Institute for Natural Language Processing
Computational Linguistics

University of Stuttgart
Pfaffenwaldring 5b
D–70569 Stuttgart

Diplomarbeit

Language Identification for German-Turkish Code-Switching Speech

Uğur Köstak

Course of Study:	Informatik
Examiner:	Jun. -Prof. Dr. Ngoc Thang Vu
	Dr. Antje Schweitzer
Supervisor:	Dr. Özlem Cetinoğlu
	Jun. -Prof. Dr. Ngoc Thang Vu
Commenced:	September 15, 2016
Completed:	March 31, 2017
CR-Classification:	I.2.7

Abstract

The importance of computers has risen in recent years in our daily lives. An average person interacts without a doubt multiple times with computers. The wide usage of computers has caused researchers to think of ways which would allow you to communicate with computers by a minimum number of interactions. Speech is the main communication instrument for humans, so researchers also used speech as an interaction method between humans and computers. However, speech has boundaries of its own, the language varies in different societies, especially in multicultural societies where people tend to use a mixed language called Code-Switching language to communicate, i.e. Germany is a multicultural country and foreigners, especially bilingual Turkish people, use German and Turkish when they speak to each other. On the other hand, computers nowadays have become more powerful and can also process complex tasks such as NLP tasks, which requires a lot of processing power. In this thesis we aimed to solve Language Identification task in German-Turkish code-switching speeches with two popular machine learning methods Support Vector Machines and Deep Neural Networks and at the end we compared the performances of these methods.

Die Bedeutung von Computern ist in den letzten Jahren in unserem alltäglichen Leben gestiegen. Die durchschnittliche Person interagiert sich ohne Zweifel mehrmals am Tag mit Computern um. Dieser verbreitete Einsatz hat dazu geführt, dass die Forscher nach Möglichkeiten suchen, die uns ermöglichen mit Computern durch die minimalste Anzahl möglicher Interaktionen zu kommunizieren. Sprechen ist das wichtigste Kommunikationsinstrument für Menschen, deswegen haben die Forscher auch die Sprache als Interaktionsmethode zwischen Mensch und Computer verwendet. Allerdings hat die Sprache ihre Gren-

zen, die Sprache variiert sich in verschiedenen Gesellschaften, vor allem in multikulturellen Gesellschaften, in denen Menschen dazu neigen eine gemischte Sprache namens Code-Switching Sprache zu benutzen. Deutschland beispielsweise ist ein multikulturelles Land wo Ausländer, vor allem zweisprachige Türken sowohl Deutsch als auch Türkisch beim kommunizieren benutzen. Dennoch sind Computern heute leistungsstärker geworden und können auch komplexe Aufgaben wie NLP-Aufgaben verarbeiten, die viel Rechenleistung erfordern. In dieser Arbeit zielen wir darauf hin, die Sprachidentifizierungsaufgabe in deutsch-türkischen Code-Switching Sprache mit zwei populären maschinellen Lernmethoden zu unterstützen. Support Vector Machines und Deep Neural Networks und ein Vergleich der Leistungen diese Methoden.

Acknowledgement

Foremost, I would like to thank my thesis supervisors Jun. -Prof. Dr. Ngoc Thang Vu and Dr. Özlem Çetinoğlu of the Institute of Natural Language Processing at University of Stuttgart for giving me the opportunity to work on this subject. They have supported me with patience and knowledge from begining to end.

I especially want to thank to Dafni Markopoulou for always standing beside me. Her encouragement and support have been my motivation to write my thesis. I would like to express my deep gratitude to my family, my mother, father, brother and elder sister, for supporting me spiritually and financially throughout my studies and my life and also for giving me the chance to follow my dreams.

Finally, I want to thank Talha Yılmaz, Panos Anestopoulos, Akvile Degutye, Cem Yazgılı and Şahin Ertekin for data collection and correction of the thesis and I wish to say thank you to all who contributed in any way to this thesis.

"I know one thing; that I know nothing."

Socrates

Contents

List of Figures

List of Tables

1 Introduction

Language identification is a common natural language processing problem and is defined roughly as a categorization task. It has its own challenges especially when a single dataset (spoken or written data) contains more than one language such as Code-Switching (CS) conversations and texts or mixed language speeches. There are plenty of options to define LID but one of the most common definition is:

Automatic language identification of speech is the process by which the language of a digitized speech utterance is recognized by a computer. It is one of several processes in which information is extracted automatically from a speech signal (Zissman, 1996). According to (Auer, 1999) CS is defined as in linguistic when a speaker alternates more than one languages in a conversation and it is a common linguistic phenomenon in multilingual communities (Adel et al., 2013).

It is a key piece of technology in many applications such as multilingual conversational systems, spoken language translation, multilingual speech recognition, and spoken document retrieval. LID is also a topic of great importance in areas of intelligence and security, where the language identities of recorded messages and archived materials need to be established before any information can be extracted. For voice surveillance over a telephone network, LID technology also makes huge amounts of online language routing possible. (Zissman, 1996)

Formally LID can be: Which of the N languages does O belong to, where N a set of languages and O the utterance? Additionally, N could be assigned to both a closed-set and an open-set of languages. The difference between them is that the output for open-set N could be one of the languages among N or none of N languages; while for an closed-set N the output has to be one N languages (Li et al., 2013). According to (Vu et al., 2013) and (Adel et al., 2013), CS tasks could be regarded speaker-dependent phenomenon,

as a result adapted models could outperform speaker-independent language models, because each individual speaker has unique language behaviour.

As mentioned prevouisly, LID is a key technology in NLP tasks and CS makes it more challenging and gives another perspective. The aim with this thesis to gain the multiple language recognition ability of a monolingual LID system fed with multilingual utterances by using two state of the art machine learning approaches. Our baseline system is the well known Support Vector Machines (SVMs), the other system being Deep Neural Networks (DNNs), which has recently became more popular. Moreover, we aim to show that DNNs outperform SVMs for LID in CS conversations.

The work is organised as follows: In section 1 LID and CS are introduced as well as the importance of LID, uses cases and issues are represented. After introducing the task and issues, related works and state of art technologies are explained in section 2. Section 3 clarifies key criteria related to the techniques and algorithms which are used in conducting the experiments. The data used in the experiments, is represented with some statistics in section 4, Section 5 describes the proposed approaches of thesis with background techniques to solve LID task. Additionaly, data collection process, annotation and trascription, and feature extraction are explained in this section. Each CS case is explained through a concrete example In Section 6, the performace of proposed methods are presented and evaluated, and in Section 7, conclusions being drawn and potential avenues for future research.

2 Related Works

In this section, related techniques and previous work in Language identification tasks are explained in more detail.

2.1 Acoustic-Phonetic Approaches

2.1.1 Gaussian Mixture Model (GMM) Classification

The Gaussian Mixture Model (GMM) is an unsupervised classification technique and no linguistic knowledge is required to use it. The motivation for using GMM LID is that each language has its own characteristic, such as different sounds and sound frequencies. In other words, spectral acoustic features of a speech utterance are independent of the language used, as such, GMM is used to approximate the overall acoustic phonetic distributions of a spoken language. In its simplest form, a GMM can be used to directly model the distribution of the acoustic features.

The notion behind GMM is to approximate the probability distribution of the samples from a class C, by a linear combination of K Gaussian distributions:

$$p(\mathbf{o}|C) = \Sigma_{i=1}^{K} \omega_i \mathcal{N}(\mathbf{o}|\mu_\mathbf{i}, \mathbf{\Sigma_i}) \tag{1}$$

Where the weights have to satisfy the constraint and N is the normal distribution:

$$\mathcal{N}(\mathbf{o}|\mu, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma_i}|^{1/2}} exp\left[-\frac{1}{2}(\mathbf{o}-\mu)^T \mathbf{\Sigma}_i^{-1}(\mathbf{o}-\mu)\right] \tag{2}$$

Given a collection of training feature vectors, a GMM is generally trained with the expectation maximization (EM) algorithm, where the model parameters are estimated using the ML criterion. In language identification, we train a GMM for each language for the recognition task formulated in equation 3.

An unknown speech utterance represented as acoustic feature vectors $O = \{\mathbf{o}(1), \mathbf{o}(2), ..., \mathbf{o}(T)\}$, in which o(t) extracted at the discrete frame t. Observed acoustic features will be matched with most likely language, so the language L will be labelled as:

$$L = argmax_l \; p(O|L_l) \tag{3}$$

which follows a maximum-likelihood (ML) criterion.(Li et al., 2013)

2.1.2 Vector Space Modelling for statistical approaches

There are successful approaches to representing speech utterances as a high-dimensional vector under the SVM method using the spectral features of speech utterances. According to (Li et al., 2013), there are two effective vector space modelling techniques to represent a speech utterance with spectral features and GMM parameters.

In the section 3.1 one of the acoustic extraction features is explained in detail. In addition, there are other techniques which can also extract acoustic features of a given speech utterance. Applying one of these approaches will result in a speech utterance being represented as a sequence of feature vectors, with these feature vectors being compared using SVM and producing SVM output vectors. A successful approach using this sequence kernel is the Generalized Linear Discriminative Sequence (GLDS). It takes the explicit polynomial expansion of the input feature vectors and applies the sequence kernel based on generalized linear discriminates. The polynomial expansion includes all the monomials of the features in a feature vector.

GMM super-vector can be an alternative to GLDS. GLDS is a nonparametric approach for language identification, despite that the GMM supervector offers a parametric alternative. Given a UBM (Universal Background Modelling) and a speech utterance drawn from a language, a GMM supervector will be derived by stacking the mean vectors of all the adapted mixture

components. In this way, a speech utterance is mapped to a high dimensional space using the mean parameters of a GMM.

Finally, for each target language, a one versus the rest SVM can be trained in the vector space with the target language being the positive set and all other competing languages being the negative set. A decision strategy can be devised to summarize the outputs of multiple one-versus the-rest SVMs during the runtime test.(Li et al., 2013)

2.2 Phonotactic Approaches

In the previous section is given information about how to discriminate languages with only using acoustic features. In this section we will have a closer look to higher level prelexical language recognition methods. The main idea behind the phonotactic approaches is spliting a given speech utterance in to sound tokens and use the sound tokens for the task rather than purely acoustic-phonetic features. A very widely used tokenization technique is GMM tokenization technique that converts a bunch of frame sequence into a Gaussian token. In practice phone recognizers are very commonly used to tokenize speech utterances. On that occasion it is an obvious that the performance of the language recognition task is influenced proportional by phone recognizers performance.

PRLM. There are several different approaches in this paragraph we are going to introduces couple of them. The first one which we will introduce is phone recognition followed by language modelling, a single-language phone recognizer is used to tokenize the input speech, i.e. English phone recognizer, to convert the input waveform into a sequence of phone symbols. The phone sequences are then analysed by the n-gram analyser and a language is hypothesized on the basis of maximum likelihood. (Zissman, 1995)

$$logP(Y|\lambda_l) = \Sigma_{j=1}^{J} log\ P_{\lambda_l}(w_j|w_{j-1}...w_{j-(n-1)}) \tag{4}$$

where Y is phone sequence, λ is language model, w is phones and J is the length of phone sequence, log likelihood of the phone sequences. This is the cross entropy between phone probability distribution of phone sequences and phone n-gram models. The log likelihood should be calculated for each language and the most likely language l will be returned as described in 3.

PPRLM. Second one is parallel phone recognizer followed by language modelling. The idea based on first model but there are small differences between them. For example a separate phone recognizer is used as front-end for each target language and in the back-end additional classifier to evaluate each score from single phone recognizer model. Finally the classifier map the given speech utterance to most likely language. PPRLM can be also expressed with the term a fusion of multiple PRLM subsystem. (Zissman, 1995)

$$logP(L_l|O) = \Sigma_{f=1}^{F} log \frac{P(Y_f|\lambda_{f,l})}{\Sigma_{i=1}^{N} P(Y_f|\lambda_{f,i})} \tag{5}$$

where O is the speech utterance, $P(Y_f|\lambda_f)$ is the likelihood score. Basically the idea is we have multiple PRLM scores and each score is normalized and the the log likelihood summed up. So we generated a final score and decision is based on this score.

The last one is a bit different than the previous two approaches. Again it has parallel phone recognizer in the front-end processing and a classifier as back-end to classify output vectors from each part, but the one in the middle has n-gram statistics for each language instead of language models. it named as vector space modelling approach. Basically the VSM in phonotactic approaches is a analogous concept to bag of words approach. The idea behind bag of words is to present a text document as a vector that contains the occurrence of words in the text, but if it comes to speech instead of letters and words, the acoustic-phonotactic units such as phone sequences were used. After tokenizing the speech utterance, the phone sequences can be converted by high dimensional phonotactic feature vectors to the n-gram statistic. Finally, the vectorized speech utterance can compared with the vectors that derived

from training data of target languages using cosine similarity and picked the most likely language. (Zissman, 1995)

2.3 Artificial Neural Networks

As in section 2.1 and section 2.2 described there are two main approaches of LID tasks. In this section these approaches with neural networks combined and related researches in the field will be introduced. In many works DNNs are used for LID and SLR tasks and obtained acceptable results in these works. (Richardson et al., 2015) (Lopez-Moreno et al., 2016) (Shivesh Ranjan , Chengzhu Yu , Chunlei Zhang , Finnian Kelly, 2016) (Gonzalez-Dominguez et al., 2014) (Ganapathy et al.)

First of all what kind of features should be used for the input has to be clear, because there are plenty of options of sound feature extraction methods and the influence of features on the results is depending on the task, but mainly for ASR and LID tasks SDC and MFCC, which are dominating the research field, are used. The prediction is made for each of those feature vectors, it means that each frame (i.e 25ms frame) is getting a label from the DNN. After deciding which features will be used for the inputs the essential question to be answered is what kind of neural network should be applied on the extracted features. In the cited works in the first paragraph of this section different type of DNNs are mentioned like Feed-Forward, Recurrent (Long-Short Term Memory) and Convolutional Neural Networks. Especially, RNNs and CNNs has big advantage over Feed-Forward Neural Networks, because of the sequential form of language. (Ganapathy et al.)

On the other hand the configuration is another challenging subtask of DNNs. There are three piece of DNNs: Input layer, hidden layers and output layer. Input layer must have same number of neurons as input vectors. For example in (Gonzalez-Dominguez et al., 2014)] they used 39 dimensional PLP coefficients and the network fed with 21 frames it means input layer has 819 (21 * 39) neurons. Hidden part can contain any desired number of

neurons and consist of multiple layers e.g. 4 hidden layers. The last layer is the output, it must have exactly same number of target classes in our case precise number of target languages. In section 5.3 it will be explained in details.

Choosing best training algorithm could also be tough as configuring the network, but dominating training algorithms for DNNs are gradient descent based algorithms some of these techniques are explained in section 3.2.4 and 5. Very classical training technique is known as Back-Propagation and for RNNs is Back-Propagation Through Time. As mentioned in previous paragraph output layer holds the scores of network.

3 Background

This section describes all the techniques and algorithms used in the experimental part.

3.1 Acoustic Features Extraction

Mel-Frequency Ceptral Coefficients (MFCCs) is a representation of speech signals. Before the sound in the LID system is used, it needs to be represented in an vector space with real numbers instead of sound waves. This preprocessing process has very big impact on LID systems performance. MFCCs has very successful acoustic features for LID tasks. (Shashidhar G. Koolagudi, Deepika Rastogi, 2012) (Leung et al., 2009)

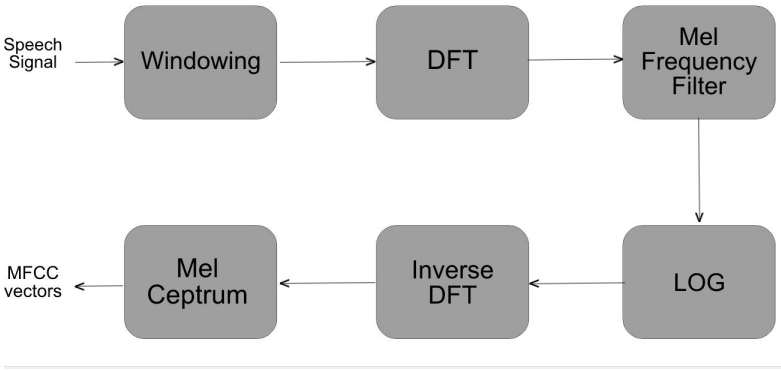


Figure 1: Step by step MFCCs feature extraction.

Figure 1 illustrates the necessary steps to extract MFCCs acoustic features. First of all, the speech needs to be windowed, and then the frequency domain representation is calculated by the Discrete Fourier Transformation (DFT). Next step is applying Mel frequency filter on the signal and then taking the logarithm because of the human non-linear perception of loudness. In the end, inverse DFT is used along with the calculation of ceptral coefficients

to filter out the speaker dependent features of speech signal.(Magre, 2013) (Hasan et al., 2015)

3.2 Machine Learning

Machine learning is a subfield of artificial intelligence. The assumption is that the machines (computers) can infer relations between data from given data samples even if they are not explicitly programmed. These methods can also be named as data-driven predictor or decision maker.

The number of devices with internet connection (smartphones, smart TVs, autonomous cars, smart home gadgets etc.) have drastically increased over the last decade, and as a result, the data being produced every day has been increased as well. With the growing amount of data, researchers have made great human-level progress; perhaps that explains why machine learning has recently been widespread in other research areas e.g. natural language processing, computer vision, genetics and genomics. However, we are not completely able to apply machine learning techniques on every data and in all research areas. Unfortunately, at the moment, these methods are very restricted and can be applied only in very specific tasks such as classification, regression, and clustering. To build a feasible system, there are three crucial questions (below listed) to consider e:

- Is there enough data?
- Does a pattern exist?
- Let the data mathematically formulated?

Now we are going to examine these questions closer and find out what all those mean in scientific sense. First of all, if the relation between data points can be explicitly formulated with a mathematical equation or inequation, then it might be that the machine learning techniques are not the best way

to solve the corresponding task; rule based techniques would be more appropriate. We will be able to solve the problem but not with the best results. Besides, this similar data points must behave similarly thus means the data has a pattern and plays a very important role to infer the relation. In real-life data, there are usually expected noisy data but generalization techniques can handle these data. The last point makes sure that the system has sufficient amount of data in order to be able to analyse it and make inferences out of the data.

3.2.1 Supervised Learning

Machine learning techniques are distinguished from each other by the input data. One of them is called as supervised learning methods given a training data set with the correct label, and the task is to predict labels for new data samples. If the labels are real-value numbers, then the task is a regression task. However, if the labels represent a class or a category, then we apply on the data a classification technique in order to map the input data to the corresponding class. For instance, in the following equation:

$$f(\mathbf{x}) = y \tag{6}$$

where $\boldsymbol{x}$ is the input data in vector form and y is the corresponding label for the $\boldsymbol{x}$, and f is an unknown mapping function which we are trying to find out the best one for. If the output y is a value for instance $y \in \mathbb{R}$ or $y \in \mathbb{N}$ of a continuous set then we should deal with a regression problem. On the other hand if y is a member of a closed set, which means a non-continuous set, e.g $y \in \{red, blue, green\}$ or $y \in \{car, human, animal, couch\}$ then we have to get a handle on the classification problem.

3.2.2 Unsupervised Learning

In supervised techniques, for each data sample an explicit label is given, but in the unsupervised learning, the data clearly differs from the input data.

All we get is only the raw data samples collected for a particular purpose but labels are not included, so the task is to try to predict hidden groups or hidden patterns in the data. Common unsupervised learning tasks are density analyses. Clustering could be solved with techniques such as k-means clustering, hidden markov models, self organizing maps etc. but these kind of tasks and techniques are out of scope for this thesis.

3.2.3 Support Vector Machines

The SVMs is a machine learning method based on a linear discriminant function, which belongs ideally in binary classification problems solutions. On the other hand it could also be applied on regression problems (Drucker et al., 1997). The main idea of SVMs is shown in 2. In the figure below there are two possible SVMs but the algorithm chooses the one on the right side on account of minimizing misclassification of unseen data.

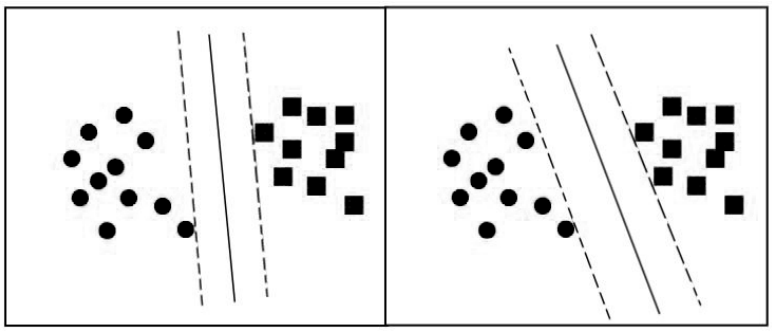


Figure 2: SVM structure. Two possible SVMs with different margin

Consider a given dataset formulated as:

$$D = \{(\mathbf{x_i}, y_i),\ \mathbf{x_i} \in R^p,\ y_i \in \{-1, 1\}\}_{i=1}^{n} \tag{7}$$

which is linearly separable. Lets assume that cirles demonstrates class +1 and squares class -1 in the figure 2. The algorithm aims to find best hyperplane with maximum margin. The hyperplane could be expressed with this formulation

$$\mathbf{w} \cdot \mathbf{x} - b = 0 \tag{8}$$

where $\mathbf{w}$ is weight vector and b is bias and the classification step could be expressed with

$$class(\mathbf{x_i}) = \begin{cases} 1 & \mathbf{w} \cdot \mathbf{x} - b > 0 \\ \text{-1} & \mathbf{w} \cdot \mathbf{x} - b \leq 0 \end{cases} \tag{9}$$

(Fokoue et al., 2013)

Kernel Trick. As it previously mentioned, SVMs can only be successful with classification when the data is linearly separable. If the data is in its feature space (not linearly separable in this case) the solution would be to map the features non-linearly in to a higher dimensional feature space and try to separate there. Whereas mapping the features in higher dimensional space is used in a non-linear kernel, this technique is called as kernel trick. As shown in figure 3 the classification data in two dimensions and not linearly separable through the kernel function ϕ is mapped into three dimensional space and linearly separated in a 2-dimensional hyperplane.

A Kernel is a replacement of an inner product with an appropriate positive definite function, which implicitly performs a non-linear mapping of the input data to a high dimensional feature space. Basically, a kernel function can be expressed like in equation 10, where x, y are n-dimensional inputs vectors. f is a function which maps input vectors from n-dimension to m-dimension space (in this case m is much bigger than n). $< x, y >$ represents the dot product.

$$K(x, y) =< f(x), f(y) > \tag{10}$$

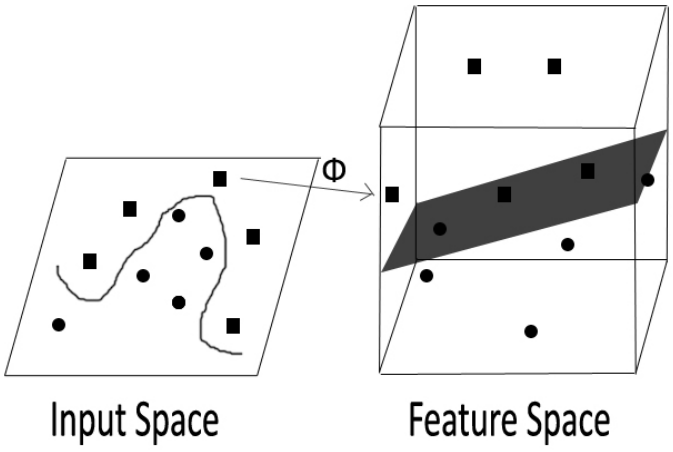


Figure 3: Mapping features in a higher feature space

Two most popular kernel functions are polynomial and radial basis functions. "Polynomial kernel" is shown below in the equation 11

$$K(x, y) = (x \cdot y + 1)^n \tag{11}$$

and Radial basis function (RBF) in equation 12

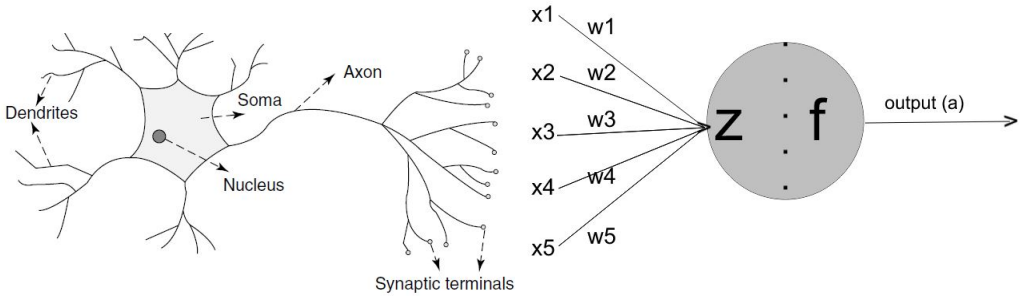
$$K(x, y) = e^{\gamma \|x-y\|^2} \tag{12}$$

3.2.4 Artificial Neural Networks

Artificial neural networks are very successful subfield in machine learning, and artificial intelligence as well. This algorithm is inspired by biological (human and animal) neural networks. A few brief comments on the perceptual brains, cognitive and control functions: The brain consists of billions of interconnected neurons which process through sensory cells perceived signals by using biochemical reactions. As mentioned above, the neurons are connected to each other with nerve fibres and those transmit signals from one

cell to another. If the electrical potential of a cell reaches a threshold, as a result the neuron is fired. Figure 4a is the structure of a mammalian neuron. Dendrites are receiving signals from the other nerve cells which pass to the nucleus. Here, the electrical potential changes by processing the received signal and finally the signal travels along the axon to synapses and synapses distributes it to the other connected neighbour nerves. (Abraham, 2005)

An artificial neural network is a mathematical model of a biological nerve system. Figure 4b iillustrates the simplified structure of a biological neuron. The synapses are represented with weights and the non-linear characteristics of the biological neuron are represented by a non-linear mathematical function known as an activation function e.g. Sigmoid, TanH, ReLU.



(a) Biological neuron (b) Artificial neuron

Figure 4: Illustration of mammal (left) and artificial (right) neuron cells

$$\mathbf{a} = f(z) = f\left(\sum_{j=1}^{n} w_j \cdot x_j\right) \tag{13}$$

where $\mathbf{w}$ is the weight vector, $\mathbf{a}$ is the output vector of a single neuron and f is the activation function. The calculation of z is as follows:

$$z = \mathbf{w}^T \cdot \mathbf{x} = w_1 x_1 + ... + w_n x_n \tag{14}$$

where $\mathbf{x}$ is input variables and $\mathbf{w}^T$ is transpose of $\mathbf{w}$. To calculate $\mathbf{a}$ we have plenty of choices, for instance

$$\mathbf{a} = f(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{15}$$

where σ refers to sigmoid function and also many other non-linear functions can be used instead of sigmoid functions. (Abraham, 2005)

An artificial neural network is illustrated in figure 5. In the previous paragraph we learned the work principle of a single nIn the previous paragraph we learned the work principle of a single neuron, and now we will explain how to build a neural network combining single neurons. An artificial neural network consists of an input layer, a hidden layer and an output layer. Each layer consists of multiple neurons, where directed and weighted connection between those neurons exists. Usually neural networks have one hidden layer, so if the number of hidden layers is more than one, then this type of neural networks is called "deep" neural network. In some cases, ANNs of multiple hidden layers with less number of neurons could give better results than ANNs of one hidden layer with big number of neurons. Further information about designing an sensible ANN is explained in section 5.3.1. (Seide et al., 2011)

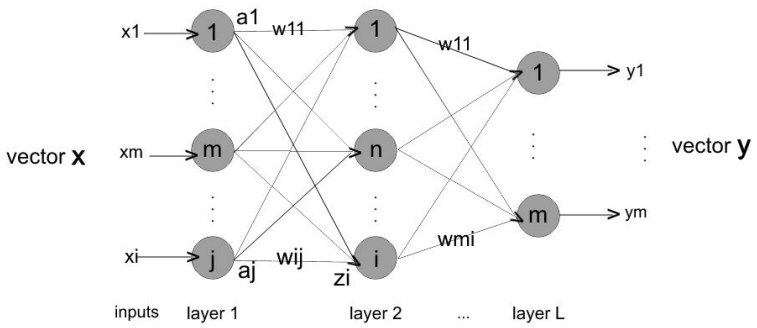


Figure 5: Fully connected multilayered neural network.

According to neural network illustrated in figure 5 computation of the final output vector $\mathbf{y}$ is as follows:

$$\mathbf{y} = f(\mathbf{x}) = \sigma(w^{(L)}...\sigma(w^{(2)}\sigma(w^{(1)}\mathbf{x} + b^{(1)}) + b^{(2)})... + b^{(L)}) \tag{16}$$

where $\mathbf{y}$ is the final output vector of the network, $\mathbf{x}$ is the input vector and σ

in equation 15 is chosen as activation function, w and b are weight matrix and bias and L represents the corresponding layer number. This is the forward propagation of the normalized input data.

$$y_i = \frac{e^{z_i^{(L)}}}{\sum_j e^{z_j^{(L)}}} \tag{17}$$

This is a normalized exponential function and it takes forward passed values m-dimensional respect to figure 5 at the output layer L and maps each prediction to a real value $\in [0, 1]$, where $\sum_{j=1}^{m} y_j$ is equal to 1.

Training an Artificial Neural Network. Data is passed forward through the weighted connections, therefore choosing the best activation function, initializing and tuning hyper-parameters, are very crucial tasks for the artificial neural networks. Usually hyper-parameters are randomly initialized and after each forward propagation they are tuned. Following paragraphs contain more information about the tuning parameters and a couple of algorithms. By tuning parameters, we are aiming to find ideal values for each parameter. Before we tune the parameters we need to identify the deviation of the results, in other words we need to evaluate the performance of ANN. Deviation could be calculated by defining an error function.

$$E(\theta) = \frac{1}{|D|} \sum_{(x,y)} c(f(x), y) = \frac{1}{|D|} \sum_{(x,y)} c(\theta) \tag{18}$$

where θ is parameter set, $f(x)$ is referred to predicted value, so that $f(x) = \hat{y}$ and y is actual value and D is the number of total data samples.

$$c(\theta) = (\hat{y} - y)^2 \tag{19}$$

is a possible function to calculate $c(\theta)$. It calculates the distance between predicted value and actual value. Now the cost function is called mean square error (MSE. There are plenty of options to calculate error, but to demonstrated mean square error is very simple. Training networks are minimizing the cost function which depends on parameters, so the cost function is

minimized by changing parameters. One simple solution of this task would be a brute-force technique, but if the number of parameters increases, the complexity of the problem will increase as well, for that reason brute-force approach is not worth to try with higher number of parameters.

To reduce the computation time we need to avoid redundant calculations. First of all the parameters are randomly initialized and then the gradient is calculated; if $\frac{\partial E}{\partial \theta}$ is negative it means that the cost function tends to downhill and if positive the cost function goes uphill; if equals zero this is the minima for cost function.

Back-Propagation algorithm. At the previous section and previous paragraph we described forward propagation, and evaluation of results, therefore cost function, and estimating the direction of local minima. Now we are going to explain the back-propagation technique that tunes θ to converge to the local minima in a smart way.

Back-propagation is based on gradient descent algorithm. As usual, at first we compute forward propagation and then back-propagate the error from the output layer via hidden layers to input layer. Finally, each of the weights is updated. Lets take a closer look to each step of back-propagation.Forward propagation is defined in equation 16 and finally at the output layer by applying softmax function defined in equation 16, we get final predictions $\hat{y}$. After that step cost function, how well $\hat{y}$ is predicted by the ANN, should be calculated with the equation 18. At this step we figured outerror of the network which should be back-propagated starting from the output layer until it reaches the input layer to minimize the cost function by tuning weights. The derivative of the cost function respect to each parameter, gives us the gradients of each function

$$\frac{\partial E}{\partial w^{(l)}} = \frac{\partial z^{(l)}}{\partial w^{(l)}} \frac{\partial E}{\partial z^{(l)}} \tag{20}$$

where ∂ is partial derivative and l represents the corresponding layer. This formula calculates derivatives of each layer. To understand a bit more we

need to investigate first and second terms both separately.

$$\frac{\partial z^{(l)}}{\partial w^{(l)}} = a^{(l-1)} \tag{21}$$

first term of equation 20 partial derivative of z respect to weights at layer l is equal to activities from the previous layer.

$$\frac{\partial E}{\partial z_i^{(l)}} = \delta^{(l)} = f'(z^{(l)})(w^{(l+1)})^T \delta^{(l+1)} \tag{22}$$

and second term is a recursive function which makes sense because we start to tune weights from last layer to the first. Now, for each layer we have gradients of all the weights in a matrix form. As mentioned before, if the cost function goes uphill which means error is growing, the derivative is positive. The essential movement to do is to subtract the negative value of a derivative from corresponding weight and vice versa if the derivative is negative.

$$w^{(l)} = w^{(l)} - \eta(a^{(l-1)}\delta^{(l)}) \tag{23}$$

where η is learning rate. This step is called updating weights and repeats all the steps until cost function reaches the local minima. In consequence, the training process has been accelerated a lot and saves computation power and time comparison in brute-force technique. (Rojas, 1996)

Limitations of back-propagation. There are some limitations with gradient decent procedures. First of all, if the cost function is a concave function, it means that there are more than one minima. Here, the weight initialization plays a very crucial role because in case of random initialization, the gradient decent algorithm detects only local minima and unfortunately can not guarantee that the local minima is a global minima at the same time.

On the other hand, it could also happen that the random initialization made a great job and landed very close to global minima (basically a "good" local minima), so the expectation is back-propagation of error will converge weights to the global minima but suddenly the local minima is left afterwards

because the gradients have been very high and after-weight updates missed the global minima. Having high initial gradient might miss the "good" local minima, furthermore low initial gradient -eventually zero- might lead the poor generalization.

Nevertheless, choosing the learning rate, affects extremely on the performance of the ANN. Small learning rate could have a bad influence on computation performance, while large learnings rate (as mentioned in the previous paragraph) overstepping global minima. Furthermore, fixed learning rate could result osculation in canyons and endless loop. Further information is given in section 5.3.1.

3.3 Accuracy Estimation

In this section we learned the basic idea of machine learning and we divided the learning tasks in two classes which are supervised and unsupervised learning algorithms. Supervised learning techniques are intended to solve classification and regression problems, where a set of labelled training data is available. Afterwards, in order to solve these problems, we have learned a couple of approaches: Support Vector machines and Artificial Neural Networks. After these steps the performance of built model needs to be evaluated whether the model works well or not. (Refaeilzadeh et al., 2009)

3.3.1 Cross-Validation

Cross Validation is a very common used evaluation method. Basically it splits the training data set into two segments known as validation sets and then the first validation part is used for training the model and the second validation set is used as test data and vice versa, finally the results are averaged.

One of the biggest problems of machine learning algorithms, is fitting too much to the training data. This problem is well known as overfitting problem. Although the model works on trainings data well, the accuracy on the unseen

dataset is very poor. Cross-Validation is good to prevent this type of problem in particular. (Refaeilzadeh et al., 2009)

4 Data and Resources

This section contains information about the data used for researches, data transcription tools, and the acoustic features extracted with. Additionally, it gives information about the Frameworks that we used to implement our baseline system and DNNs.

4.1 Data Corpus

The corpus that used for the research, contains interviews and conversations in multiple languages. All the data has been collected from bilingual Turkish people who were either born or grown up in Germany. There are in total 28 participants (20 females, 8 males), most of them students at the university of Stuttgart, between 19 and 28 years old. Additionally, there are a few younger and older participants also. As it was previously mentioned, the majority of participants are students and the others are from diverse professions such as bankers, school students, social pedagogues etc. Another remarkable fact, is the diversity of their origin in Germany and Turkey as their families originate from various cities and areas of Turkey and all across Germany.

Total length of recorded conversations is 4.5 hours containing Turkish and German languages, although other languages like English, French, Arabic etc. are regarded as other languages. Table 1 gives information about the training, development and the evaluation datasets. We made sure that the sets of speakers of the training set and the test set were disjoint.

	Training	Development	Evaluation
Speaker	24	2	2
Duration	233 min	24 min	23 min

Table 1: Durations for training, development and evaluation datasets

Table 2 gives statistical information about the data (Çetinoglu, 2017). Right column represents number of left column. In the table 2, the number of sentences is equivalent to the number of utterances which we used for training, development, and evaluation.

sentences	3614
tokens	41056
sentence boundaries (SB)	2166
intersentential switches (SCS)	1448
intrasentential switches (WCS)	2113
intra-word switches (§)	257
switches in total	3818
sent. with at least one WCS	1108

Table 2: Basic Information about the data

4.2 Frameworks and Tools

We decided to develop our language identification system using a bunch of tools and frameworks. Praat helped a lot in transcriptions and silence detection in speech. The voice records were handled by OpenSMILE which is used for acoustic features extraction. We chose Python programming language because of its powerful and robust Deep Learning and Machine Learning frameworks such as Theano and Lasagne plus Scikit-learn.

4.2.1 Praat

Praat[1] is a free software for phonetic analysis of speech data and developed by Institute of Phonetic Sciences at University of Amsterdam. It was developed by the Institute of Phonetic Sciences at the University of Amsterdam,

[1]Praat resource: `http://www.fon.hum.uva.nl/praat/`

and it is used in different tasks like speech analysis and synthesis, labelling and segmentation. However, we have used the transcription function rather than other functions. There are different labelling styles. First one is the transcription of an interval (interval tier) i.e. a whole sentence, word, or even a letter, and the other one is the transcription of a time point (point tier). We labelled our speech with interval and point tiers. Code Switching points are marked with point tier and four different labels and other information like speakers, spoken words, and languages which are stored in a multiple interval tiers. See section 5.1.2 for further information.

4.2.2 OpenSMILE

OpenSMILE[2] toolkit is a modular and flexible feature extractor for signal processing and machine learning applications. OpenSMILE can read data from the RIFF-WAVE file format, the audio recorded in. The primary focus is clearly put on audio-signal features. MFCC low-level descriptors can be computed by openSMILE. Further information about feature extraction is given in section 3.1. The Software takes as input a sound file and returns corresponding features in many different formats e.g. Comma Separated Value (.csv), .htk parameter file, .arff WEKA file, .libsvm feature file format, and "Binary float matrix format". However we only use .csv, .arff and .libsvm files; other file formats are irrelevant to our research. (Eyben, 2014)

4.2.3 Scikit-learn

Scikit-learn[3] is a free machine learning library for Python. It includes a large selection of state-of-the-art machine learning algorithms such as support vector machines, k-means, and k-nearest neighbour for supervised and unsupervised problems. The main purpose of the library is: ease of use, good performance on a high-level language, and consistency of the user interface. The

[2]OpenSMILE resouce: `http://audeering.com/technology/opensmile/`

[3]source: `http://scikit-learn.org/stable/`

working principle of scikit-learn is as follows: First we choose the *estimator* and then train the *estimator* using training data with the *fit* method. Once the model has been trained, we evaluate the performance of the estimator by using the score method. The *predict* method which is only with supervised learning estimators available, allows the estimator to predict new data labels. On the other hand. *GridSearchCV* makes possible to tune the parameters to the best working ones. Source code, binaries, and documentation can be downloaded from [4]. (Pedregosa et al., 2012)

4.2.4 Theano and Lasagne

Theano[5] is a general mathematical library for Python. Theano is very powerful for defining, optimizing. and evaluating expressions involving high-level operations on tensors with a GPU support. Besides being a general mathematical library. Theano aims to speed up the deep learning application. (Bergstra et al., 2011)

Lasagne [6] is a Python module built on top of Theano which makes building neural network models very simple. It has been developed as an extension of Theano; basically it inherits Theanos conventions. and functionalities accept and return Theano expressions. In this way, it makes constructing commonly used network structures easy but also allows arbitrary/unconventional models.

Theano is a strongly typed language and it requires that all the associated variables are in the same shape (dimension) and same type. This feature is important for input and output variables. With *theano.function* the network can be built and trained as aimed. (Theano Development Team, 2016). At this point, Lasagne offers a lot of simple solutions. Lasagne has plenty of predefined updates and cost functions. On the other hand, the *lasagne.layers* has a lot of different predefined layer types including input and output layers..

[4]download: `http://scikit-learn.sourceforge.net`

[5]source: `http://deeplearning.net/software/theano/`

[6]source: `https://lasagne.readthedocs.io/en/latest/`

5 Proposed Approaches

This section explains the essential steps for LID tasks. Figure 6 is a block diagram of LID Model. In this section, we firstly explain the data collection and preprocessing process, and then we continue with selecting the best parameters to train models (SVMs and DNNs). In section 6 we evaluate the results which have come out of the LID-systems by using these parameters.

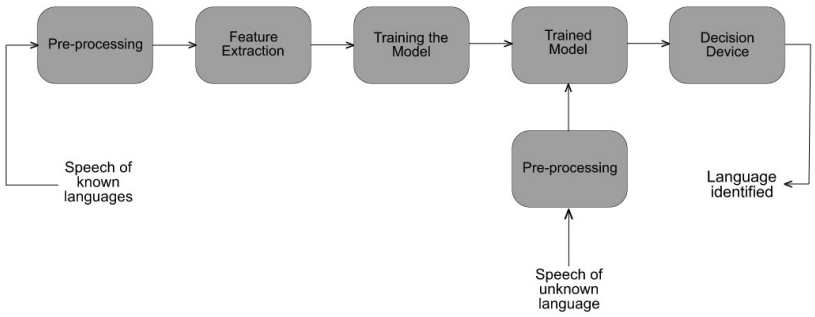


Figure 6: Block diagram of Language Identification Model

5.1 Data Collection and Processing

Data has generally a big impact on the NLP and other machine learning tasks. As mentioned in the previous sections, we have to have enough and well-prepared data to make a satisfying generalization or else we may have a bad generalization. In this section we explained how the data was collected and processed.

5.1.1 Data Collection

Germany is a multicultural country and one of the most popular migration destinations in the world. In Germany, one out of five people is immigrant and with three million population, Turks are the second biggest immigrant group after the Russians (BAMF, 2016). This cultural diversity leads to code-switching conversations as in other multicultural countries like Singapore, USA, Switzerland, China (Lyu et al., 2015). Recent research has shown that speech recognizers outperform for the monolingual speech and data, while on the other hand, monolingual speech recognizers are not able to recognize these code-switched languages. Recognizing code-switched languages makes the problem more challenging and leads the interest of research on how to handle this problem. (Adel et al., 2013)

Hence the data collected for training and test processes, is very crucial in LID tasks, and on this account it has to meet certain standards and quality level. Not only external factors such as background noise, voice overlapping, volume of voice, but also social impact and psychological factors, should be considered in order to reach the desired quality level. In this section we aim to give information about issues we have had on data collection process and interesting observations about speech behaviour which varies extremely under particular conditions.

Foremost, two different scenarios took place, and these scenarios were conducted among two test participants. All scenarios included interviews or were interview alike. In both scenarios, each speaker had a certain role; the first one asked questions the other one answered, and sometimes the other way round. In general, the roles could be named as interviewer and interviewee. Before the experiment started, the participants were asked what they wanted to speak about. They were free to choose either topics that were prepared to talk about or a self suggested topic, which is related to Turkey or Germany. Scenarios briefly described: The topics had been selected so that the speakers had felt encouraged to switch languages as much as possible.

They were told where the speech was going to be used and they were asked for permission.

Before they started we gave them some instructions and advice in order to have all the records uniformed and in clean form. Afterwards they started asking basic questions about the topic and answering the questions subjectively according to their own knowledge and opinion. Finally, a small questionnaire about personal information, consciousness and emotions, was conducted with the interviewees (see appendix 8). We thought that this procedure could give interesting information about demography and statistics. The participants were also requested to give their opinion and suggestions about the data collection process. They replied that if they would not know the intention of the experiment consequently, they would speak more naturally and feel less pressure so the quality of the data could be more satisfactory. In respect of the observations during the data collection, we had the same opinion. If we had not told them the aim of the work, we might have a more natural conversation.

5.1.2 Annotation and Transcription

The data contains three different code-switching types which vary between sentences, words, or a single word, meaning that the first part of a word could be German and the other part Turkish and vice-versa. The first two code-switching type might not sound very interesting because it is very common CS which occurs in many languages like Spanish - English, Mandarin - English etc. However, the last one, CS within a single word, does make German-Turkish CS conversations more interesting and challenging (Lyu et al., 2015). Both languages have different morphological structure, while Turkish is categorized as an agglutinative language (Oflazer, 1994), German is an inflected language (Clahsen, 1999). The agglutinative structure of Turkish makes this interesting language phenomena possible by replacing the stem of a turkish word with a corresponding German word and adding the turkish suffixes.

Thus the transcription and the annotation requires a lot of experience this task is managed by my supervisor Dr. Özlem Çetinoğlu from the beginning to the end. First of all we segmented each conversation by CS point or a new sentence beginning. Each segment contains different information which means verbal and normalized text of each participants speech, language and code-switching points. All this information represented in Praat, is described in 4.2, with six separated tiers. Tiers are named as followed: *spk1 verbal, spk1 normal, spk2 verbal, spk2 normal, lang, codesw.*

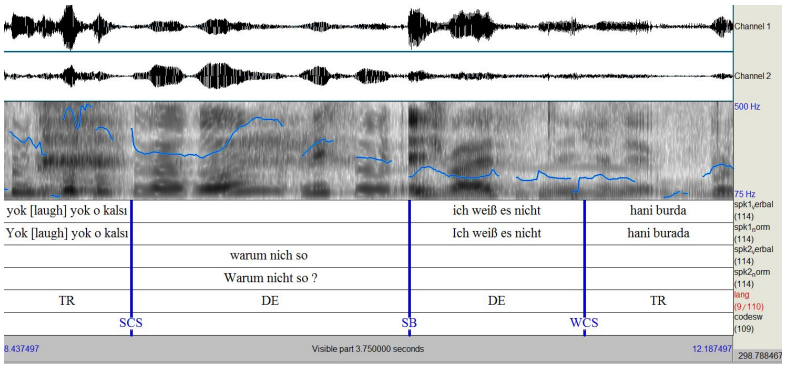


Figure 7: An example of German-Turkish CS utterance segmented with Praat.

Figure 7 represents an example of a 3.75 seconds long CS utterance. A two channel speech signal in two dimensions (horizontal: time and vertical: frequency) is represented on the top of the figure. Spectrogram, a three dimensional representation of sound, (horizontal: time, vertical: frequency, intensity of color: energy distribution) is located right below the 2D signal in the middle. On the bottom the tiers mentioned in the previous paragraph are visible.

Each speaker is transcribed with a dedicated tier (*spk1* and *spk2*). IIncidentally, we transcribed each speakers speech in a verbal form; in other words not grammatically correct as it was spoken out, as well as in a normalized form. As in figure 8, it is obviously represented that written and spoken languages are relatively different from each other (these differences

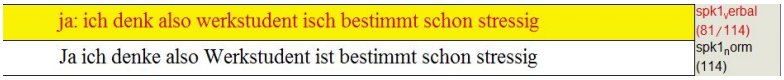


Figure 8: An example for normalized and verbal layer of same speech segment.

are also mentioned in (Tannen, 2016)), however both written and spoken languages are carrying parallel information but in different forms. Normalized text could be more interesting in text analysis task, while the information on verbal tier could be used for example in speech recognition task. The first four tier contain lexical, phonetic and paralinguistic information e.g. noise, silence, cough. The fifth tier contains language information for the segment. Each segment is labelled as one of the following labels: DE, TR, and LANG3. DE stands for German, TR stands for Turkish and LANG3 stands for all the other languages in case that there is any third language. The sixth layer contains the points where the code switching occurs precisely.

SB	sentence boundary (between DE-DE or TR-TR)
SCS	code-switching between sentences (between TR-DE or DE-TR)
WCS	code-switching between words
§	code-switching within a word

Table 3: Code-Switching labels and brief annotation explanations

There are different code-switching labels in table 3 3 briefly described. These labels are described in detail with proper examples to make their usage more understandable. For each label we give one single example which has been taken from the corpus.

$$Ah\ Verlag < \S >' larda < WCS > Okey...$$

This example is explaining in which case §could be possibly used. The word started in German but ended up with Turkish. In this case the word

"Verlag" (in eng. Publishing company) was spoken in German and completed with the Turkish suffix "lar", which in English means the plural suffix "-s".

$$Ja, < WCS > ama < WCS > ich\ muss\ auch\ ehrlich\ sagen...$$

This is an example for the usage of *WCS* label in German-Turkish code-switching corpus transcription (Roughly translated "Yes, but also to be honest..."). The speaker used German grammar rules and German words for the whole sentence except one single word "ama" (in Eng. but). In this case we used WCS label to transcribe this segment.

$$Hast\ du\ auch\ überlegt\ deinem\ Master? < SCS > yok\ yok\ o\ kalsn.$$

A simple example for *SCS* label is given in those sentences. The speaker started to speak in German and right after finishing the sentence he decently switched language to Turkish and spoke the remained part in Turkish.

$$Die\ Stellenangebote\ ...\ hier\ am\ IMS. < SB > Aber\ ich\ glaube\ ...\ haben.$$

We always have job offers from Daimler here at IMS. But you have to be one of the best or you have to know some people there. If the two whole sentences are spoken in the same language, to separate them, we only use as a mark a *SCS* label during the transcription.

5.1.3 Acoustic Feature Extraction

We modified one of the **.conf** files to extract mfcc features from .wav files. This configuration extracts Mel-frequency Cepstral Coeficients from 25 ms audio frames (sampled at a rate of 10 ms) (Hamming window). It computes 12 MFCC (1-12) from 26 Mel-frequency bands and the log-energy is appended

to the MFCC 1-12 instead of the 0-th MFCC, and applies a cepstral filtering filter with a weight parameter of 22. 13 delta, and 13 acceleration coefficients are appended to the MFCC and finaly the features are mean normalised with respect to the full input sequence (usually a turn or sub-turn segment).

SMILExtract -C config/MFCC12_E_D_A_Z.conf -I input.wav -O output.arff

In the above line, a command-line example is given on how the software should be ran. *SMILExtract* command runs the software and this command takes a couple of parameters. Firstly *-C*; this one stands for the path to config file. In the above example the config file is located in the subdirectory *config* as *MFCC12_E_D_A_Z.conf*. Other parameters, *-I* and *-O*, stand for the path of input and output files. As input we feed the system with a *.wav* file and as a result a *.arff* file is obtained.

OpenSMILE returns a vector for each 25 ms frame; the first two components refer to time. Regarding these two components, we assign each vector to a corresponding label which is gathered from the annotated data.

5.1.4 Silence Detection and Merging Frames

In the section we 5.1.3 is explained that each 25 ms frame has a feature vector and in section 5.1.2 explained that each segment has labelled with one of the labels DE, TR or LANG3. HHowever, each segment consists of multiple frames depending on the length of the segment, and naturally between words and sentences short breaks occur, sometimes even for a few seconds. These breaks are leading to ambiguous values for the similar frames, i.e. while silence frames between two German words or sentences are labelled as DE, same silence frames between Turkish words are labelled as TR. Certainly, it is not possible to classify those single ambiguous frames correctly without any context. Because of this reason we detected those silence frames and set a separate label. For the detection, we followed two different approaches; the

first one is energy value extraction with OpenSMILE choosing a threshold value where each frame which has smaller value than threshold is labeled as silence. The command below extracts energy values for each frame with OpenSMILE and saves it in a .csv file.

SMILExtract -C config/Energy.conf -I input.wav -O output.csv

The second approach is silence detection with Praat. Praat offers also a function to detect silence segments. For analogue to manual transcription we get Praats .TextGrid and a .Table files; from those files we can read the silence segments. An example for silence detection can be seen in figure 9.

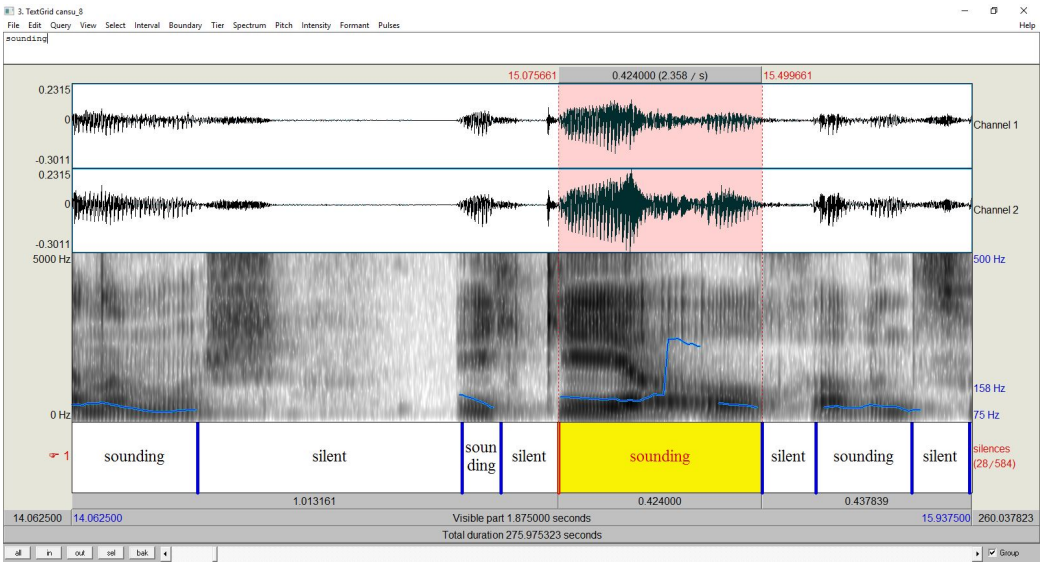


Figure 9: An example .TextGrid file of silence detection with Praat.

After extracting and labelling silence frames we have two different datasets. Since the speech is in the sequential form we thought that merging frames can help to improve the evaluation scores, so we created a new feature vector; we concatenated surrounded n frames to each frame, and created quasi: a n-gram feature vector. Table 4 shows different datasets which we created with praat, opensmile and merging neighbour frames. Prefix represents the tool

silence frames detected with, and suffix represents how many frames a single feature vector contains, i.e. praat-3frames tells us that the silences detected with Praat and three frames, one from the left side, one from the right side, and one in the middle, are merged together. The last column refers to the number of attributes of feature vectors.

Dataset	**Description**	**feature**
praat-1frame	silence detection with praat	40
praat-3frame	silence detection with praat and	120
praat-5frame	silence detection with praat and	200
praat-7frame	silence detection with praat and	280
opensmile-1frame	silence detection with opensmile	40
opensmile-3frame	silence detection with opensmile	120
opensmile-5frame	silence detection with opensmile	200
opensmile-7frame	silence detection with opensmile	280

Table 4: Differently labelled and shaped datasets

5.2 Support Vector Machines

The reduction of errors on the training data and the generalisation on the new data, makes SVMs perform better on small dataset and high-dimensional features. In the section 3.2.3 two different kernel functions were introduced: RBF and polynomial kernel function; there are also other kernel functions like linear or sigmoid functions. For our language discrimination task we decided to use radial basis function kernel because of its non-linear mapping function; it also requires relatively less number of parameters to change. Furthermore, the linear kernel is a special case of RBF, and the sigmoid kernel behaves like a radial basis function for certain parameters (Lin and Lin, 2003).

SVMs are ideally suited for binary classification tasks but the problem we faced in the language identification task has more than two classes. There are

different strategies to classify more than two classes with a binary classifier. One of them is called as One-versus-Rest. It trains one single classifier for each class; in other words there should be N classifiers for N classes, one classifier for each class. $Classifier_c$ calculates how likely sample x is in class c. After calculating probabilities the classifier labels the sample with the most likely class. Equation 24 is a formal description of this process. (Murty).

$$\hat{y} = \underset{c \in C}{argmax} f_c(x) \tag{24}$$

where $\hat{y}$ is the predicted class, c refers to a class, C is a set of classes, x is a sample and f_c is a classifier.

5.2.1 Parameter Selection

There are two parameters to consider at SVMs with Radial Basis Function kernel, error penalty C and γ in equation 12. Parameter C regularizes classification error and maximum margin. On the other hand γ specifies the influence of a single sample data point. In the next paragraph we explained how to tune these parameters parameters in order to get best scores.

To find the optimum parameters we proposed an exhaustive search with a set of random values. This method is also known as grid search. Basically, it applies all the possible combinations of parameter values and then evaluates the scores for each parameter combination in order to get best scores. The Cross-Validation technique is used to measure the performance and produce validation scores. We proposed the following steps to choose the best parameters: First of all we set a reasonable value range for C and γ, $C \in \{0.1, 1, 10, 100, 1000\}$, $\gamma \in \{1, 0.1, 0.01, 0.001, 0.0001\}$ as parameter subset and applied the classifier for each combination of these values on the sample data separately (Chih-Wei Hsu, Chih-Chung Chang and Lin, 2008). Figure 10 shows the accuracies for each (C, γ) pair with a heatmap. Grid-

search was performed on the development dataset of opensmile-1frame (see Table 4).

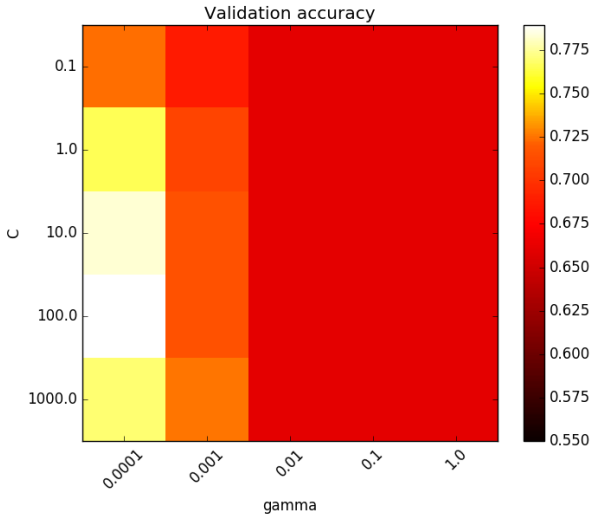
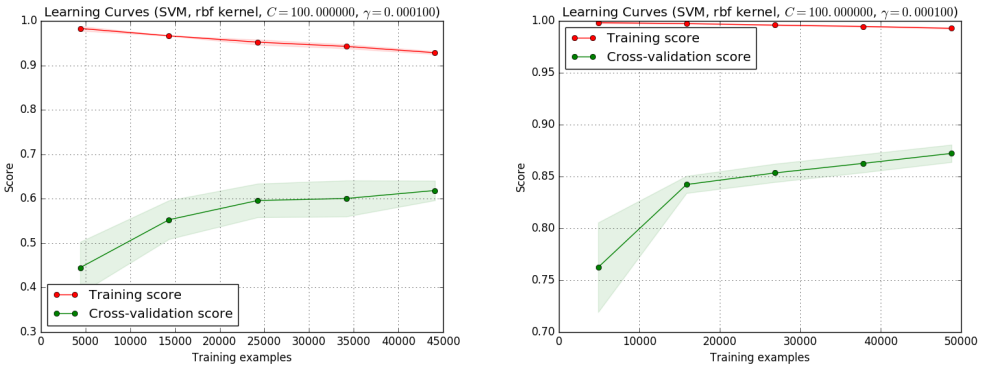


Figure 10: Heatmap of Gridsearch scores

Finally, the results were evaluated with a 10-fold cross-validation technique. In the figure 11 shows the behaviour of cross validation scores over the number of training example. Figures 11a and 11b demonstrates learning curves of the best parameter pair on the development dataset praat-1frame and opensmile-1frame. If the system can not learn on training data of course we can not expect that it will work well on evaluation data.

5.3 Deep Neural Networks

In section 3.2.4 we analysed the components of an typical ANNs. In brief, neural networks consists of an input layer, hidden layers and an output layer. The inputs have passed through the network, layer by layer applying a non-linear activation function and the error was measured with an objective function and the error back-propagated again layer by layer through entire network and parameters were updated with gradient descent algorithm. This section



(a) Learning curve on praat-1frame (b) Learning curve on opnesmile-1frame

Figure 11: Learning curves with best (C, γ) pair on two different datasets.

explains the network architecture, the training strategies and the optimum parameter selection.

5.3.1 Network designing

The network architecture is a main factor in DNNs and has to be carefully decided, because the number of parameters indicates the power of the network and it has a very strong influence on the output. Additionally, for the number of units in the hidden layers there are neither a fixed number nor a rule to calculate optimal number of units exists. It depends on different factors, i.e the number of variables, size of data sets, validation methods, data types, quality of data sets and the nature of your data.

Shape of Input Layer. As it was mentioned before input layer has the same number of neurons with the number of attributes in the feature vectors. We have four different feature vector shapes (see table 4); i.e. data with one frame has 40 attributes, with three frames 120 attributes because each single frame has 40 attributes and we merged three frames, so 40 multiplied by three is 120, this calculationis analogous to the approximation of five and seven frames feature vectors, hereby the number of the input layer neurons

had to be adjusted for each dataset regarding the number of frames so we have four different input layers.

$$\text{shape of input } \epsilon \{40, 120, 200, 280\}$$

Shape of Hidden Layers. There are two crucial tasks to solve for hidden layers. The first task is the decision of the number of hidden layers and the number of units in each layer. According to (Heaton, 2008), ANNs with no hidden layers are only able to classify linear separable data; one hidden layer can classify non-linear separable data, and more than one layer could separate more complex data. We trained our network once with two hidden layers and once with three hidden layers.

The second problem is how many neurons should be in the hidden layers. There is a sanity check for the total number of parameters; the number of parameters should not be greater than the number of the training point, because using too many neurons can result to overfitting, meaning that the outputs is accurate on the training data but on the test data it results in a poor generalization; using few neurons could result to the opposite of overfitting called underfitting. In this case the generalisation will not be sufficient neither on training data nor on test data therefore, we experimented with different number of units in the hidden layers {16, 32, 64, 128, 256}.

Shape of Output Layer. Since we have a classification task in the output layer, we have to have same number of units with the number of total classes. Thus the data contains three different classes (German, Turkish and LANG3) and additionally silence class; the output layer has only four neurons. The difference between units in the output layer and other layers is that the neurons in the output layer has a different activation function, which we mentioned in section 3.2.4 (equation 17). Each neuron contains the probability that the sample in a certain class, so the outputs could be seen as probabilities. This layer is very crucial for training, because the parameters (weights and biases) are tuned regards the output error.

5.3.2 Function Selection

Non-Linear Unit. In section 3.2.4 we showed how to activate the neurons and also explained the traditional activation function sigmoid σ in equation 15. However, according to (Maas and Ng, 2013) the rectify activation function improves the neural network acoustic models on working faster than its alternatives (tanh and sigmoid functions) hence the reason we used rectify activation function as a non-linear function. Equation 25 is the rectified linear function:

$$\varphi(x) = \max(0, x) \tag{25}$$

where x is the input of a neuron, if the input has a positive value then the neuron is activated; otherwise it stays inactive.

Loss Function. In equation 18 we mentioned a cost function to evaluate how far is the output from the actual label and gave an example of objective function in equation 19. Since we have more than one class, same as the output, this function is not suitable to calculate the distance between actual label and output and can lead the network to bad generalization; this was mainly used in regression tasks. In multiclass classification tasks the most common used function is cross entropy function and it is shown in equation 26

$$c(\theta) = -\sum_{i \,\epsilon\, C} y \, log(p_i(x)) \tag{26}$$

where i is a class in C, y is the actual value, $p_c(x)$ is the probability of sample x is in class i.

Training Algorithm. In section 3.2.4 we also explained the ANN Back-propagation training algorithm which has its own challenges by choosing learning rate such as never reaching or overstep the local minima, and the oscillations in canyons. To prevent this problems, we decided to use the

Adaptive Moment Estimation (Adam), one of the adaptive learning rate algorithms.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\delta_t \tag{27}$$

$$\hat{v}_t = \beta_2 \hat{v}_{t-1} + (1 - \beta_2)\delta_t^2 \tag{28}$$

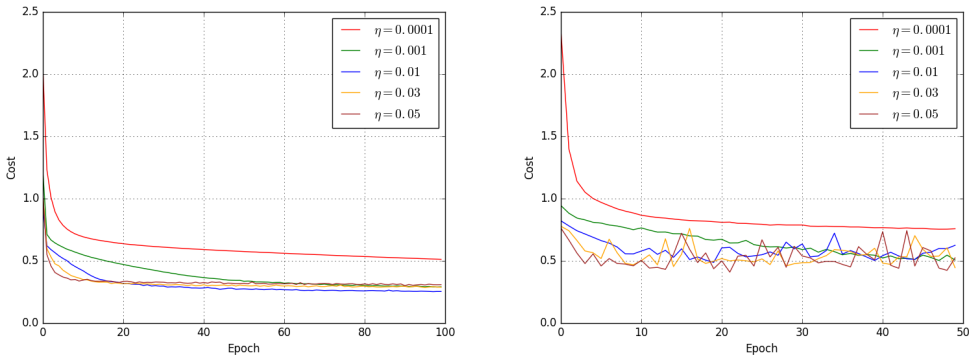
where m_t is the first moment, $\hat{v}_t$ is the second moment and δ is the gradient; according to equations 27 and 28 the update rule is as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}} + \epsilon}\,\hat{m}_t \tag{29}$$

where η is the learning rate and θ the parameters and suggested values of 0.9 for β_1 0.999 for β_2 and 10^{-8} for ϵ. Roughly, this method adapts the learning rates for each parameter from the estimation of the first and second moments of the first order gradients. This method is the state-of-the-art method. (Kingma and Ba, 2015)

5.3.3 Parameter selection

Choosing the learning rate could effect tremendously on performance. In the last paragraph of section 3.2.4 we mentioned some possible problems which could be caused by learning rate; overstepping can occur with high learning rate for instance; on the other hand low learning rate can lead to bad computation time. We tested a set of learning rates on the development data tracked the cost function and in respect to improvement we chose the best learning rate. Figure 12 shows us the change of cost over the epochs for different learning rate on opensmile-1frame data. In figure 12a the red line shows that the cost function is improving very slow and blue, orange and brown lines overfits, but the green line looks well (not too slow or too fast); it corresponds also with validation data in figure 12b.

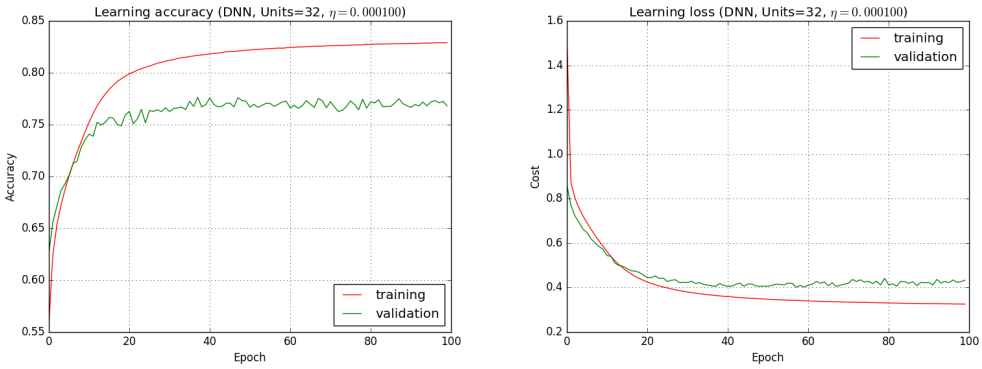


(a) Cost - Epoch function on development set (b) Cost - Epoch function on validation set

Figure 12: Different learning rates cost-epoch function tracking.

On the other hand, we had to decide when we should stop the training. There is a common approach in order to decide the number of epochs, which is called early stopping. Basically, it says: train the network until it starts to overfit and stop at that point. (NIPS 2015 Tutorial)[7]. From the term "Overfitting" arises another question; how can we recognize when the the DNN overfits. The answer is very simple. We have to track the accuracy if the accuracy does not improve, meaning that the network overfits on the training data or the validation cost does not really improve. This method is also prevents overfitting on the training data. Figure 13 shows us the accuracy and cost over the number of epochs. The red lines represent cost and accuracy on the training data and the green line on the validation data. While green line stops to improve the red lines continues to improve after around 40th epoch.

[7]http://www.iro.umontreal.ca/ bengioy/talks/DL-Tutorial-NIPS2015.pdf



(a) Monitoring acuracy

(b) Monitoring cost

Figure 13: Preventing over fitting by monitoring cost and accuracy over time.

6 Results

In this section we represent the performance of SVMs and DNNs, which we proposed in section 5. Finally, we compare the performances of classifiers trained on one of the datasets opensmile-1frame (see section 4.1.)

Tables 5 shows us the total number of frames for each class in opensmile1frame dataset, additionally the number of frames for training, development and evaluation datasets which are produced by splitting opensmile-1frame dataset, and table 6represents the same values for praat-1frame dataset. We did not include any statistics about other concatenated datasets (openmile3frame, openmile-5frame etc.), because all the numbers are almost same. The total number of data in praat-1frame is a little less than total number of data in praat-7frame, because for each utterance first three frames have merged with the fourth frame and last three frames have merged with fourth frame from the end. Since the number of utterances is much smaller than total number of frames we can ignore this difference and assume that the values in the tables are also valid for concatenated datasets. As the tables show that the numbers from tables are completely different, the major difference is in the last column the number of silence frames. While majority of opensmile-1frame dataset consists of silence frames, other classes comprise only a small part of praat-1frame dataset. In contrast to this, the numbers of other classes (German, Turkish and lang3) are in table 6 larger than the numbers in table 5.

	german	turkish	lang3	silence
development	34272	21999	175	87356
evaluation	27112	19806	825	93164
training	289675	266300	2251	821278
total	**351059**	**308105**	**3251**	**1001798**

Table 5: Numbers of frames of each class in opensmile-1frame dataset

	german	turkish	lang3	silence
development	62245	43297	1873	34119
evaluation	63081	38143	286	43089
training	647494	476087	4250	250249
total	**772820**	**557527**	**6409**	**327457**

Table 6: Numbers of frames of each class in praat-1frame dataset

As in section 5.3 described we trained and tested our network with different hidden units and hidden layers. First of all, we tested the performance of the DNN with two hidden layers and different numbers of hidden units and then we monitored the performance of the DNN with three layers on different datasets. Table 7 and 8 show the results. The rows represent datasets and the columns represent differently designed DNNs. In addition to annotation of DNNs, the first part represents the number of hidden layers, i.e. 3L means a DNN with three layers, and second part represents the number of units in each hidden layer, i.e. 128U means that each hidden layer contains 128 hidden units. For training and evaluation purposes, we used disjoint datasets, see table 1, 5 and 6 for further information.

Table 7 shows the performance of the DNNs on opensmile-xframe datasets with different frames. Apparently, concatenating neighbour frames did not help much because majority of the DNNs outperformed on the opensmile-1frame data. On the other hand, DNNs with three hidden layers, 32 and 64 hidden units, have the best overall performance on the opensmile datasets. DNNs with two hidden layers, 32 and 64 hidden units, also indicate high performance among other two hidden layered DNNs. On top of that, another remarkable observation is that DNNs with 128 and 256 hidden units performed worst.

Table 8 shows the performance on the praat-xframe datasets with the same configuration as table 7, but there are huge differences between scores.

	2L 16U	2L 32U	2L 64U	2L 128U	2L 256U	3L 16U	3L 32U	3L 64U	3L 128U	3L 256U
1 frame	82.60	82.80	82.18	82.12	81.80	83.42	82.16	83.75	81.85	81.80
3 frame	81.93	82.58	82.59	81.97	81.62	81.79	82.24	83.16	82.43	82.06
5 frame	80.97	82.24	82.73	82.10	79.40	82.74	82.87	80.59	82.13	80.47
7 frame	80.70	81.73	80.38	80.96	80.98	82.14	82.63	82.03	81.97	81.46

Table 7: Performances of the DNNs on opensmile-xframe datasets

Almost all the best scores of each DNN are reached on praat-7frame dataset and other datasets with multiple frames improved the scores only a little bit. Similarly to the opensmile-xframe datasets, the DNNs with 32 and 64 hidden units performed slightly better than other DNNs, not only with three layers but also with two layers. Obviously, the best score on praat-xframe datasets has been reached with the DNN with two layers and 16 hidden units.

	2L 16U	2L 32U	2L 64U	2L 128U	2L 256U	3L 16U	3L 32U	3L 64U	3L 128U	3L 256U
1 frame	58.69	62.13	59.30	59.73	59.33	58.65	59.79	59.97	59.29	60.50
3 frame	60.67	60.96	61.36	60.82	61.09	59.90	62.17	59.31	60.24	59.59
5 frame	59.88	61.82	61.97	59.80	60.39	61.86	60.31	62.02	61.77	60.10
7 frame	65.24	63.67	61.79	62.53	61.09	63.85	63.03	63.49	61.31	59.94

Table 8: Performances of the DNNs on praat-xframe datasets

Table 9 represents two different scores for each dataset. We took the best performed DNNs on each dataset from table 7 and 8 and tested each of them excluding the silence frames on evaluation dataset, i.e. for praat-7frame dataset we took the DNN with two layers and 16 hidden layers and for opensmile-1frame dataset, the DNN with three layers and 64 hidden units.

The table shows that DNNs could not classify well the frames excluding silence frames. Apparently, on opensmile-xframe datasets only the silence frames are well classified and all the other frames could not be classified as aimed. However, DNNs behaved similarly on praat-xframe datasets even the silence frames excluded, the silence frames have only a light influence on the scores in contrast to the silence frames in opensmile-xframe datasets. In appendix B all correnponding confusion matrices are attached. According to confusion matrices generally class *german* and *silence* are classified relatively well, but class *turkish* and *lang3* could not be classified well.

	w silence frames	w\o silence frames
opensmile-1frame	83.75	50.84
opensmile-3frame	83.16	51.20
opensmile-5frame	82.87	49.68
opensmile-7frame	82.63	49.04
praat-1frame	62.13	54.32
praat-3frame	62.17	55.35
praat-5frame	62.02	57.41
praat-7frame	65.24	59.86

Table 9: Performances of the DNNs on praat datasets

Table 10 shows the best performances of two proposed approaches on opensmile-1frame data. While SVMs could achieve 82.50% accuracy, DNN achieved 83.75%. Due to limitation of time we could not train SVMs to represent the performance measurement on praat-1frame dataset. In figure 11a SVMs converged to 60% on a small part of praat-1frame dataset. Contrary to SVMs, DNN achieved 62.13% on the entire praat-1frame dataset.

	SVMs	DNN
opensmile-1frame	82.50	83.75

Table 10: Comparison of Performances of the DNN and SVMs on opensmile-1frame dataset

7 Conclusions and Future Work

The aim of this thesis was to develop a language identification system with deep neural networks and to support vector machines for German-Turkish Code-Switching language. We concentrated on data collection, transcription and acoustic feature extraction. We experimented with support vector machines and deep neural networks during this thesis, and finally we represented the performances of these two approaches on the collected data. In this section we intend to make some remarkable observations, identify any issues we faced during experiments and propose any suggestions for future researches.

7.1 Conclusions

The most serious issue we faced during this thesis was silence extraction. We developed frame based language identification approach and frame size was only 10ms. The frame size was only 10ms and the data transcription does not contain the word boundaries. The data rather transcribed sentence by sentence and indeed this misled labelling of frames. Taken into account my supervisor Jun. Prof. Ngoc Thang Vu's advice, this issue was solved with two different strategies. As a result the quality of the data was strongly dependent on the silence frames. In section 6 (table 5 and 6) we gave the number of frames in each class and identified there was a great difference between the numbers. On the other hand, we also merged the frames to improve the performances of classifiers, and found the performances either did not improve or only slightly improved.

Another serious issue we faced was training time for SVMs. As notified in the previous paragraph, we discovered a serious issue with silence frames. After discovering this issue, we repeated acoustic features extraction as well as data preparation process and then we started to train SVMs again. Unfortunately, due to a large amount of training of the data and quadratic runtime of SVMs, we could not complete the training for each datasetx (see table 4)

and could not compare all of the performances properly to each other.

In appendices B confusion matrices of the best classifiers are attached and the class *silence* has the best prediction accuracy. Due to lack of samples of *lang3* class the classifiers could not classify this class accuratly. The class *german* has an acceptable accuracy unlike the class *turkish*. The majority of the participants stated that the language proficiency is much higher for German than for Turkish. It could be that most of them pronounced even the Turkish words with German accent.

7.2 Future Work

We implemented a frame-based language identification system. Instead of using acoustic features of each speech frame we can use higher level speech features for instance phonotactic features.We might improve the performance of the language identification system integrating a German and a Turkish phone recognizer as front-end processor to the system and training system with these features. The selection of optimal parameters is also a though task. We executed a grid search with a set of values to choose the optimal parameters of support vector machnines. Additionally, we might obtain better scores through extending the range of gridsearch values and / or decreasing the size of steps, i.e. executing a finer grid search.

We only implemented feed forward neural networks, besides feed forward networks there are other types of deep neural networks such as recurrent neural networks (RNNs), long-short term memory (LSTMs), and convolutional neural network. Thus speech signal is in a sequential form RNNs and LSTMs are ideal to solve classification on such data. Basically, they store values of previous frames in their memory cells use these values to predict current frame (Gonzalez-Dominguez et al., 2014).

References

Ajith Abraham. NEURAL NETWORKS. 2005.

Heike Adel, NT Vu, and T Schultz. Combination of Recurrent Neural Networks and Factored Language Models for Code-Switching Language Modeling. *Acl (2)*, pages 206–211, 2013. URL `http://stage-csl.anthropomatik.kit.edu/downloads/Adel{_}Vu{_}ACL{_}2013.pdf`.

Peter Auer. Main Articles — From codeswitching via language mixing to fused lects : Toward a dynamic typology of bilingual speech *. 3(4):309–332, 1999.

BAMF. Das Bundesamt in Zahlen 2015. Asyl. *Bundesamt für Migration und Flüchtlinge*, Jahr 2015:54, 2016. URL `http://www.bamf.de/SharedDocs/Anlagen/DE/Publikationen/Broschueren/bundesamt-in-zahlen-2015.pdf;jsessionid=DF6E16E43DFFCF33780D72DC91AFDB7E.1{_}cid368?{_}{_}blob=publicationFile`.

James Bergstra, Frédéric Bastien, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, David Warde-Farley, Ian Goodfellow, Arnaud Bergeron, and Yoshua Bengio. Theano: Deep Learning on GPUs with Python. *Journal of Machine Learning Research*, 1:1–48, 2011.

Özlem Çetinoglu. A Code-Switching Corpus of Turkish-German Conversations. 2017.

Chih-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin. A Practical Guide to Support Vector Classification. *BJU international*, 101(1):1396–400, 2008. ISSN 1464-410X. doi: 10.1177/02632760022050997. URL `http://www.csie.ntu.edu.tw/{~}cjlin/papers/guide/guide.pdf`.

H Clahsen. Lexical entries and rules of language: a multidisciplinary study of German inflection. *The Behavioral and brain sciences*, 22:991–1013; discussion 1014–1060, 1999. ISSN 0140-525X. doi: 10.1017/S0140525X99002228.

Harris Drucker, Chris J C Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in Neural Information Processing Dystems*, 1:155–161, 1997. ISSN 10495258. doi: 10.1.1.10.4845. URL `http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf`.

Florian Eyben. *Real-time speech and music classification by large audio feature space extraction*. 2014. ISBN 978-3-319-27298-6. doi: 10.1007/978-3-319-27299-3.

Ernest Fokoue, Zichen Ma, and Ernest Fokoué. Speaker Gender Recognition via MFCCs and SVMs. 2013. URL `http://scholarworks.rit.edu/article/1749`.

Sriram Ganapathy, Kyu Han, Samuel Thomas, Mohamed Omar, Maarten Van Segbroeck, and Shrikanth S Narayanan. Robust Language Identification Using Convolutional Neural Network Features.

Javier Gonzalez-Dominguez, Ignacio Lopez-Moreno, Joaquin Gonzalez-Rodriguez, and Pedro J Moreno. Automatic Language Identification using Long Short-Term Memory Recurrent Neural Networks. *Proceedings of the 15th Annual Conference of the International Speech Communication Association, INTERSPEECH 2014*, (September):2155–2159, 2014.

Rashidul Hasan, Mustafa Jamil, Golam Rabbani, and Saifur Rahman. Speaker Identification Using Mel Frequency Cepstral Coefficients SPEAKER IDENTIFICATION USING MEL FREQUENCY. (September), 2015.

Jeff Heaton. *Introduction to Neural Networks for Java, 2Nd Edition*. Heaton Research, Inc., 2nd edition, 2008. ISBN 1604390085, 9781604390087.

Diederik P Kingma and Jimmy Lei Ba. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. pages 1–15, 2015.

C. C. Leung, R. Tong, B. Ma, and H. Li. A lattice-based phonotactic language recognition system with cmllr adaptation and its implementation issues. In *2009 International Conference on Asian Language Processing*, pages 285–288, Dec 2009. doi: 10.1109/IALP.2009.67.

Haizhou Li, Bin Ma, and Kong Aik Lee. Spoken language recognition: From fundamentals to practice. *Proceedings of the IEEE*, 101(5):1136–1159, 2013. ISSN 00189219. doi: 10.1109/JPROC.2012.2237151.

Ht Lin and Cj Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. *Neural Computation*, (2):1–32, 2003. doi: 10.1.1.14.6709. URL `http://home.caltech.edu/{~}htlin/publication/doc/tanh.pdf`.

Ignacio Lopez-Moreno, Javier Gonzalez-Dominguez, David Martinez, Oldich Plchot, Joaquin Gonzalez-Rodriguez, and Pedro J. Moreno. On the use of deep feedforward neural networks for automatic language identification. *Computer Speech & Language*, 40:46–59, 2016. ISSN 08852308. doi: 10.1016/j.csl.2016.03.001. URL `http://linkinghub.elsevier.com/retrieve/pii/S088523081530036X`.

Dau Cheng Lyu, Tien Ping Tan, Eng Siong Chng, and Haizhou Li. MandarinEnglish code-switching speech corpus in South-East Asia: SEAME. *Language Resources and Evaluation*, 49(3):581–600, 2015. ISSN 15728412. doi: 10.1007/s10579-015-9303-x.

Andrew L Maas and Andrew Y Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. 28, 2013.

Smita B Magre. A Comparative Study on Feature Extraction Techniques in Speech Recognition A comparative study on feature extraction techniques in speech recognition in. (June 2015), 2013.

M N Murty. *SPRINGER BRIEFS IN COMPUTER SCIENCE Support Vector Machines and Perceptrons Learning , Optimization , Classification , and Application to Social Networks*. ISBN 9783319410623.

Kemal Oflazer. Two-level description of turkish morphology. *Literary and Linguistic Computing*, 9(2):137–148, 1994. ISSN 02681145. doi: 10.1093/llc/9.2.137.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2012. ISSN 15324435. doi: 10.1007/s13398-014-0173-7.2. URL `http://dl.acm.org/citation.cfm?id=2078195{%}5Cnhttp://arxiv.org/abs/1201.0490`.

Payam Refaeilzadeh, Lei Tang, and Huan Liu. *Cross-Validation*, pages 532–538. Springer US, Boston, MA, 2009. ISBN 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_565. URL `http://dx.doi.org/10.1007/978-0-387-39940-9_565`.

Fred Richardson, Senior Member, Douglas Reynolds, and Najim Dehak. Deep Neural Network Approaches to Speaker and Language Recognition. *IEEE Signal Processing Letters*, 22(10):1671–1675, 2015. ISSN 1070-9908. doi: 10.1109/LSP.2015.2420092. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7080838`.

R. Rojas. *The Backpropagation Algorithm*. Springer Verlag, Berlin, 1996.

Frank Seide, Gang Li, and Dong Yu. Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. (August):437–440, 2011.

K. Sreenivasa Rao Shashidhar G. Koolagudi, Deepika Rastogi. Identification of Language using Mel-Frequency Cepstral Coefficients (MFCC). *INTER-*

NATIONAL CONFERENCE ON MODELLING OPTIMIZATION AND COMPUTING, 38:3391–3398, 2012. doi: 10.1016/j.proeng.2012.06.392.

John H . L . Hansen Shivesh Ranjan , Chengzhu Yu , Chunlei Zhang , Finnian Kelly. LANGUAGE RECOGNITION USING DEEP NEURAL NETWORKS WITH VERY LIMITED TRAINING DATA. pages 5830–5834, 2016.

Deborah Tannen. The Relation between Written and Spoken Language Author (s): Wallace Chafe and Deborah Tannen Source : Annual Review of Anthropology , Vol . 16 (1987), pp . 383-407 Published by : Annual Reviews Stable URL : http://www.jstor.org/stable/2155877 REFEREN. 16(1987):383–407, 2016.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

Ngoc Thang Vu, Heike Adel, and Tanja Schultz. An Investigation of Code-Switching Attitude Dependent Language Modeling. 2013.

Zissman. Automatic Language Identification of Telephone Speech. *Cis.Hut.Fi*, 8(2):115–144, 1995.

M.a. Zissman. Comparison of four approaches to automatic language identification of telephone speech. *IEEE Transactions on Speech and Audio Processing*, 4(1):31–44, 1996. ISSN 1063-6676. doi: 10.1109/TSA.1996.481450.

Appendices

A Speaker Questionnaire

first name: ...
last name: ...
mother tongue 1: ...
mother tongue 2: ...
foreign language 1: ...
birth place: ...
birth country: ...
age: ...
gender: ...
education: ...
occupation: ...
most used language daily: ...
most used language with family and close friends: ...
proficiency in mother tongue 1 (1-10): ...
proficiency in mother tongue 2 (1-10): ...
mother tongue 1 origin: ...
mother tongue 2 origin: ...

B Confusion Matrices

		Predicted				
		german	turkish	lang3	silence	accuracy
Actual	german	16723	7147	0	3242	61.68%
	turkish	11281	6024	0	2501	30.41%
	lang3	381	309	0	135	0.0%
	silence	81	50	0	93033	99.86%

Table 11: Confusion Matrix: opensmile-1frame evaluation dataset

		Predicted				
		german	turkish	lang3	silence	accuracy
Actual	german	18505	7018	0	1534	68.40%
	turkish	12625	5893	0	1255	29.80%
	lang3	466	282	0	76	0.0%
	silence	285	48	0	92474	99.64%

Table 12: Confusion Matrix: opensmile-3frame evaluation dataset

		Predicted				
		german	turkish	lang3	silence	accuracy
Actual	german	17336	7153	0	2510	64.21%
	turkish	11560	6295	0	1886	31.89%
	lang3	444	279	0	99	0.0%
	silence	41	5	0	92407	99.95%

Table 13: Confusion Matrix: opensmile-5frame evaluation dataset

		Predicted				
		german	turkish	lang3	silence	accuracy
Actual	german	18842	5018	0	3077	69.95%
	turkish	12917	4435	0	2359	22.50%
	lang3	558	150	0	112	0.0%
	silence	36	3	0	92062	99.96%

Table 14: Confusion Matrix: opensmile-7frame evaluation dataset

		Predicted				
		german	turkish	lang3	silence	accuracy
Actual	german	43913	16762	29	2377	69.61%
	turkish	25271	11228	10	1634	29.44%
	lang3	164	119	0	3	0.0%
	silence	7681	681	0	34727	80.59%

Table 15: Confusion Matrix: praat-1frame evaluation dataset

		Predicted				
		german	turkish	lang3	silence	accuracy
Actual	german	43936	17333	0	1720	69.75%
	turkish	24783	12147	0	1126	31.92%
	lang3	177	104	0	3	0.0%
	silence	8786	536	0	33506	78.23%

Table 16: Confusion Matrix: praat-3frame evaluation dataset

		Predicted				
		german	turkish	lang3	silence	accuracy
Actual	german	45385	16281	19	1212	72.16%
	turkish	25509	11536	0	927	30.38%
	lang3	191	90	0	1	0.0%
	silence	9800	642	0	32122	75.47%

Table 17: Confusion Matrix: praat-5frame evaluation dataset

		Predicted				
		german	turkish	lang3	silence	accuracy
Actual	german	56069	4860	0	1876	89.27%
	turkish	32271	4370	0	1247	11.53%
	lang3	268	10	0	2	0.0%
	silence	9386	19	0	32895	77.77%

Table 18: Confusion Matrix: praat-7frame evaluation dataset

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature