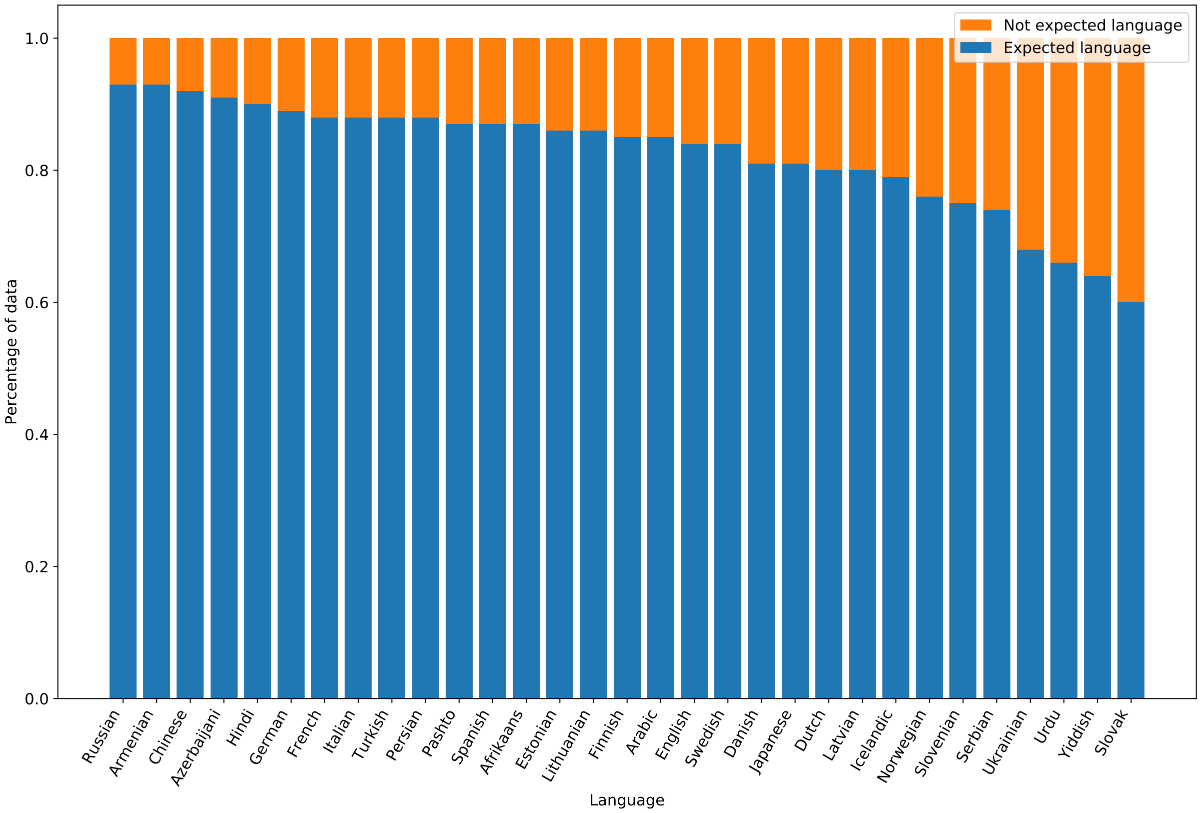


Figure 15. Validation results distribution in terms of expected and not expected language label.

In the previous figure all of the answers given by the annotators that were not “Expected language” were considered as the orange part of the bar. In reality a lot of the “Not expected language” data is made up by segments that were labeled to not contain speech as seen in Figure 16. This could possibly be improved on by applying more strict preprocessing. If not considering the no speech part as false then the expected label percentages are much higher.

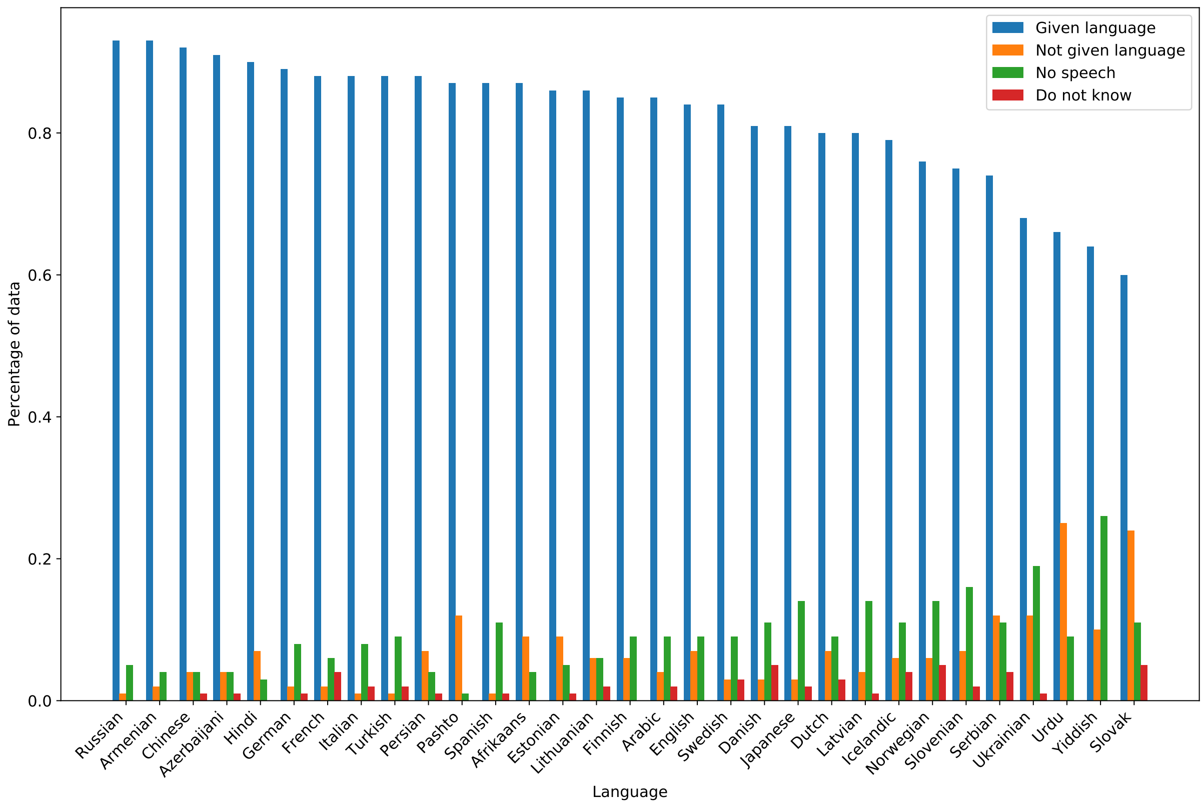


Figure 16. Validation results grouped by answers and languages.

For all languages, on average the percentage of data with the expected label is 85.3%. If not counting the no speech portion which could be improved on then the given language percentage is over 92% as in Figure 17.

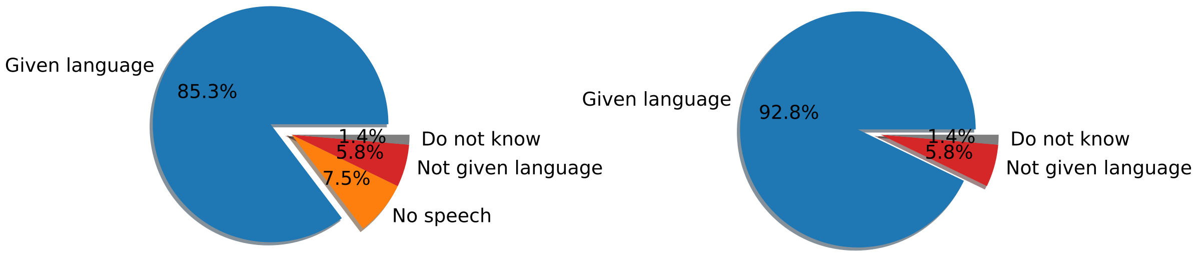


Figure 17. Validation results grouped by answers. In the left pie plot all four possible answers are shown with their percentages. In the right figure the “no speech” answer has been ignored, since it can be improved on.

Overall the validation process was successful. Thousands of labels were given to the audio clips, which can be used for filtering the data and building the models. As a future improvement to get more validation results, the validation tasks could be distributed to platforms like Amazon Mechanical Turk[9] or Figure-Eight (Appen)[10]. This would require some financial investments, but the validated dataset sizes could be considerably increased.

[9] https://www.mturk.com/

[10] https://appen.com/

5 Language identification models

This section describes the spoken language identification models, built on the YouTube datasets and gives an overview of the tools and frameworks that were used in the process.

5.1 Tools

The tools used for building the language identification models are based on Python programming language. The main component used for neural networks is the popular machine learning framework Pytorch [34]. Around that a wrapper called Pytorch-Lightning [13] has been used that simplifies and speeds up the process of building the models by abstracting away parts of code that need to always be present in the training cycle and decoupling science code from engineering. A tool that is also well integrated with the Pytorch Lightning package is Tensorboard. It has also been very useful during the model training process to visualise the model performance and other metrics. The work of training the models itself was performed on TalTech's server with GPU availability, which made the process of dealing with computationally heavy tasks and a lot of data much easier. The grid engine features available on the server were often used to perform large computations without disrupting others' work.

5.2 Model architecture

The deep neural network models for language identification are trained on the datasets collected in previous steps from YouTube. The datasets have been divided into training and validation sets. Validation sets contain most of the validated data which is known to have true labels. Model implementations are based on Tanel Alumäe's work, that the author of this thesis has contributed to and modified to experiment with noise adaption methods and other improvement techniques.

The language identification model's architecture is based on the x-vector approach described in Section 2.6. There are four main blocks of layers. Firstly, the input and feature extraction layers, that take raw audio as WAV files, convert it into numerical form, calculate the mel spectrograms and perform some other data transformations. Secondly, the pre-pooling component that contains convolutional layers, batch normalization and Rectified Linear Unit (ReLU) units. Then a statistical pooling layer and lastly a post-pooling group with linear layers, more batch normalization, ReLUs and a final softmax output layer. The x-vectors can be extracted from the last block of layers if needed. For some models, that were experimented with in this work, some additional layers were added that were expected to increase the performance.

Other parameters for the models that were used in this work were mostly similar. The Adam optimizer was used to update the weights during training. A relatively small learning rate of 0.0001 to 0.0005 was used depending on the model size. The input data was augmented with speed perturbation, noise and reverberation.

5.3 Noise robust loss functions

When training with the automatically collected YouTube data there may be noisy labels in the data. Some loss functions used in neural networks are somewhat robust to label noise, meaning that even if some percentage of the training data has false labels then the loss function itself is able to account for that (to some extent) and therefore improve the overall accuracy of the model. The following subsections describe some of such loss functions that were experimented with in this work. The results achieved with these losses are described in Section 6.2.

5.3.1 Soft bootstrapping loss

Soft bootstrapping loss (L_{soft}) has initially been proposed in [40] for the computer vision and image recognition domain. It has then been successfully used in the audio domain by [14] and [49]. The loss function dynamically updates target labels based on the state of the model. The main goal is to pay more attention to the model predictions, which should be more reliable as the learning continues and less attention to the noisy labels. The loss function is shown in Equation 5

$$L_{soft} = -\sum_{k=1}^{K} [\beta y_k + (1-\beta)\hat{y}_k] \log \hat{y}_k, \quad \beta \in [0,1] \tag{5}$$

where β is the parameter used to assign the weight of each component in the total loss, y_k is the k-th element of the target label, $\hat{y}_k$ is the k-th element of the network predictions and K is the total number of classes. The value of β used in this work is 0.3.

5.3.2 L_q loss

The L_q loss is a combination of the CCE and Mean Absolute Error (MAE) losses introduced in [62]. By following [14] it can be written as

$$L_q = \frac{1 - (\sum_{k=1}^{K} y_k \hat{y}_k)^q}{q}, \quad q \in [0,1] \tag{6}$$

By using a generalized combination of the two losses a better robustness to noise should be possible. CCE itself assigns less importance to the predictions that differ much from the target labels, which should increase the noise robustness. On the other hand MAE weighs all predictions equally, which should make it robust against false labels. The value of q in this work is 0.7.

5.3.3 Batchwise loss masking

Another approach to minimize the loss caused by noisy labels is the use of loss masking. The used method is based on [20]. By summing together the cross entropy losses for single data points in a minibatch, we get the conventional loss for a minibatch

$$L = \sum_{n} \sum_{c} t_{n,c} \log(y_n, c) \tag{7}$$

where t is the vector of true labels and y is the vector of network predictions.

Since some of the collected data has been validated and hand annotated we can treat these data samples as correctly labelled. All of the other data might contain false labels and can be partially ignored. Now the loss function can be customized to take advantage of this knowledge

$$\sum_n m_n \sum_c t_{n,c} \log(y_n, c) \tag{8}$$

where m_n is now 1 if the n-th data point is validated and correct and otherwise 0.

The authors of [20] proposed two conditions that should be evaluated when deciding the value for m. If the data is verified then it is a true label, otherwise if some data has very high loss in the model then it can be considered an outlier with potentially false label

$$m_n = \begin{cases} 1 & \text{if } v_n = 1 \text{ or } C_n < \mu \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where v_n shows if the n-th data is validated or not. μ is used as $\mu = \alpha \times \underset{n}{max}\, C_n$ and the value for α is 0.8.

In this work this final loss function was partially modified, by following [19] and ignoring the top n data samples in a batch with the highest loss values, rather than setting a threshold value. This also means that some of the potentially noisy data is ignored during training. The value used for n in this work was 10. Increasing the amount of masked elements started to degrade the network accuracy. Less ignored batches showed no difference when compared to the baseline model.

5.3.4 Additive angular margin loss

A loss function called Additive Angular Margin (AAM) loss that has initially been developed for face recognition [11] (called ArcFace) and has then been adapted to speaker recognition [2] has shown promising results and outperforming many other methods. In current thesis this loss function has been used for language identification purposes. The additive angular margin loss has been created by modifying the regular softmax loss and can be seen in more detail in [58].

In this work AAM loss was used in conjunction with NLL loss from the baseline model. The model was trained for 10–15 epochs with NLL loss and then the loss function was switched to AAM. With the initial 10–15 epochs the model was able to gain almost the maximum accuracy that can be achived with NLL loss. Then for the following epochs it could be tuned further by using the additive angular margin loss.

6 Results and validation

This section gives an overview of the results that were achieved with the language identification models built on YouTube data. A baseline model is described and results from models with different noise robust loss functions are shown. Model performance on validation data is also provided. Most of the results are from models trained on a smaller subset of languages but one of the sections shows the results from a large 107 language identification model. In the end validation results on two other datasets are also shown.

6.1 Baseline model

Most of the development of language identification models was done on a smaller subset of languages, initially eight and later ten. In the end the languages used in this smaller collection were the ones that had the most validation results or that just were generally important. In the 10 language subset were: Estonian, English, Finnish, Russian, Arabic, German, Latvian, Swedish, French and Spanish. Since training the language identification models takes a long time, the development process is much more easier and rapid if there are less languages to work with.

The main model for ten languages was trained with the Negative Log Likelihood (NLL) loss as a baseline. This model trained in total for 43 epochs, which took about 7 hours. The best accuracy of 95.49% was achieved after the 33rd epoch, as seen in Figure 18.

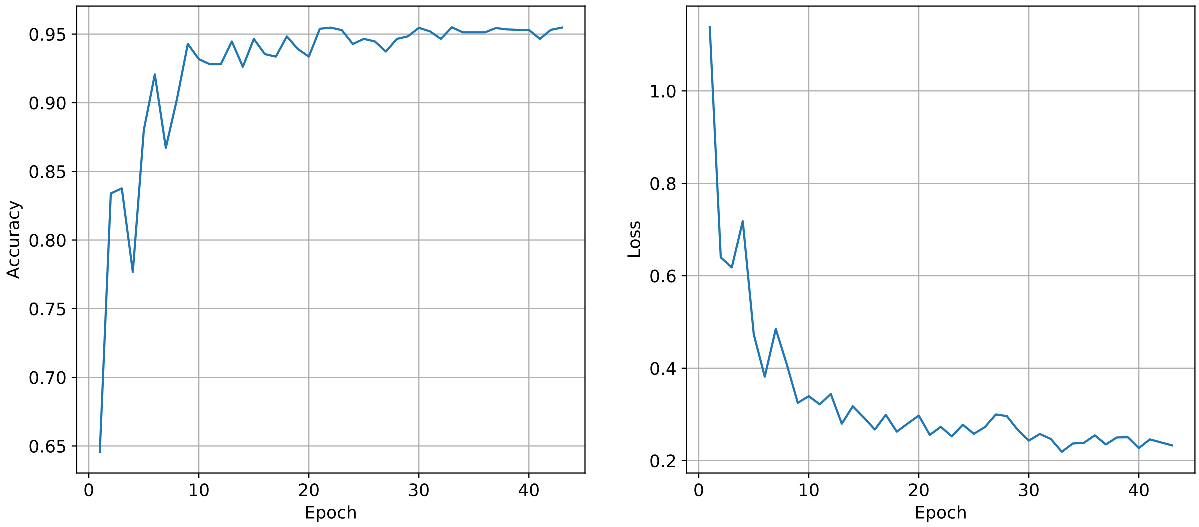
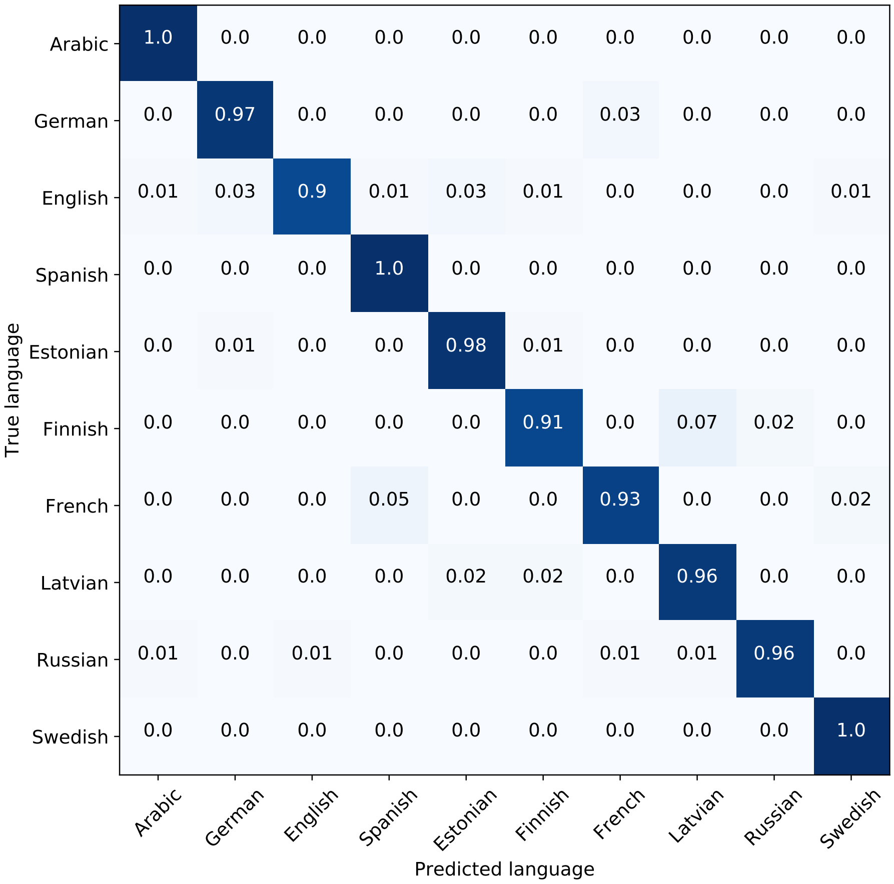


Figure 18. Validation loss and accuracy for the baseline 10 language model (trained with NLL loss).

The confusion matrix for the same model, in Figure 19 shows that Arabic, Spanish and Swedish are the languages with the highest accuracies, where all of the validation data points used as inputs have been predicted correctly. English has the worst performance with an accuracy of 0.9. Also the Finnish language is predicted on 7% of the times as Latvian which is quite interesting. For humans these last two languages do not sound too similar, but for the model the confusion rate is higher than for other languages.



True language \ Predicted language	Arabic	German	English	Spanish	Estonian	Finnish	French	Latvian	Russian	Swedish
Arabic	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
German	0.0	0.97	0.0	0.0	0.0	0.0	0.03	0.0	0.0	0.0
English	0.01	0.03	0.9	0.01	0.03	0.01	0.0	0.0	0.0	0.01
Spanish	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Estonian	0.0	0.01	0.0	0.0	0.98	0.01	0.0	0.0	0.0	0.0
Finnish	0.0	0.0	0.0	0.0	0.0	0.91	0.0	0.07	0.02	0.0
French	0.0	0.0	0.0	0.05	0.0	0.0	0.93	0.0	0.0	0.02
Latvian	0.0	0.0	0.0	0.0	0.02	0.02	0.0	0.96	0.0	0.0
Russian	0.01	0.0	0.01	0.0	0.0	0.0	0.01	0.01	0.96	0.0
Swedish	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Figure 19. Confusion matrix for the baseline 10 language model (trained with NLL loss).

Figure 20 shows the model accuracy when the speech segments used as inputs have been grouped by their lengths (rounded to the nearest second) from 2 to 20 seconds. The results are quite as expected, the longer the input audio sequence is the more accurate the network prediction is. Dealing with really short utterance language detection is a research topic on its own.

The average prediction accuracy of 80% for the really short clips with a duration of only two seconds is actually quite good. Also the clips with a duration of three seconds seem to performing well, compared to the audios with a duration that is a few seconds longer.

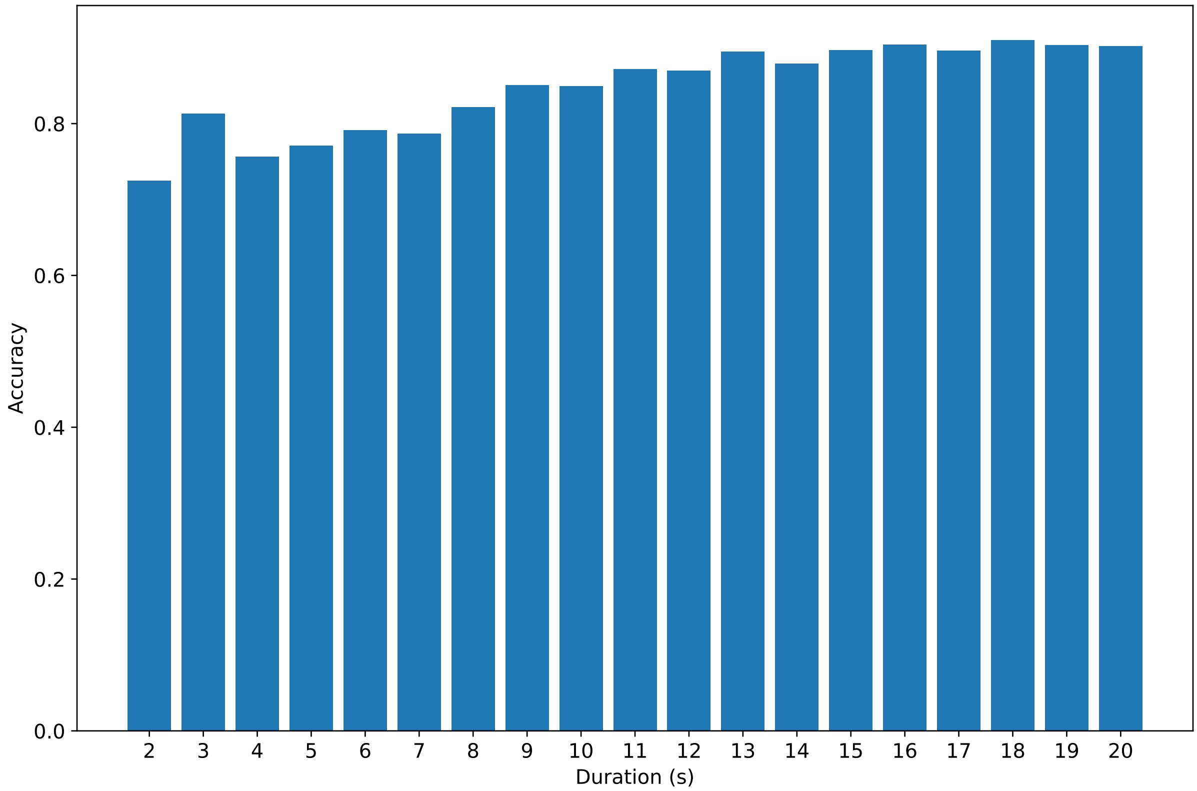


Figure 20. Language identification model accuracies grouped by the input segment durations.

6.2 Models with noise robust loss functions

In this section the experiment results for the models that use the different loss functions shown earlier are described and compared with the baseline model.

6.2.1 Soft bootstrapping loss

The first model was trained with the soft boostrapping (L_{soft}) loss. The training cycle ended after 76 epochs and the best accuracy of 97.05% was achieved after the 75-th epoch as shown in Figure 21. Training process took in total about 9 hours. This model had the best overall accuracy out of all the loss functions that were experimented with. Compared with the baseline model, the prediction accuracy increased by 1.56%.

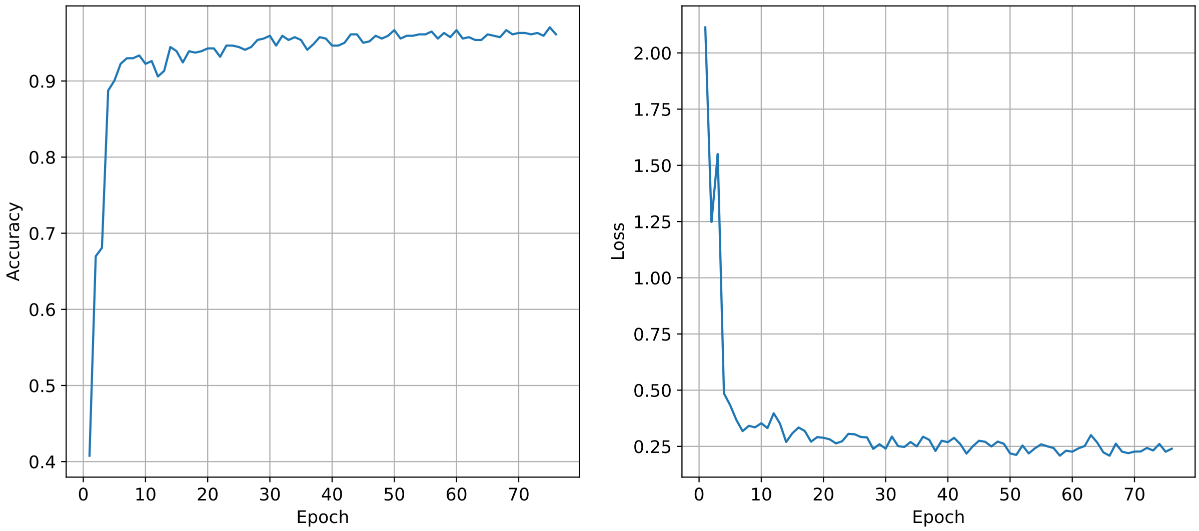
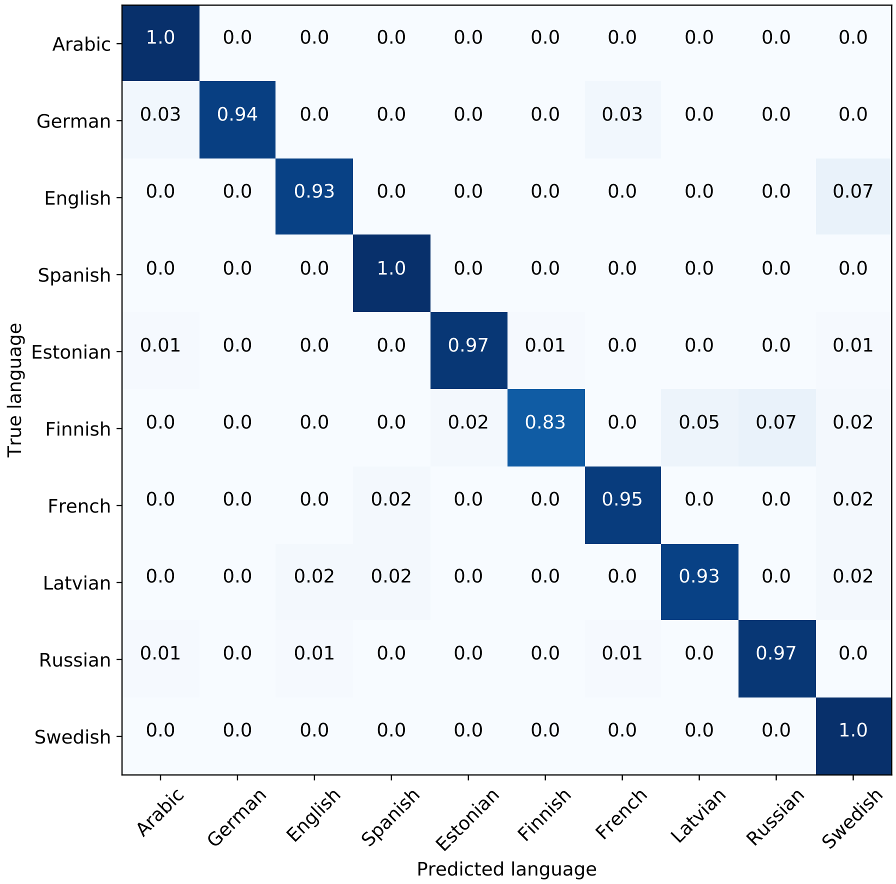


Figure 21. Validation loss and accuracy for the 10 language model that was trained with L_{soft} loss.

The same model's confusion matrix in Figure 22 shows that the top three languages, with the highest accuracies are again Arabic, Spanish and Swedish. The overall accuracy for all of the languages has improved, when comparing to the baseline model. The least performing language is Finnish again with an accuracy of 84%. With this model, Finnish is being wrongly classified as Latvian and Russian.



True language \ Predicted language	Arabic	German	English	Spanish	Estonian	Finnish	French	Latvian	Russian	Swedish
Arabic	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
German	0.03	0.94	0.0	0.0	0.0	0.0	0.03	0.0	0.0	0.0
English	0.0	0.0	0.93	0.0	0.0	0.0	0.0	0.0	0.0	0.07
Spanish	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Estonian	0.01	0.0	0.0	0.0	0.97	0.01	0.0	0.0	0.0	0.01
Finnish	0.0	0.0	0.0	0.0	0.02	0.83	0.0	0.05	0.07	0.02
French	0.0	0.0	0.0	0.02	0.0	0.0	0.95	0.0	0.0	0.02
Latvian	0.0	0.0	0.02	0.02	0.0	0.0	0.0	0.93	0.0	0.02
Russian	0.01	0.0	0.01	0.0	0.0	0.0	0.01	0.0	0.97	0.0
Swedish	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Figure 22. Confusion matrix for the 10 language model, trained with L_{soft} loss.

6.2.2 L_q loss

Identification model trained with the L_q loss had no real improvements. It was trained for 35 epochs and a best accuracy of 95.49% was achieved, which is same as the baseline model. Figures 23 and 24 show the validation accuracy, loss and confusion matrix for this model.

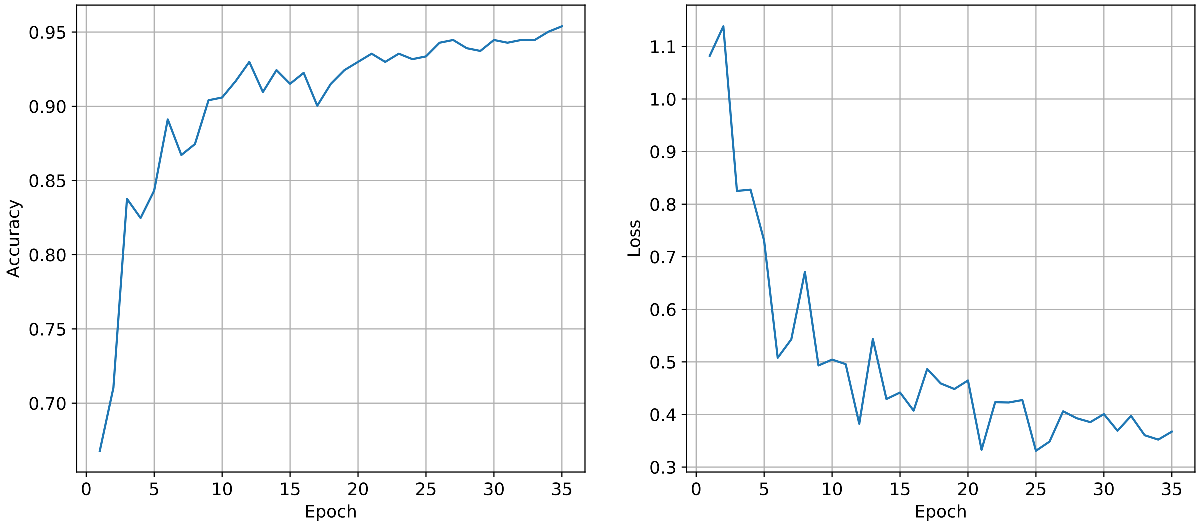
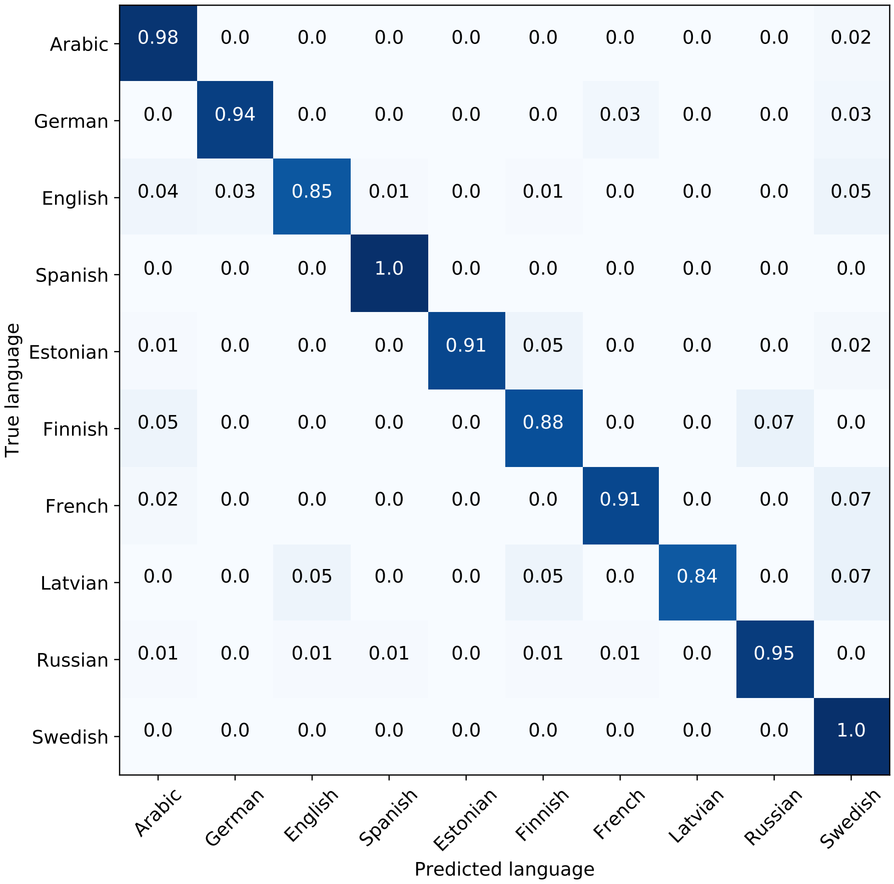


Figure 23. Validation loss and accuracy for the 10 language model that was trained with L_q loss.



	Arabic	German	English	Spanish	Estonian	Finnish	French	Latvian	Russian	Swedish
Arabic	0.98	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.02
German	0.0	0.94	0.0	0.0	0.0	0.0	0.03	0.0	0.0	0.03
English	0.04	0.03	0.85	0.01	0.0	0.01	0.0	0.0	0.0	0.05
Spanish	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Estonian	0.01	0.0	0.0	0.0	0.91	0.05	0.0	0.0	0.0	0.02
Finnish	0.05	0.0	0.0	0.0	0.0	0.88	0.0	0.0	0.07	0.0
French	0.02	0.0	0.0	0.0	0.0	0.0	0.91	0.0	0.0	0.07
Latvian	0.0	0.0	0.05	0.0	0.0	0.05	0.0	0.84	0.0	0.07
Russian	0.01	0.0	0.01	0.01	0.0	0.01	0.01	0.0	0.95	0.0
Swedish	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Figure 24. Confusion matrix for the 10 language model, trained with L_q loss.

6.2.3 Batchwise loss masking

A model with the batchwise loss masking approach was trained for 51 epochs and had a best accuracy of 96.31%. The used method was quite simple, but the results shown in Figure 25 were better than the baseline model.

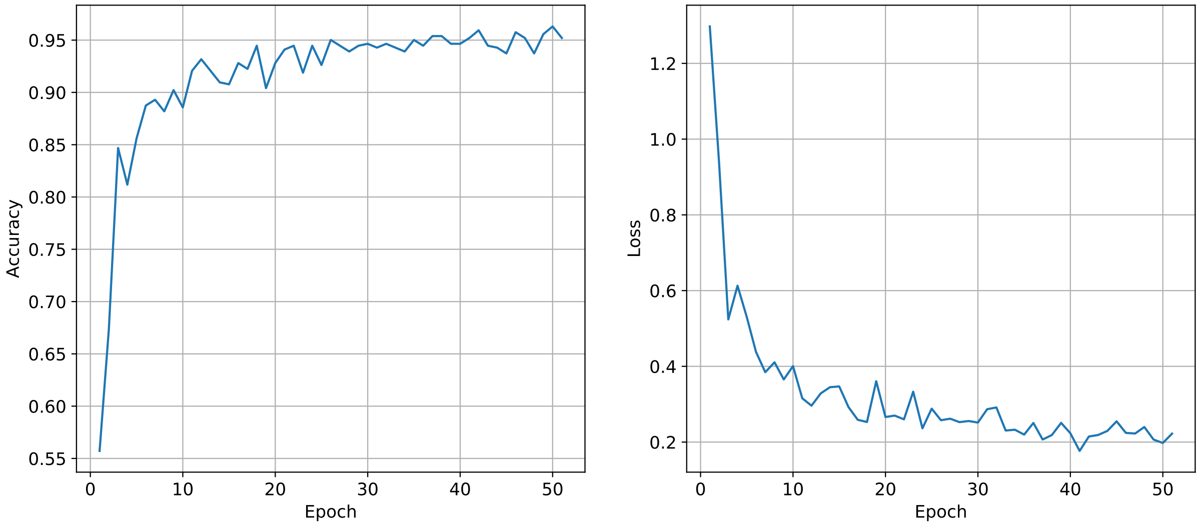
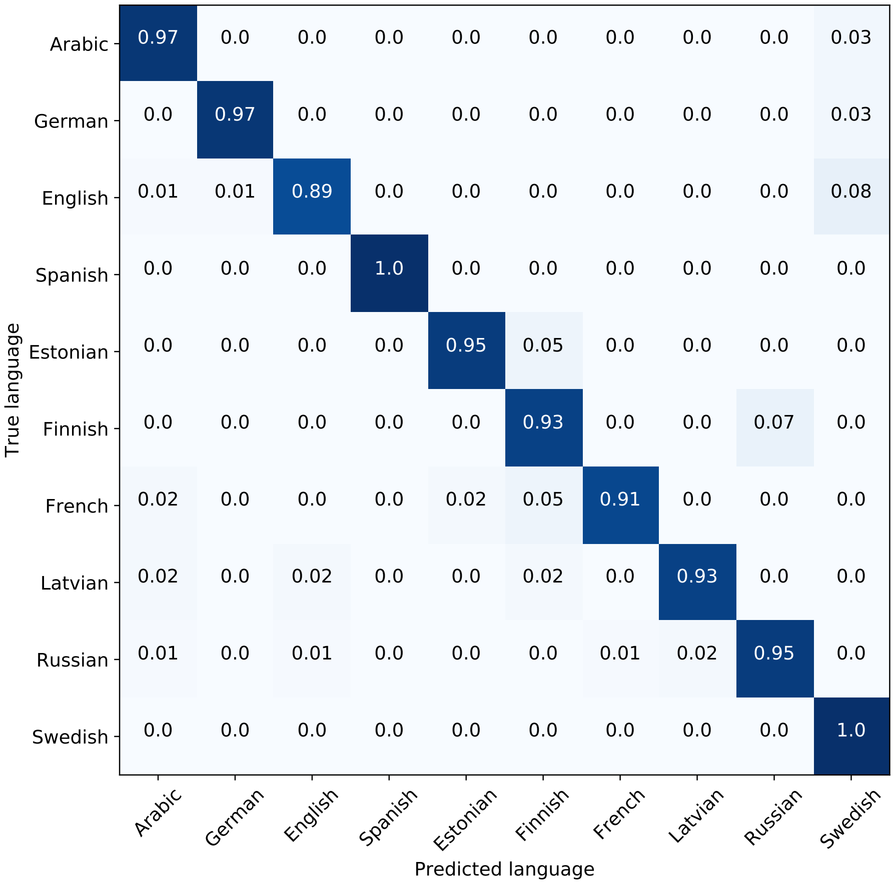


Figure 25. Validation loss and accuracy for the 10 language model that was trained with the batchwise loss masking approach.

For this model, the confusion matrix in Figure 26 shows that the accuracy for the Finnish language that had quite bad performance in the previous models, has actually grown.



	Arabic	German	English	Spanish	Estonian	Finnish	French	Latvian	Russian	Swedish
Arabic	0.97	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.03
German	0.0	0.97	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.03
English	0.01	0.01	0.89	0.0	0.0	0.0	0.0	0.0	0.0	0.08
Spanish	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Estonian	0.0	0.0	0.0	0.0	0.95	0.05	0.0	0.0	0.0	0.0
Finnish	0.0	0.0	0.0	0.0	0.0	0.93	0.0	0.0	0.07	0.0
French	0.02	0.0	0.0	0.0	0.02	0.05	0.91	0.0	0.0	0.0
Latvian	0.02	0.0	0.02	0.0	0.0	0.02	0.0	0.93	0.0	0.0
Russian	0.01	0.0	0.01	0.0	0.0	0.0	0.01	0.02	0.95	0.0
Swedish	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Figure 26. Confusion matrix for the 10 language model, trained with the batchwise loss masking approach.

6.2.4 AAM loss

In terms of validation accuracy the AAM loss performed the worse with an accuracy of 94.83%, as in Figure 27.

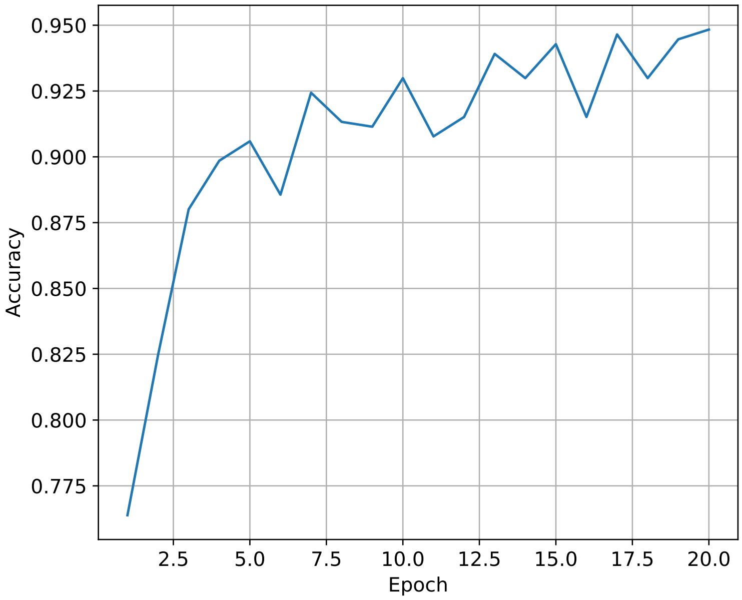


Figure 27. Validation accuracy for the 10 language model that was trained with the AAM loss.

6.2.5 Conclusion on the noise robust losses

Overall the use of noise robust loss functions helps to increase the network accuracy by a few percentage points. For evaluating and comparing the different loss functions also the Detection Error Tradeoff (DET) [29] curves were calculated in Figure 28. The DET curves display the false negative rate vs the false positive rate, while giving uniform treatment to both types of error. As seen from the figure, most of the noise robust loss functions improve the overall results of the models. The baseline model with the NLL loss is initially performing the worst. According to this figure the L_q loss is performing quite well, but it also has a slight curve when the false positive rate is around 40 percent.

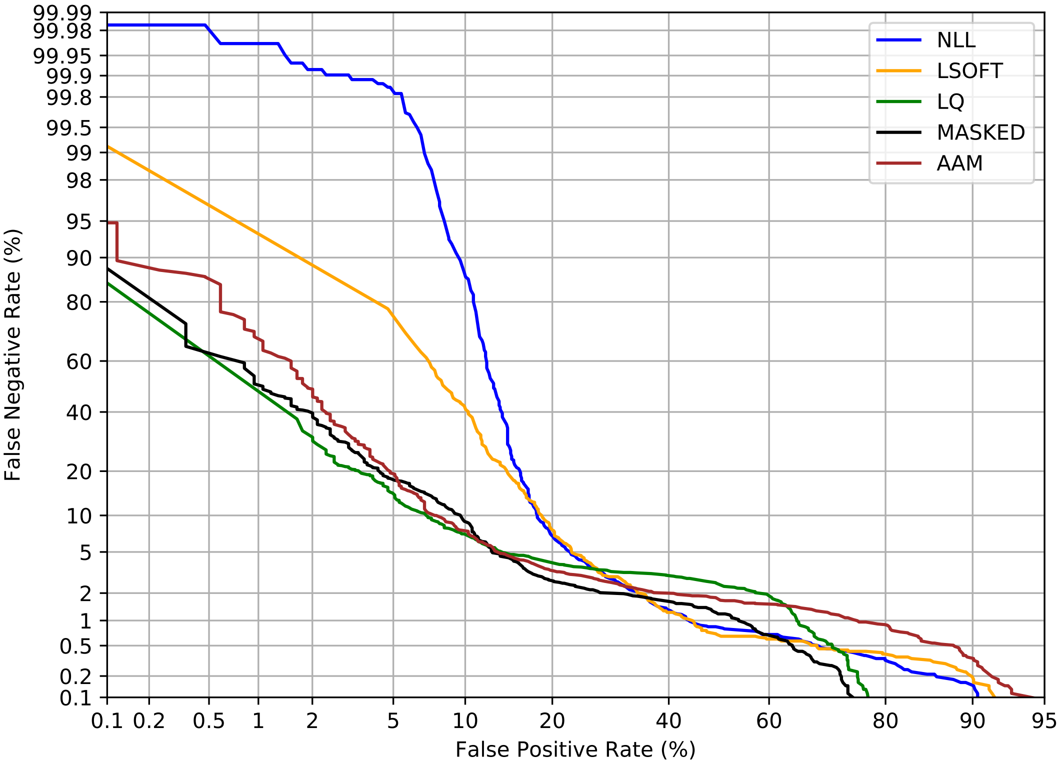


Figure 28. DET curve for five models that use different noise robust loss functions.

6.3 Identification model for 107 languages

Since the YouTube data had been collected for 107 languages, the last large step was to train a model on the entire dataset. Training the 107 language model took almost 10 days. As shown in Figure 29, a best accuracy of 91.11% was achieved after the 28th epoch. For a model this size that is able to distinguish between 107 input languages this is quite good.

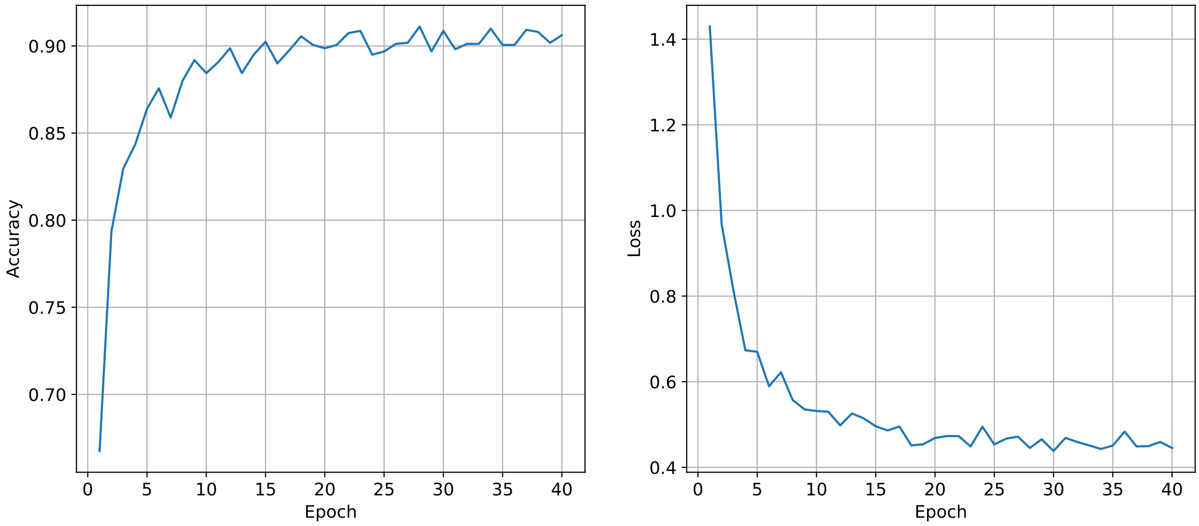


Figure 29. Validation accuracy and loss for the 107 language model.

6.4 Validation on other datasets

With the help and contributions from Tanel Alumäe it was possible to get the model evaluation results on KALAKA-3 and the proprietary LRE07 dataset. KALAKA-3 was described in Section 2.1.2 and contains training data from TV broadcasts and tuning and testing data from YouTube. LRE07 [17] is a dataset created by the National Institute of Standards and Technology (NIST) for the purpose of language recognition evaluation and contains telephone speech data.

6.4.1 KALAKA-3

The KALAKA-3 dataset has four different evaluation sets Plenty-Closed (PC), Plenty-Open (PO), Empty-Closed (EC) and Empty-Open (EO). The PC and PO cover classification for six languages (Basque, Catalan, English, Galician, Portugese and Spanish). The EC and EO sets handle classification for four languages (German, Greek, French, Italian).

For KALAKA-3 a 21 language model was used that was trained on a subset of the 107 language data covering all of the KALAKA-3 development and evaluation data. For the first model the final logistic regression classifiers were trained on the x-vectors extracted from our own language data. In the second approach KALAKA-3 development data was used. Table 6 shows the evaluation results using KALAKA-3 official evaluation metric F_{act} and EER. The baseline is a fusion of multiple i-vector and phonotactic systems reported in [43]. In the four language tasks (EC and EO) our systems have quite similar scores. For the PC and PO tasks using the official KALAKA-3 development data has better results, probably because the automatically collected data that is in our datasets has a high confusion rate between Catalan, Spanish and other similar languages.

Table 6. Results on KALAKA-3 dataset in terms of F_{act}/EER (lower is better).

Training data	**PC**	**PO**	**EC**	**EO**
Baseline [43]	0.079/5.74	0.115/6.67	0.104/6.16	0.169/6.96
Our x-vectors (our dev. data)	0.124/7.23	0.150/8.32	0.028/0.32	0.083/3.08
Our x-vectors (KALAKA-3 dev. data)	0.055/4.36	0.083/5.95	0.033/0.32	0.059/3.68

6.4.2 LRE07

The LRE07 dataset contains 26 language and dialect categories that are used as detection targets [17]. As mentioned before, the LRE dataset contains telephone speech, which has completely different domain when compared to the YouTube audio data collected in this work. Table 7 shows the results between different systems that are evaluated on the LRE dataset. The first one is the model from this work that uses the x-vector approach and is trained on the automatically collected YouTube data.

Comparing the system from this work, that was trained on the collected YouTube data with some older systems that used to be state-of-the-art before i-vectors and x-vectors, then it can be seen that our work outperforms them in all cases.

When looking at the difference between our system and the i-vector systems [47] with in-domain data, then our system has better results in terms of the C_{avg} metric and the EER values are similar.

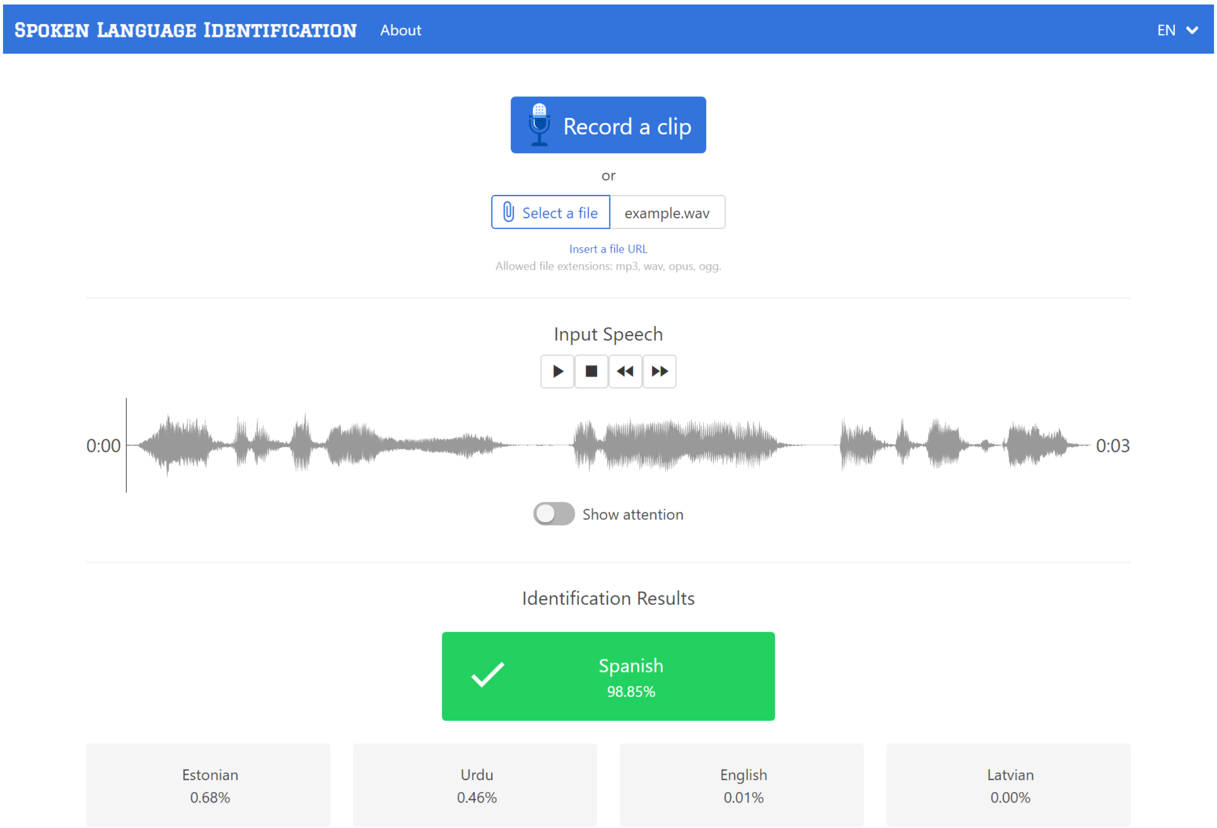
Other newer systems [25], [8] that use similar models and in-domain data perform better as can be expected from the in-domain and out-domain data difference.

Table 7. Results from different systems on the LRE07 dataset.

System	**3 sec**		**10 sec**		**30 sec**		**Average**	
Current model (YouTube data)	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**
x-vector DNN	14.46	30.31	5.89	14.18	2.62	7.65	7.65	17.38
Old state-of-the-art (in-domain data) [30]	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**
GMM256-MMI	18.43	-	8.61	-	4.15	-	10.40	-
GMM256-MMI-chcf	20.98	-	9.81	-	3.73	-	11.51	-
GMM2048-eigchan	17.14	-	7.38	-	2.76	-	9.09	-
GMM512-SVM	20.14	-	8.77	-	3.80	-	10.90	-
I-vector systems (in-domain data) [47]	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**
i-vector DNN	19.67	31.43	7.84	12.38	3.31	4.73	10.27	16.18
Modern systems (in-domain data) [25], [8]	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**	C_{avg}	**EER (%)**
CNN-SAP	8.59	9.89	2.49	4.27	1.09	2.38	4.06	5.51
CNN-LDE	8.25	7.75	2.61	2.31	1.13	0.96	4.00	3.67
DNN PPP features	8.99	6.90	2.20	1.43	0.61	0.32	3.93	2.88

6.5 Demo application

The trained spoken language identification models are quite interesting to experiment with, in terms of feeding random audio clips through them and looking at the predictions. To demonstrate the x-vector based models, trained on YouTube data in a more production like real environment and to make using them with simple test inputs easier, a demo application was built, shown in Figure 30. The application has two parts - a language identification API that can be used by different clients and a demo client that provides a web user interface for the same API.



example.wav

Spanish 98.85%

Estonian	Urdu	English	Latvian
0.68%	0.46%	0.01%	0.00%

Figure 30. A screenshot of the demo application for spoken language identification.

On the backend side the previously trained identification model is loaded from its best and final checkpoint and used in evaluation mode. The input to the service can be an audio file, which is then fed through the identification model and a prediction is made about the spoken language. The model is kept in application's memory and therefore the prediction speed is quite fast.

In the demo client users can upload a local audio clip or create a recording using their microphone. Then the input is processed and top five identified languages with their confidence scores are returned. Also an attempt was made on visualizing the input audio's attention. That being the most important parts of the audio clip that have the most effect when making the decision between the spoken languages.

7 Discussion and future work

In this work the experiments with using automatically collected web data, like the audio from YouTube videos for building spoken language identification models proved to be successful. There were two main problems that had to be dealt with. First, the task of developing a pipeline for collecting such datasets without many falsely labeled audio clips. Secondly, the use of such data to build the identification models that would work well on the partially noisy data.

The main advantage of using such automatically collected datasets is that they are quite easily reproducible, free and can have more languages than any of the already existing ones. The datasets in this work are in some ways indefinitely scalable, as long as there is enough content on YouTube. The predefined number of 100–150 hours of audio per language, that was used in this work, could be increased for most of the collected languages. Currently, data was collected for 107 languages, but this number can also be increased to some extent. For low resource languages the methods do not work as well as for languages with plenty of content on YouTube. Dataset collection for these smaller languages requires changes in the collection pipeline.

One of the main takeways from the work is that the quality of the collected data is very important when creating the identification models. The less falsely labeled data there is the higher the final model accuracies hopefully will be. If the initial data collection process could be improved further in such a way that even more of the false positive results would be filtered out then that would only be beneficial to the entire task. More heavier filtering could be applied in the speaker diarization step to remove the non-speech audio segments some of which were problematic in this work. This would increase the data label quality noticeably over the 92% achieved in this work.

The identification models were trained for 10 and 107 languages. Both of these models work well for the identification task when the input speech is clear and from native speakers. However the model's performance degrades when it is applied on non-native accented speech. In reality it is not possible to collect training datasets for all language and accent combinations, so extending the training datasets is not possible to improve the performance in that area. The solution to identifying the spoken language from accented speech is a research topic on its own, that probably requires a completely different and unique approach to be applied in conjunction with the current models.

In the real world when such identification models are used in practise there usually is a set of target languages which are known to be in the input data that needs to be analyzed. This means that smaller and more specific models can be created for the task at hand, which should increase the performance. If it is known that the data that needs to be analyzed can contain only three possible languages then there is no need to deploy a 107 language model which probably has lower accuracies.

8 Conclusion

The goal in this thesis was to use audio data extracted from open source media like YouTube for the purpose of building spoken language identification models. The task required two problems to be solved. First, the speech datasets for many different languages had to be collected. For that a data collection pipeline had to be developed that would find the required content from the internet and also filter out as many of the data that does not fit the predefined requirements. Secondly, the automatically collected datasets had to be used in the language identification models, while taking into account the noisy properties of such web data.

In the data collection stage Wikipedia and YouTube were used as the multilingual data sources. Wikipedia dumps were used to generate the search phrases that could be used to find the required content from the YouTube platform. Using automatically collected data without knowing its exact contents inevitably leads to falsely labelled elements in the datasets. To filter out such problematic data a text based language identification model was introduced to the data collection step. The models could be applied on the search phrases and later the video results, which produces much cleaner datasets. Finally as a result of the data collection process a dataset for 107 languages was acquired, resulting in 14 044 hours of audio content.

Since the quality of the data in the automatically collected datasets is quite unknown a separate validation experiment was carried out. For that a custom web application was built that allowed to distribute the work to different people, who could then listen to random audio clips in the collected datasets and assign true language labels to each clip. The results from the validation process had two important outcomes. Firstly, they showed that the average "label accuracy" for the collected data is between 85-92% which is quite good. Secondly, by using the validation data, separate datasets could be created that were used for the validation steps when evaluating the identification models.

The spoken language identification models themselves were built by following the state-of-the-art approach of using the x-vectors. The collected YouTube audios were used for training the models. To account for the label noise in the web data different noise robust loss functions were experimented with.

The accuracy for the baseline model, that was able to identify 10 different languages was 95.49%, which is close to other identification models that have been trained on entirely validated datasets created for the exact purpose of language identification. A model able to identify 107 languages achieved an accuracy of 91.11% which is also quite impressive if taking into account the number of identifiable languages. The use of noise robust loss functions increased the models' accuracies by a few percentage points. On other datasets our model, that was trained on YouTube data, had similar or better results than older state-of-the-art and i-vector models that were trained on in-domain data. Current state-of-the-art models trained on in-domain data outperformed our model as expected from the data domain difference.

Overall the use of YouTube audios for many different languages for the task of language identification proved to be successful. Identification accuracies were close to other results, that used specifically created datasets and in-domain data. The large dataset collected for the purpose of spoken language identification can be used in future work and could also be useful to others who need to solve the same task. In terms of future work, the models could be improved in areas like recognizing the language from accented speech, distinguishing between similar languages and short input utterance identification. All of which can be research topics on their own.

References

[1] Charu C. Aggarwal. *Neural networks and deep learning. A Textbook.* Springer, 2018.

[2] Jahangir Alam et al. "ABC system description for NIST multimedia speaker recognition evaluation 2019". In: *NIST SRE 2019 Workshop.*

[3] Eliathamby Ambikairajah et al. "Language identification: a tutorial". In: *IEEE Circuits and Systems Magazine* 11.2 (2011), pp. 82–108.

[4] Christian Bartz et al. "Language identification using deep convolutional recurrent neural networks". In: *International Conference on Neural Information Processing.* Springer. 2017, pp. 880–889.

[5] Jacob Benesty, M Mohan Sondhi, and Yiteng Huang. *Springer handbook of speech processing.* Springer, 2007.

[6] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning.* Vol. 1. MIT press, 2017.

[7] Julien Boussard, Andrew Deveau, and Justin Pyron. *Methods for spoken language identification.* 2017.

[8] Weicheng Cai, Jinkun Chen, and Ming Li. "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system". In: *arXiv preprint arXiv:1804.05160* (2018).

[9] Najim Dehak et al. "Front-end factor analysis for speaker verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4 (2010), pp. 788–798.

[10] Najim Dehak et al. "Language recognition via i-vectors and dimensionality reduction". In: *Twelfth annual conference of the international speech communication association.* 2011.

[11] Jiankang Deng et al. "Arcface: Additive angular margin loss for deep face recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2019, pp. 4690–4699.

[12] Mireia Diez et al. "Bayesian HMM based x-vector clustering for speaker diarization". In: *Proc. Interspeech 2019* (2019), pp. 346–350.

[13] W.A. et al. Falcon. *PyTorch Lightning.* `https://github.com/PytorchLightning/pytorch-lightning`. 2019.

[14] Eduardo Fonseca et al. "Learning sound event classifiers from web audio with noisy labels". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 21–25.

[15] Daniel Garcia-Romero and Carol Y Espy-Wilson. "Analysis of i-vector length normalization in speaker recognition systems". In: *Twelfth annual conference of the international speech communication association*. 2011.

[16] Yoav Goldberg. "Neural network methods for natural language processing". In: *Synthesis Lectures on Human Language Technologies* 10.1 (2017), pp. 1–309.

[17] NIST LRE Group et al. *The 2007 NIST language recognition evaluation plan (LRE07)*. 2007.

[18] Jonathan Hanna and Nick Webb. *Deep Learning for low-resource automatic language recognition.* 2019.

[19] Kexin He, Yuhan Shen, and Wei-Qiang Zhang. *THUEE system for DCASE 2019 challenge task 2.* Tech. rep. Tsinghua University, 2019.

[20] Il-Young Jeong and Hyungui Lim. "Audio tagging system for DCASE 2018: focusing on label noise, data augmentation and its efficient learning". In: *DCASE Challenge* (2018).

[21] Ahilan Kanagasundaram et al. "A study of x-vector based speaker recognition on short utterances". In: Sept. 2019.

[22] Elie Khoury and Matt Garland. "I-vectors for speech activity detection." In: *Odyssey*. 2016, pp. 334–339.

[23] Tom Ko et al. "A study on data augmentation of reverberant speech for robust speech recognition". In: *ICASSP*. 2017, pp. 5220–5224.

[24] Haizhou Li, Bin Ma, and Kong Aik Lee. "Spoken language recognition: from fundamentals to practice". In: *Proceedings of the IEEE* 101.5 (2013), pp. 1136–1159.

[25] Shaoshi Ling, Julian Salazar, and Katrin Kirchhoff. "Contextual phonetic pretraining for end-to-end Utterance-level language and speaker recognition". In: *arXiv preprint arXiv:1907.00457* (2019).

[26] Weibo Liu et al. "A survey of deep neural network architectures and their applications". In: *Neurocomputing* 234 (2017), pp. 11–26.

[27] Ignacio Lopez-Moreno et al. "Automatic language identification using deep neural networks". In: *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2014, pp. 5337–5341.

[28] Zhanyu Ma et al. "Short utterance based speech language identification in intelligent vehicles with time-scale modifications and deep bottleneck features". In: *IEEE transactions on vehicular technology* 68.1 (2018), pp. 121–128.

[29] Alvin Martin et al. *The DET curve in assessment of detection task performance*. Tech. rep. National Inst of Standards and Technology Gaithersburg MD, 1997.

[30] Pavel Matějka et al. "BUT language recognition system for NIST 2007 evaluations". In: *Ninth Annual Conference of the International Speech Communication Association*. 2008.

[31] Gregoire Montavon. "Deep learning for spoken language identification". In: *NIPS Workshop on deep learning for speech recognition and related applications*. 2009, pp. 1–4.

[32] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. "Voxceleb: a large-scale speaker identification dataset". In: *arXiv preprint arXiv:1706.08612* (2017).

[33] Michael A Nielsen. *Neural networks and deep learning*. Vol. 2018. Determination press San Francisco, CA, USA: 2015.

[34] Adam Paszke et al. "PyTorch: an imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[35] Josh Patterson and Adam Gibson. *Deep learning: a practitioner's approach*. "O'Reilly Media, Inc.", 2017.

[36] Trang Pham et al. "Deepcare: a deep dynamic memory model for predictive medicine". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2016, pp. 30–41.

[37] Daniel Povey et al. “The Kaldi speech recognition toolkit”. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, Dec. 2011.

[38] Zhaodi Qi, Yong Ma, and Mingliang Gu. “A study on low-resource language identification”. In: *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE. 2019, pp. 1897–1902.

[39] Yoav Ramon. *Speaker diarization with Kaldi*. 2019. URL: `https://towardsdatascience.com/speaker-diarization-with-kaldi-e30301b05cc8` (visited on 03/21/2020).

[40] Scott Reed et al. “Training deep neural networks on noisy labels with bootstrapping”. In: *arXiv preprint arXiv:1412.6596* (2014).

[41] Fred Richardson, Douglas Reynolds, and Najim Dehak. “A unified deep neural network for speaker and language recognition”. In: *arXiv preprint arXiv:1504.00923* (2015).

[42] Fred Richardson, Douglas Reynolds, and Najim Dehak. “Deep neural network approaches to speaker and language recognition”. In: *IEEE signal processing letters* 22.10 (2015), pp. 1671–1675.

[43] Luis Javier Rodrıguez-Fuentes et al. “KALAKA-3: a database for the assessment of spoken language recognition technology on youtube audios”. In: *Language Resources and Evaluation* 50.2 (2016), pp. 221–243.

[44] Luis Javier Rodrıguez-Fuentes et al. “KALAKA-3: a database for the recognition of spoken European languages on YouTube audios.” In: *LREC*. 2014, pp. 443–449.

[45] Mickael Rouvier et al. “An open-source state-of-the-art toolbox for broadcast news diarization”. In: *Interspeech*. 2013.

[46] Ramon Sanabria et al. “How2: a large-scale dataset for multimodal language understanding”. In: *arXiv preprint arXiv:1811.00347* (2018).

[47] Mousmita Sarma. *Script for NIST 2007 general language recognition closed-set evaluation with the Kaldi framework*. URL: `https://github.com/kaldi-asr/kaldi/blob/master/egs/lre07/v2/run.sh` (visited on 05/06/2020).

[48] Shikhar Shukla2 B Sarthak and Govind Mittal. “Spoken language identification using convnets”. In: *Ambient Intelligence: 15th European Conference, AmI 2019, Rome, Italy, November 13–15, 2019, Proceedings*. Vol. 11912. Springer Nature. 2019, p. 252.

[49] Shubhr Singh, Arjun Pankajakshan, and Emmanouil Benetos. *Audio tagging using linear noise modelling layer*. 2019.

[50] David Snyder, Guoguo Chen, and Daniel Povey. “Musan: a music, speech, and noise corpus”. In: *arXiv preprint arXiv:1510.08484* (2015).

[51] David Snyder et al. “Deep neural network embeddings for text-independent speaker verification.” In: *Interspeech*. 2017, pp. 999–1003.

[52] David Snyder et al. “Speaker recognition for multi-speaker conversations using x-vectors”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 5796–5800.

[53] David Snyder et al. “Spoken language recognition using x-vectors.” In: *Odyssey*. 2018, pp. 105–111.

[54] David Snyder et al. “X-vectors: robust dnn embeddings for speaker recognition”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 5329–5333.

[55] Pedro Javier Ortiz Suárez, Benoıt Sagot, and Laurent Romary. “Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures”. In: *Challenges in the Management of Large Corpora (CMLC-7) 2019* (2019), p. 9.

[56] Pulkit Verma and Pradip K Das. “I-vectors in speech processing applications: a survey”. In: *International Journal of Speech Technology* 18.4 (2015), pp. 529–546.

[57] Wikimedia. *List of Wikipedias*. 2020. URL: `https://meta.wikimedia.org/wiki/List_of_Wikipedias` (visited on 03/26/2020).

[58] Xu Xiang et al. “Margin matters: towards more discriminative deep neural network embeddings for speaker recognition”. In: *arXiv preprint arXiv:1906.07317* (2019).

[59] Yan Xu et al. “Improved i-vector representation for speaker diarization”. In: *Circuits, Systems, and Signal Processing* 35.9 (2016), pp. 3393–3404.

[60] Shoou-I Yu, Lu Jiang, and Alexander Hauptmann. “Instructional videos for unsupervised harvesting and learning of action examples”. In: *Proceedings of the 22nd ACM international conference on Multimedia*. 2014, pp. 825–828.

[61] Ruben Zazo et al. “Language identification in short utterances using long short-term memory (LSTM) recurrent neural networks”. In: *PloS one* 11.1 (2016).

[62] Zhilu Zhang and Mert Sabuncu. “Generalized cross entropy loss for training deep neural networks with noisy labels”. In: *Advances in neural information processing systems*. 2018, pp. 8778–8788.

Appendix 1 – Code for the work

The code written for most of the tasks in this thesis can be seen on GitHub.

Data collection scripts: `https://github.com/jorgenvvv/youtube-lid-data`

Audio validation application: `https://github.com/jorgenvvv/audio_validator`

Baseline spoken language identification model: `https://github.com/jorgenvvv/lang-id-model`

Language identification demo API: `https://github.com/jorgenvvv/lang-id-server`

Language identification demo client: `https://github.com/jorgenvvv/lang-id-client`

Appendix 2 – Collected data

Table 8. Parameters of the audio data collected for 107 languages.

#	Language	Number of files	Duration (h)	Size (GB)
1	Abkhazian	176	13.27	1.42
2	Afrikaans	895	132.73	14.24
3	Albanian	676	85.26	9.15
4	Amharic	380	91.95	9.87
5	Arabic	698	100.72	10.81
6	Armenian	768	107.61	11.55
7	Assamese	2478	188.43	20.22
8	Azerbaijani	718	79.80	8.56
9	Bashkir	910	79.03	8.48
10	Basque	585	41.38	4.44
11	Belarusian	1190	172.15	18.47
12	Bengali	697	69.14	7.42
13	Bosnian	809	143.91	15.44
14	Breton	870	62.74	6.73
15	Bulgarian	400	62.25	6.68
16	Burmese	874	51.44	5.52
17	Cambodian	516	46.55	5.00
18	Catalan	1383	125.47	13.46
19	Cebuano	135	7.76	0.83
20	Chinese	381	57.68	6.19
21	Croatian	910	158.53	17.01
22	Czech	537	82.85	8.89
23	Danish	507	56.81	6.10
24	Dutch	621	68.28	7.33
25	English	759	180.49	19.37
26	Esperanto	113	12.09	1.30
27	Estonian	561	150.26	16.12

#	Language	Number of files	Duration (h)	Size (GB)
28	Faroese	936	87.74	9.41
29	Finnish	573	122.03	13.09
30	French	681	113.48	12.18
31	Galician	951	103.17	11.07
32	Georgian	756	110.42	11.85
33	German	516	135.93	14.59
34	Greek	672	85.36	9.16
35	Guarani	87	3.00	0.32
36	Gujarati	783	60.76	6.52
37	Haitian	741	108.82	11.68
38	Hausa	883	110.20	11.82
39	Hawaiian	260	19.27	2.07
40	Hebrew	857	114.06	12.24
41	Hindi	962	105.95	11.37
42	Hungarian	585	88.52	9.50
43	Icelandic	1455	132.86	14.26
44	Indonesian	650	55.04	5.91
45	Interlingua	45	3.68	0.39
46	Italian	627	76.96	8.26
47	Japanese	820	79.96	8.58
48	Javanese	956	70.53	7.57
49	Kannada	790	62.90	6.75
50	Kazakh	920	91.51	9.82
51	Korean	680	92.31	9.91
52	Laotian	796	53.84	5.78
53	Latin	862	91.83	9.85
54	Latvian	686	100.04	16.20
55	Lingala	481	106.75	11.46
56	Lithuanian	602	99.78	10.71
57	Luxembourgish	829	92.19	9.89

#	Language	Number of files	Duration (h)	Size (GB)
58	Macedonian	1191	143.05	15.35
59	Malagasy	1013	131.10	14.07
60	Malay	1520	119.78	12.85
61	Malayalam	629	58.21	6.25
62	Maltese	1273	78.06	8.38
63	Manx	109	5.36	0.58
64	Maori	652	44.90	4.82
65	Marathi	1160	102.68	11.02
66	Mongolian	566	87.98	9.44
67	Nepali	460	83.72	8.98
68	Norwegian	1228	149.28	16.02
69	Norwegian Nynorsk	1096	89.57	9.61
70	Occitan	333	26.99	2.90
71	Panjabi / Punjabi	481	65.79	7.06
72	Pashto	726	55.99	6.01
73	Persian	584	96.81	10.39
74	Polish	706	95.15	10.21
75	Portuguese	697	77.35	8.30
76	Romanian	547	78.81	8.46
77	Russian	581	181.03	19.43
78	Sanskrit	164	25.69	2.76
79	Scots	71	4.49	0.48
80	Serbian	322	73.15	7.85
81	Shona	484	37.41	4.01
82	Sindhi	1386	105.00	11.27
83	Sinhalese	648	84.94	9.11
84	Slovak	428	55.94	6.00
85	Slovenian	1197	155.73	16.71
86	Somalia	807	120.98	12.98
87	Spanish	479	130.08	13.96

#	**Language**	**Number of files**	**Duration (h)**	**Size (GB)**
88	Sundanese	1140	87.28	9.37
89	Swahili	588	75.18	8.07
90	Swedish	481	68.95	7.40
91	Tagalog / Filipino	1134	119.56	12.83
92	Tajik	762	76.99	8.26
93	Tamil	531	64.15	6.88
94	Tatar	1379	128.89	13.83
95	Telugu	1033	99.42	10.67
96	Thai	686	78.17	8.39
97	Tibetan	550	117.24	12.58
98	Turkish	727	89.77	9.63
99	Turkmen	928	98.29	10.55
100	Ukrainian	550	72.68	7.80
101	Urdu	709	138.51	14.86
102	Uzbek	368	54.63	5.86
103	Vietnamese	573	81.80	8.78
104	Waray	319	14.70	1.58
105	Welsh	1501	100.26	10.76
106	Yiddish	573	62.44	6.70
107	Yoruba	547	107.25	11.51
Total		77 606	14 044.60	1004.38

Appendix 3 – Data validation results

Table 9. Data validation results.

Language	Given language	Not given language	No speech	Do not know	Total	Label accuracy
Estonian	3266	336	178	30	3810	85.7%
Armenian	1895	43	89	3	2030	93.3%
English	1021	83	112	4	1220	83.7%
German	1073	23	95	9	1200	89.4%
Finnish	997	70	100	3	1170	85.2%
Arabic	955	41	98	26	1120	85.3%
Russian	1000	14	55	1	1070	93.5%
Dutch	618	57	72	23	770	80.3%
French	548	12	36	24	620	88.4%
Latvian	489	24	88	9	610	80.2%
Swedish	480	19	54	17	570	84.2%
Urdu	339	127	44	0	510	66.5%
Persian	394	33	18	5	450	87.6%
Spanish	340	5	42	3	390	87.2%
Chinese	284	12	12	2	310	91.6%
Danish	244	8	34	14	300	81.3%
Turkish	211	3	22	4	240	87.9%
Norwegian	151	11	28	10	200	75.5%
Azerbaijani	163	8	7	2	180	90.6%
Slovenian	127	12	28	3	170	74.7%
Ukrainian	115	20	33	2	170	67.6%
Serbian	118	19	17	6	160	73.8%
Japanese	121	5	21	3	150	80.7%
Icelandic	110	9	15	6	140	78.6%
Italian	115	1	11	3	130	88.5%
Lithuanian	112	8	8	2	130	86.2%

Language	**Given language**	**Not given language**	**No speech**	**Do not know**	**Total**	**Label accuracy**
Slovak	66	26	12	6	110	60.0%
Pashto	96	13	1	0	110	87.3%
Afrikaans	61	6	3	0	70	87.1%
Hindi	54	4	2	0	60	90.0%
Yiddish	32	5	13	0	50	64.0%
Telugu	36	1	2	1	40	90.0%
Norwegian Nynorsk	16	6	5	3	30	53.3%
Hungarian	23	0	7	0	30	76.7%
Macedonian	19	0	0	1	20	95.0%
Tajik	18	1	1	0	20	90.0%
Greek	18	1	1	0	20	90.0%
Croatian	16	0	2	2	20	80.0%
Polish	15	0	4	1	20	75.0%
Portuguese	5	2	2	11	20	25.0%
Scots	3	7	7	3	20	15.0%
Latin	10	2	1	7	20	50.0%
Kazakh	9	0	1	0	10	90.0%
Albanian	10	0	0	0	10	100.0%
Bosnian	7	2	1	0	10	70.0%
Swahili	9	0	1	0	10	90.0%
Bengali	6	0	4	0	10	60.0%
Czech	8	0	1	1	10	80.0%
Bulgarian	8	0	1	1	10	80.0%
Georgian	9	1	0	0	10	90.0%