Spoken Language Identification with Artificial Neural Network

CS74 2013W

Professor Torresani

Jing Wei Pan, Chuanqi Sun

March 8, 2013

1. Introduction

1.1 Problem Statement

Spoken Language Identification(SLiD) is the problem in which a system takes a speech audio as input and identifies the language used in the speech from a selection of possible world languages. (Figure 1)

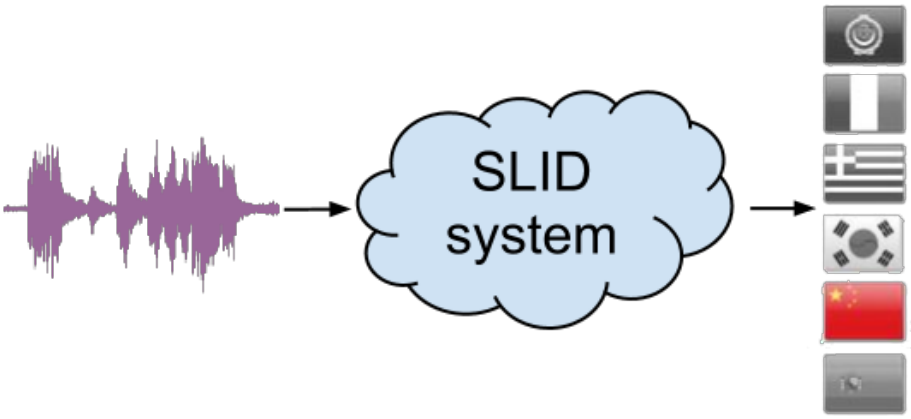

Figure 1 - Problem Statement

1.2 Motivation

SLiD plays a significant role in the preprocessing of speech for further manipulation. Application includes the improvement for natural language processing interface such as Siri and online multilingual voice-based translator such as Google translate, both of which currently require manual selection of the input language. SLiD can also work with human operators. For instance, NSA agents may efficiently identify the language spoken by terrorists with SLiD.

2. Background

2.1 Global Methods

We identify two predominant approaches to SLiD from the literature. Zissman preprocesses the signal with mel-frequency cepstral coefficients (MFCC) and identifies the language with phone recognition followed by language modeling (PRLM) (Figure 2) [1]. This method achieves a 15% error rate in identifying 10-second speeches in English, Spanish, and Japanese. More recently, Montavon improves

error rate to 8.8% by preprocessing the signal with spectrogram and learning language features with time-delay neural network (TDNN) (Figure 3) [2]. Remarkably, Montavon uses 5-second speeches in English, French, and German, which are linguistically less distinguishable than Zissman's dataset.

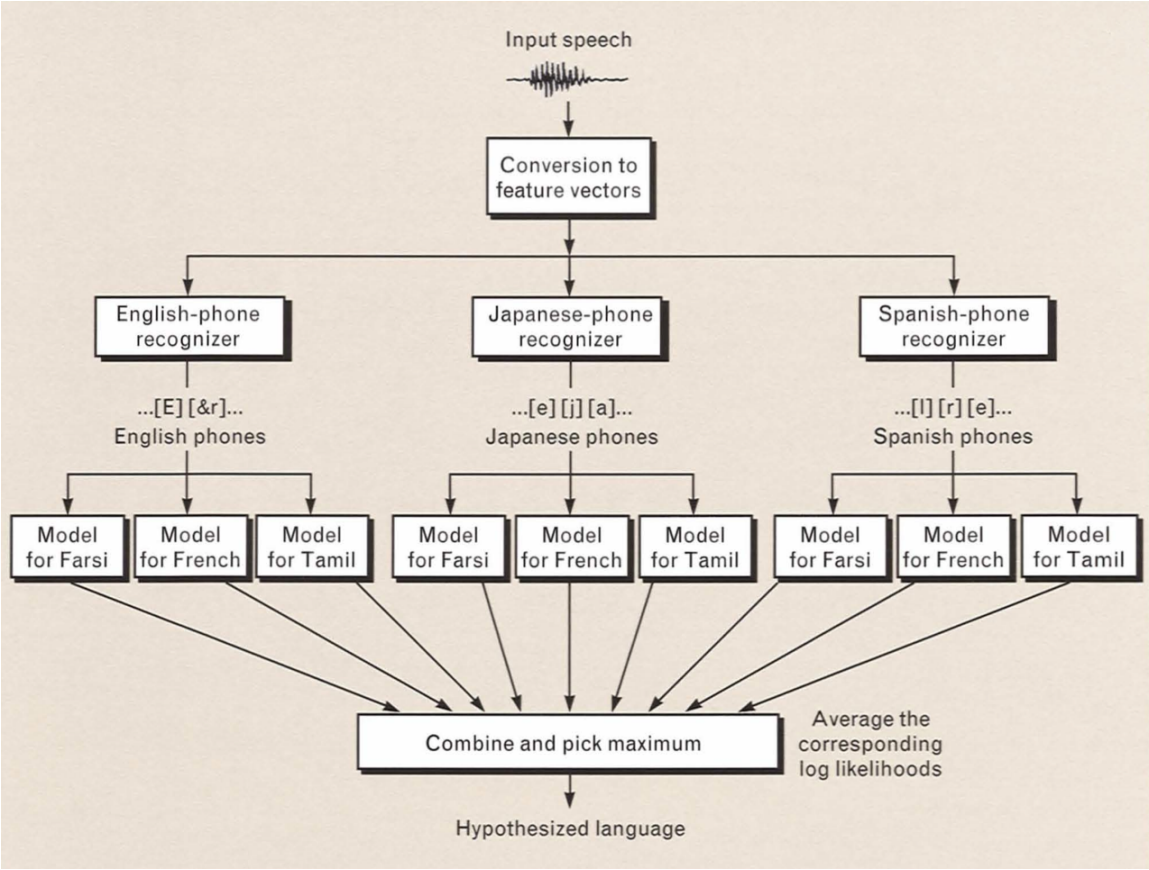

Figure 2 - Zissman's Parallel PRLM Model

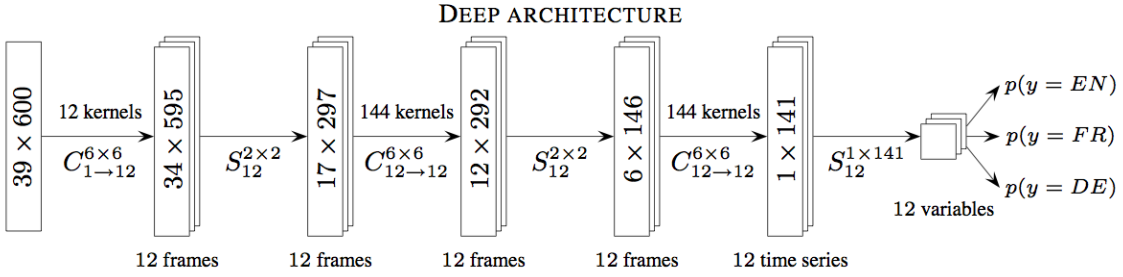

Figure 3 - Architecture for Montavon's TDNN

2.2 Limitations and Our Solution

Limitations exist for the aforementioned methods. The PRLM in Zissman's SLiD system requires prior knowledge of each language in order to extract the phones and build the language models accordingly. This method is not applicable to unfamiliar languages whose phones have not been labeled for training. Montavon's method, though being self-contained, relies on deep learning, which can be computationally expensive. It is not applicable in cases where hardware resources are limited. We resolve the limitations by combining the two methods. By preprocessing signals with MFCC and learning language features with a shallow Artificial Neural Network (ANN), we hope to solve SLiD problem economically without having prior knowledge of the languages.

3. Methodology

3.1 Dataset

Speech examples are collected from VoxForge database. We limit our SLiD system to the identification of English, French, and German. For each language, the collected files are divided into training set and test set. (table 1)

	English	French	German
Training File	800	700	800
Test File	200	150	200
Avg. Length (s)	5.04	5.51	4.68

table 1 - Dataset

3.2 Preprocessing

Our preprocessing resembles Zissman's method (Figure 4) with the following exceptions: (1) Instead of using Speech-activity detection, we remove all silent parts from the original signal, including those within each sentence. Since rhythmic feature is not considered by our system, this silence removal step is justified; (2) While Zissman only computes MFCC and ΔMFCC, we add ΔΔMFCC in hope to capture more dynamic features and to compensate for the lack of rhythmic analysis. Such practice is justified by Hosford [3].

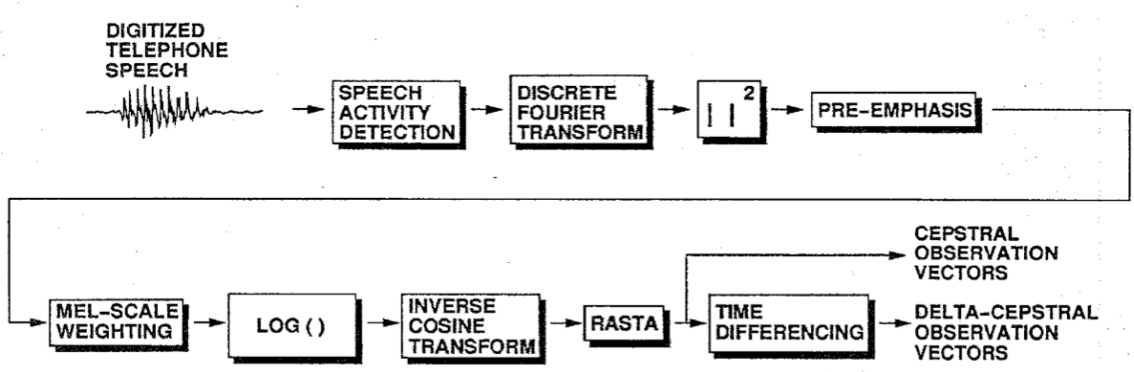

Figure 4 - Zissman's Preprocessing

3.3 The Network

We employ a multilayer feedforward backpropagation network with the cost function shown as below:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

The size of the input layer depends on the number of MFCC generated in preprocessing. The size of the output layer is three, with each neuron corresponding to one possible language. We experiment with the number of hidden layers and the size of each layer.

3.4 Prediction

We make prediction on each file by summing up per frame predictions, and selecting the language with the highest accumulated prediction.

$$\hat{k} = \arg\max_j \sum_{i=1}^{n} \hat{y}_j^{(i)}$$

The longer the input signal, the better the performance we expect.

4. Results and Discussion

4.1 Overall Performance

The best performance is achieved by preprocessing the signals with 40 MFCC, 200 ms window size, 70% overlap, and training the network with 120:80:80:3 topology, 0.5 learning rate, and 0.01 regularization coefficient. We are able to obtain an average accuracy of ~65% for English and German files. The accuracy for French files is significantly lower--the 34% accuracy is no better than random guess. The overall accuracy is 56.73%. (table 2)

		Predicted Label				
		En	Fr	De	Total	Average
Actual Label	En	137 24.90%	8 1.45%	55 10.00%	200 68.50%	56.73%
	Fr	61 11.10%	51 9.27%	38 6.91%	150 34.00%	
	De	55 10%	21 3.82%	124 22.55%	200 62.00%	

table 2 - Performance Summary

4.2 MFCC

The number of MFCC proves to be a deciding factor for performance. Accuracy grows as more MFCC are employed (Figure 5). This makes intuitive sense since additional MFCC extract more information from the raw audio files. We commit to this trend by using 40 MFCC, the largest set we can afford.

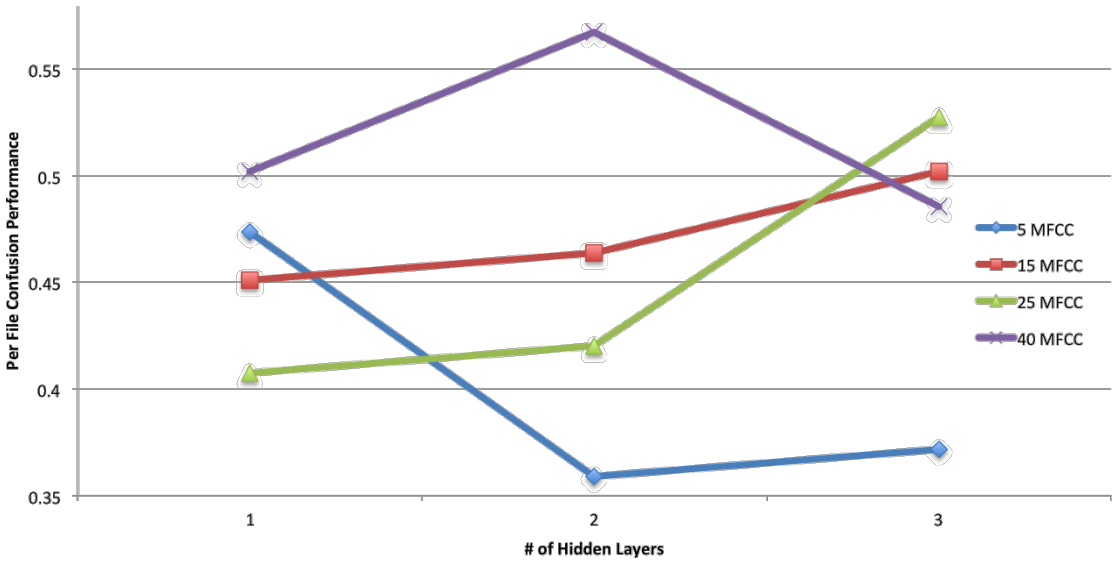

Figure 5 - Performance vs MFCC and Hidden Layers

4.3 Window Size and Overlap

Performance of the ANN model varies across different combinations of window size and overlap between windows. Keeping the number of MFCC constant, we use the same set of audio files to generate a collection of training sets. The performance of the ANN on each of these training sets is plotted, and a ridge is observed between 50 ms windows and 200 ms windows (Figure 7).

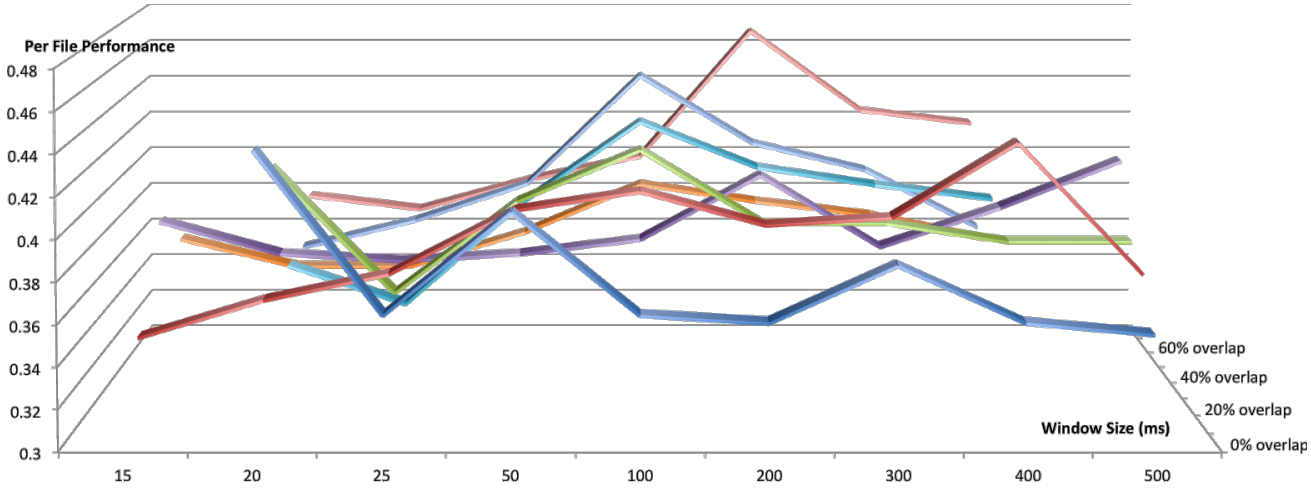

Figure 7 - Performance vs Window Size and Overlap

There exists a correlation between performance and overlap. Across all window sizes, training sets with 20% overlap and 70% overlap produce better models than the rest. Performance is not linearly related to the amount of overlap given the observed deep valley at 50% overlap (Figure 8). Taking the behavior of

both parameters into account, we optimally pick a window size of 200 ms and an overlap of 70%, the combination that has peak performance in Figure 7.

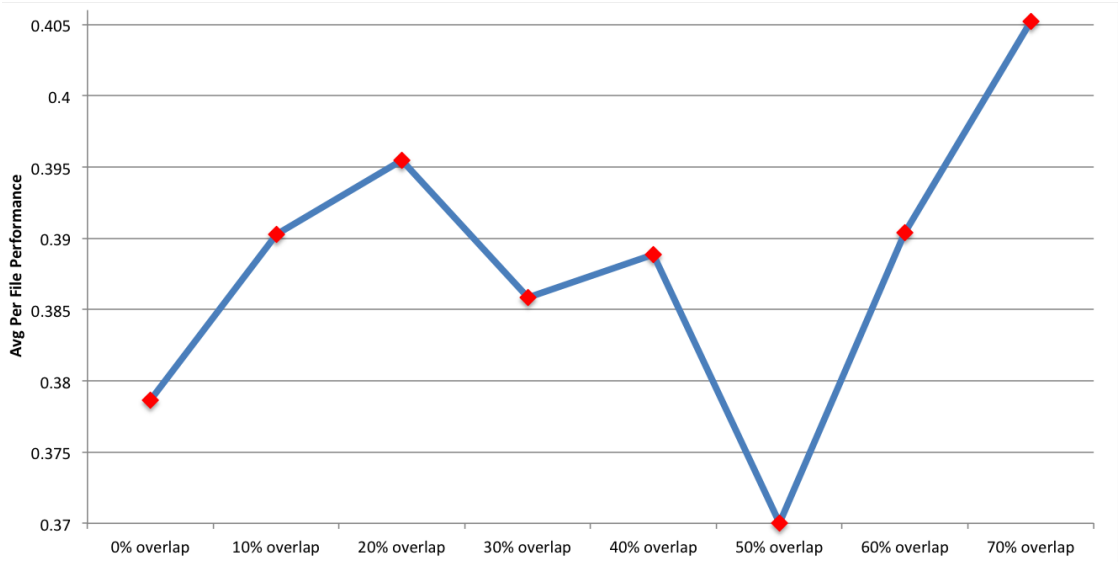

Figure 8 - Performance vs Overlap

4.4 Topology

The result suggests no direct correlation between performance and the number of hidden layers across different MFCC sets(Figure 5), thus we choose the two-layer topology that optimizes performance when the MFCC set size is 40. Using the same method employed to optimize window size and time shift, we find that setting each of the two hidden layers to be two thirds the size of the input layer produces the best performance. (Figure 6)

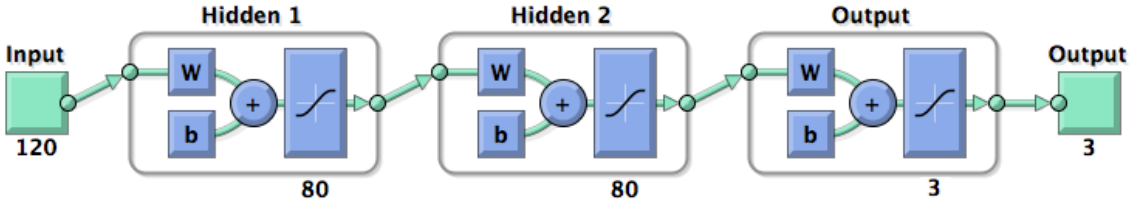

Figure 6 - Network Topology

4.5 Comparison with Support Vector Machine (SVM) and Decision Tree (DT)

In order to understand the relative performance of ANN, we implement SVM and DT to solve SLiD problem given the same preprocessing. Unfortunately, SVM works only after we reduce the data set to 10

examples per language. For larger data sets, SVM crashes the system without returning any result. DT, on the other hand, returns promising results. (Figure 9)

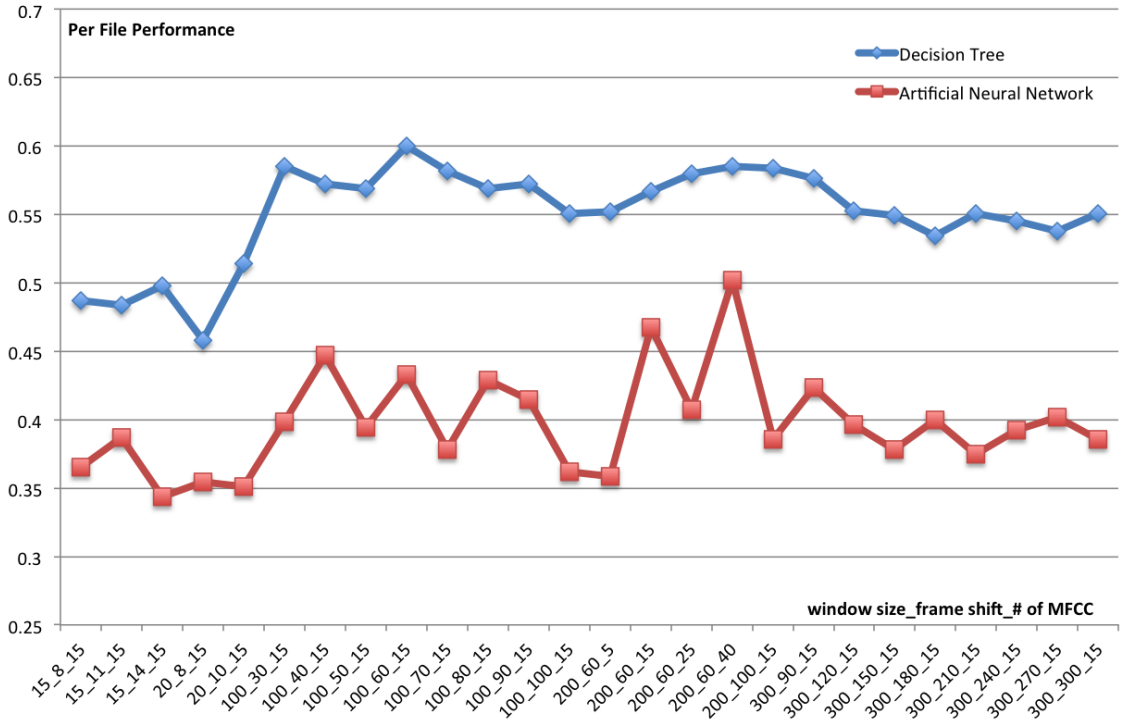

Figure 9 - Comparison between Decision Tree and Artificial Neural Network

With regularization term C=0.01, DT outperforms ANN for every combination of parameters used for preprocessing. Even without much tuning, the average performance of DT is better than the fine-tuned ANN. We contribute the poor performance of ANN to the premature training (see 5.1). Using MATLAB Neural Network Toolbox, we achieve 60.38% accuracy, which is the same as the best performance achieved by DT. Remarkably, DT and ANN suffers from the same problem in which French is confused with the other two languages. The problem is therefore very likely a result of problematic preprocessing.

4.6 Comparison between Per Frame and Per File Prediction

If treating each frame as an example with the same label as the file it comes from, we obtain an average error rate of 63.2%. By making accumulative prediction based on each file, we decrease the error rate to 61.2%. The expected improvement in performance by using accumulative prediction is marginal. Besides, no correlation is found between per frame error rate and per file error rate. (Figure 10)

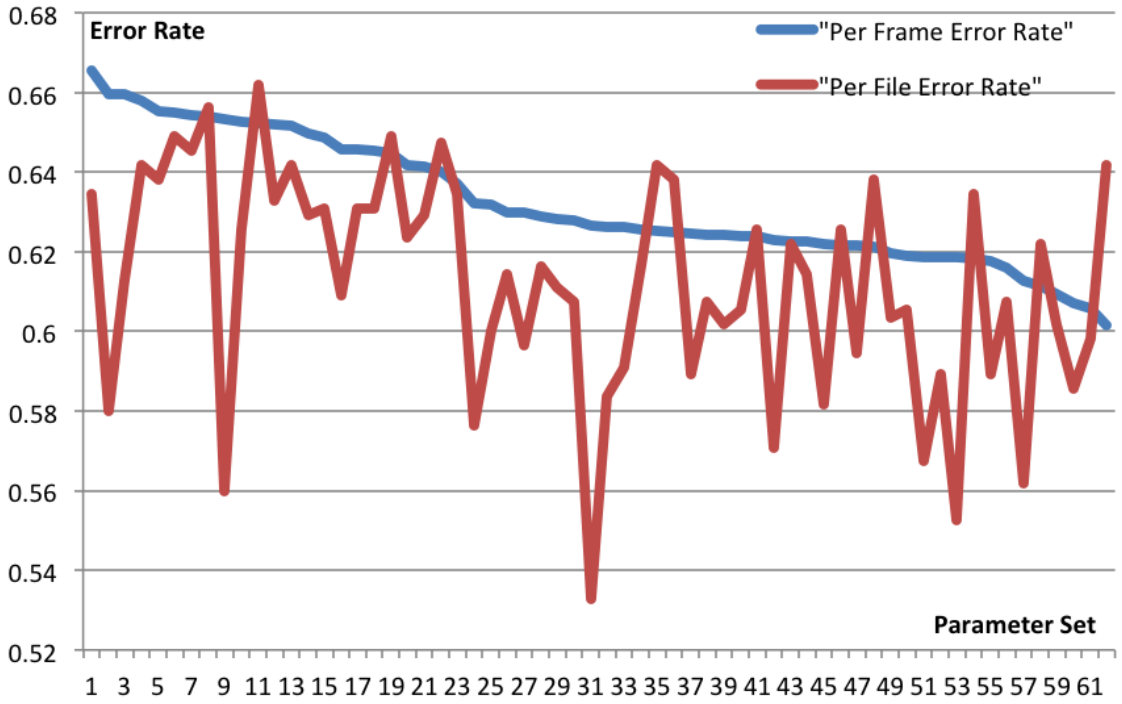

Figure 10 - Comparison between Per Frame and Per File Error Rate

One possible explanation is that MFCC may contain a considerable amount of speaker information along with language information. Since many test examples share speakers with training examples, the prediction may be heavily influenced by the speaker's individual voice features. The accumulated language evidence is therefore marginalized by the accumulated speaker evidence, which varies little between frames within each file.

4.7 Linguistic Perspective

The failure to distinguish French from English and German may seem bizarre from a linguist's point of view. While both English and German are derived from the Germanic language family, French is from the Latin family (Figure 11). The greater linguistic distance between French and the other two languages doesn't result in higher resolution of French for SLiD, but rather more confusion. This implies that the language distances captured by MFCC is largely different from the linguistic distances. Not surprisingly, MFCC captures phones for each language while linguists also considers grammar, rhythm, vocabulary, cadency, etc. Another explanation is the same as in 4.6, in which language identities may have been overridden by speaker identities.

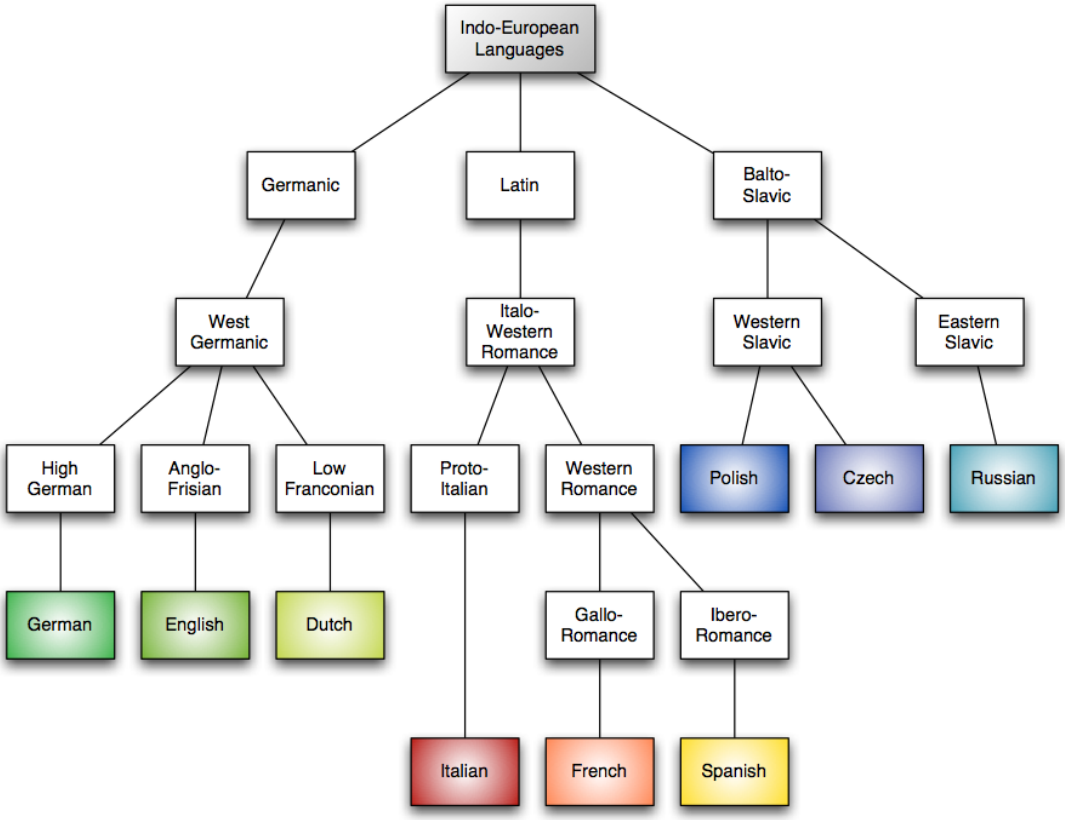

Figure 11 - The Indo-European language tree [4]

5. Future Work

5.1 Learning Rate and Regularization

Tuning our network with low learning rate and loose regularization takes more than 12 hours. To obtain enough data for analysis, we have to terminate training prematurely by enforcing strict regularization and using a high learning rate. Experiments with more combinations of learning rates and regularization terms may yield better performance.

5.2 Pitch Contours and Speech Rhythm

Our SLiD system doesn't utilize the relational information between frames, which might explain the overall

poor performance. We propose that in addition to MFCC, pitch contours and speech rhythm can be used to greatly enhance the performance. Hosford shows a 6% improvement in error rate when adding pitch contour to MFCC-only prediction [3].

6. External Resources

Local Directory: **./slid/mfcc/**
Discription: Computes mel frequency cepstral coefficient (MFCC) features from a given speech signal. The speech signal is first preemphasised using a first order FIR filter with preemphasis coefficient. The preemphasised speech signal is subjected to the short-time Fourier transform analysis with a specified frame duration, frame shift and analysis window function. This is followed by magnitude spectrum computation, followed by filterbank design with M triangular filters uniformly spaced on the mel scale between lower and upper frequency limits. The filterbank is applied to the magnitude spectrum values to produce filterbank energies (FBEs). Log-compressed FBEs are then decorrelated using the discrete cosine transform to produce cepstral coefficients. Final step applies sinusoidal lifter to produce liftered MFCC that closely match those produced by HTK. Demo scripts are included.
Link: http://www.mathworks.com/matlabcentral/fileexchange/32849-htk-mfcc-matlab

Local Directory: **./slid/vad/**
Discription: This is a simple method for silence removal and segmentation of audio streams that contain speech. The method is based in two simple audio features (signal energy and spectral centroid). As long as the feature sequences are extracted, as thresholding approach is applied on those sequence, in order to detect the speech segments.
Link: http://www.mathworks.com/matlabcentral/fileexchange/28826-silence-removal-in-speech-signals

Local Directory: **./slid/voicebox/**
Discription: VOICEBOX is a speech processing toolbox consists of MATLAB routines that are maintained by and mostly written by Mike Brookes, Department of Electrical & Electronic Engineering, Imperial College, Exhibition Road, London SW7 2BT, UK. Several of the routines require MATLAB V6.5 or above and require (normally slight) modification to work with earlier veresions.
Link: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

7. References

[1] Zissman, Marc A. “Comparison of Four Approaches to Automatic Language Identification of Telephone Speech”, 1996.
[2] Montavon, Gregoire. "Deep learning for spoken language identification." *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications,* 2009.
[3] Hosford, Alexander W. “Automatic Language Identification (LiD) Through Machine Learning”, 2011
[4] Ramat, Paolo.*The Indo-European languages*. Routledge, 1998.