# Spoken Language Recognition Using Deep Learning

Brady Arendale[1], Ryan Goodwin[1], Samira Zarandioon[1], and Douglas Reynolds[2]

[1] Master of Science in Data Science, Southern Methodist University, Dallas TX 75275 USA {barendale,rgoodwin,szarandioon}@smu.edu
[2] MIT Lincoln Laboratory, Lexington MA USA dar@ll.mit.edu

**Abstract.** In this paper, we present a methodology to identify language from speech audio and compare recently developed models on open-source data. We take advantage of new techniques including x-vectors and deep learning models. We gathered audio data from various sources containing many spoken languages. We then applied such-and-such transformation to convert the audio into a usable format for deep learning, and used so-and-so method to classify the language spoken. We were able to achieve x% accuracy, performing especially well in such-and-such langauges.

## 1 Introduction

Language comprehension is a major frontier towards achieving general artificial intelligence. New techniques are being developed constantly for both text and spoken langauge comprehension. There are many services commonly available that are capable of both identifying and translating written language from text or image inputs. There are also products such as Google Translate that have the capability to translate speech. However, this generally requires knowing what language one wants to translate. Even applications that can identify a language being spoken generally have only a few of the most spoken languages. A model that could accurately identify many languages would be extremely useful. For example, in call centers language routing is typically done by presenting options that require the caller to press a number. With a spoken language identifier, the caller would only need to say a couple sentences and would be routed automatically to an operator that speaks their language.

The field of language recognition is somewhat limited compared to speech and speaker recognition. Popular corpuses for speaker recognition are typically mostly or only in English. Many datasets that do exist are not freely available, such as those used in the National Institute of Standards and Technology's Language Recognition Evaluation series and those from the Language Data Consortium. However, in recent years there has been an increase in the number of open-source datasets available, including Common Voice from Mozilla [14]. This surge in available data now makes it possible to develop language recognition models without having access to private or expensive datasets.

These developments motivated us to develop a model that could identify a great number of languages from spoken audio only using open-source datasets. Unllike previous works that generally used data from one or two sources, we gathered large amounts of audio data from many sources and compiled them together into one large dataset. This gave us a much broader range of speech for our model to learn the nuances of each language. We then transformed the audio data into a usable form for deep learning using such-and-such method. We built so-and-so model and trained it on our transformed data.

In the next section, we will examine common techniques used in spoken language recognition projects. We will also explore previous projects, what techniques they used, and what results they achieved. We will then explain the methods used for collecting, transforming, and analyzing the data. Next, we will look at our process for the analysis. Then, we will examine the results and see how our model performed. Lastly, we will conclude how the project went and talk about possibilities for future analysis.

## 2 Overview of Common Techniques

### 2.1 Approaches

There are several approaches to spoken language recognition, and projects are typically categorized by the approach they take. The common approaches are acoustic-phonetics, phonotactics, prosodic, and lexical. These approaches attempt to extract features from speech data at varying levels of abstraction. In linguistics, distinct sound events are called phones, and the smallest units that distinguish meaning of one sound from another are called phonemes. The acoustic-phonetic approach is concerned with individual phones and phonemes. Different languages have different sets of phonemes, and the phonemes are used in different frequencies. The phonotactic approach considers phonotactic contstraints, which dictate what phonemes can be used together in a language [1].

The prosodic approach analyzes suprasegmental features such as stress, duration, rhythm, and intonation. For example, many linguists divide languages into three rhythmic classes: stress-timed, syllable-timed, and mora-timed, although this classification is disputed [**?**]. However, prosodic features have been shown to be less effective and more challenging to extract than other approaches. The lexical approach consists of analyzing each language's uinque phonology and syntax for identification. One option is to run each utterance, or period of speech separated by silence, into a semantic understanding model for each language, and choose the one that gives the highest likelihood. However, this is essentially equivalent to learning a whole language to identify it. This is much more complicated and costly than other approaches, so it is not generally used. The acoustic-phonetic and phonotactic approaches have historically delivered the best results with the most efficiency, and are the most commonly used. We will examine the methodologies used for each approach [1].

### 2.2 Phonotactic approach

The phonotactic approach is used because we believe that languages can be characterized by their lexical-phonographical constraints. That is, we expect certain sequences of phones to be more or less common, if present at all, in different languages, and we attempt to classify languages based on these sequences. One of the challenges with this approach is isolating and identifying phones. This is done by a speech tokenizer, which converts an utterance into a sequence of sound tokens, which may be sound frames, phones, syllables, or words. The goal of a tokenizer is to provide the most accurate phonotactic representation possible, greatly influencing the effectiveness of the model itself. One tokenization technique uses Gaussian mixture models (GMMs) at the frame level. GMM tokenization converts speech frames into a sequence Gaussian labels. The drawback to this technique is that the short timeframes fail to capture enough information to be useful. Another tokenization approach is the attribute-based approach, which tokenizes speech into articulatory attributes. These can be made to have a set of attributes common to all languages to build a universal attribute recognizer (UAR). A third technique called phone tokenization uses phone units. These are typically modelled using a hidden Markov model (HMM). Phone tokenizers, or phone recognizers, are often used in state-of-the-art systems due to the tradeoff between development cost and effectiveness. In practice, phone recognizers can be used to model languages that they have not been trained on [1].

The resulting tokenized speech is used as the input to a model. One such model is a phone n-gram model. This model is trained on one of N target languaged for a total of N models. Each model measures the probability of sequences of phones occurring in that language. The output of a phone tokenizer is inputted into each model, returning N likelihood values. We can fuse these likelihood values into one or use them as inputs into another model. We can optionally use multiple phone recognizers and train N phone n-gram models for each. This process is called phone recognition followed by langauge modeling (PRLM), and the process using multiple phone recognizers is called parallel phone recognition followed by language modeling (PPR-LM).

A futher modeling option is to use vector space modeling (VSM) to map the output of a phone n-gram model into a high-dimensional vector. One such technique is the bag-of-sounds framework, analogous to the bag-of-words paradigm in information retrieval and document classification. The bag-of-words maps a document to a vector containing word counts over a known vocabulary, and the bag-of-sounds is much the same but with phones or phone n-gram statistics. We can take the vectorized output and treat it as a vector-based classification problem. Support vector machines (SVMs) are very powerful and efficient for this purpose. We train N one-versus-the-rest SVM models and generate N output scores. Again, we can take this outputs as-is and use the maximum likelihood as our decision, or we can input these into another classification model.

### 2.3 Acoustic-phonetic approach

The acoustic-phonetic approach is based on the idea that languages fundamentally differ in terms of the phones used and the frequency that each phone is used. These approaches attempt to model the acoustic-phonetic distribution of a language using the acoustic features. Instead of identifying and labeling phones as in the previous approach, the acoustic-phonetic approach attempts to extract features directly from the audio. The Mel-frequency cepstral coefficients (MFCCs) are one of the most popular feature extraction techniques for audio data. At a high level, the cepstrum extracts the characteristic features of the audio waveform, and it is transformed to the mel scale, which represents the range of human hearing. MFCCs are computed at short time intervals together with their first and second derivatives. Another popular alternative is the shifted-delta-cepstral (SDC) coefficients. They contain information about not only the time frame they are calculated for but also for a number of adjacent time frames [1].

A common model used after feature extraction is the universal-background-model-based GMM (GMM-UBM), popular in speaker recognition. A universal background model (UBM) is first trained on the entire dataset containing all langauges, and represents the general characteristics of speech among all languages in the data. Separate GMMs for each language are then trained using the maximum a posteriori (MAP) technique, with the UBM as the prior distribution. The GMM is used for its ability to approximate arbitrary distributions and to model the underlying components with individual Gaussian distributions. We can approximate the acoustic-phonetic distributions of a spoken language using this method. However, GMMs are susceptible to acoustic variability caused by non-language characteristics such as speaker and noise.

As with the phonotactic approach, we can use vector space modeling to map spectral features of an utterance into a high-dimensional vector. We can again use SVM methods to classify these vectors.

### 2.4 Deep learning approaches

A productive platform for understanding speech recognition is Neural Networks emerged as a model in ASR (Automatic Speech Recognition). Neural networks have been used at spoken language recognition in classification, isolated word recognition, and audiovisual speech recognition. Neural networks make small assumptions about the statistical qualities of a great model for spoken language identification. Also tremendous progress in spoken language recognition from technology in spoken language recognition, which benefited from technological breakthroughs in related areas, such as signal processing, pattern recognition, cognitive science, and machine learning [3].

Initially, Feed-Forward Deep Neural Networks (DNNs) can be utilized but only model by working with a limited and fixed size sliding window of acoustic frames. This precludes the models from utilizing speech dependencies on other part of the speech input. To address these concerns, Recurrent Neural Networks

(RNNs) are used. RNNs utilize looped connections that allow for connections between different parts of speech by allowing previous inputs or time steps to influence calculations for subsequent time steps by retaining information within internal states.

Nevertheless, RNNs still have issues with retaining memory which can influence time steps farther away from each other. The next evolution is to include a mechanism that retains memory to influence multiple parts of a speech vector using LSTMs (Long Short Term Memory). LSTMs are modeled from RNNs but include special memory blocks within each hidden recurrent layer. Additional improvements to LSTMs include BLSTMs (Bidirectional LSTMs) which operate by using input from both directions of a speech excerpt in order to calculate decisions on a current input.

One of the major advances in recent years is the development of x-vectors as a new way of extracting embeddings of utterances. A deep neural network is trained using extracted audio features as inputs and language as the target. Outputs are extracted from one of the last hidden layers and used as embeddings [9]. X-vectors extracted in this way have been shown to perform better than i-vectors extracted from GMM-UBM models in both language recognition and speaker recognition tasks [10].

Another advantage to the shift to deep learning frameworks is the types of feature extraction available. In addition to MFCCs, deep learning approaches can also use filterbanks [10] and bottleneck features [9]. Filterbanks are calculated during the derivation of MFCCs, but coefficients are highly correlated, which poses a problem for many earlier models. To account for this, the discrete cosine transformation (DCT) is applied, resulting in uncorrelated features. However, deep learning models are able to handle the correlated coefficients from filterbanks, resulting in potential gains by minimizing loss of information from the DCT[3]. Bottleneck features (BNFs), unlike MFCCs and filterbanks, are extracted from neural networks. A deep neural network is created with one "narrow" layer with relatively few neurons in between several "wide" layers with many more neurons. This creates a "bottleneck" where the information needed to predict the target from the input is assumed to be compressed. We can then take the weights from the bottleneck layer and use them as features in an x-vector network or other model [11].

## 3 Tutorial: Mel-frequency cepstral coefficients and filterbanks

Audio data presents many unique challenges. Such challenges include background noise, accents, variation in tone and potentially limitless vocabulary that can be spoken. To mitigate these obstacles, a variety of techniques have been researched and must be used to get the data in a format that can be analyzed. Each step

[3] https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html. Last accessed May 22, 2020.

of the speech recognition system must deal with its respective challenges before the next step can be approached. Because of this, the entire process as a whole must be broken down into steps, each step must have its challenges laid out, and each challenge must have a solution in place to allow for the ability to move to the next step in the spoken language classification process.

The most primitive challenge is first getting the recorded audio in a format that can be interpreted by a computer for analysis. This is known as Audio to Digital Conversion and can generally done by any modern computer automatically when inputting audio signal into an instrument such as a microphone that is directly connected with the computer.

After getting the data in a computer-useable format, background noise is the first challenge faced when dealing with spoken language data. Before the speech in the audio can be interpreted, the background noise must be filtered out allowing a model to focus solely on the intended sound. Although microphones and other speech recording systems are constantly being upgraded to reduce background noise and pick up only intended audio, background noise continues to be present in almost every scenario. The process of extracting actual speech from background noise is known as Voice Activity Detection (VAD). This is a necessary step in any speech recognition system as the background noise can heavily affect a models interpretation of audio data. Most VAD methods exist for the primary reason of feature extraction [4]. Numerous methods of VAD have been researched, and our analysis works with a method known as Mel-Frequency Cepstral Coefficients, or MFCC. The idea behind MFCCs is to break the audio down into very short intervals that can be considered stationary. Figure 1 below [5] simulates a high-level overview of MFCC generation.

Audio signals are constantly changing, and it is very difficult to evaluate data that is non-stationary, therefore we need to break the audio down into timeframes in which we can view the data as statistically stationary. This timeframe is generally determined to be within 5-100 milliseconds when working with audio data. After about 0.2 seconds, signals are non-stationary and begin to reflect more holistic speech sounds that can be too complex for current machine learning techniques [7].

MFCCs gained popularity in the early days of speech and language recognition because they were easy to calculate and uncorrelated, making them ideal for the popular models in use at the time, including GMMs and HMMs. With recent advances in deep learning, however, there is not as much of a need for uncorrelated features. During the calculation of MFCCs, we calculate filterbanks, which are highly correlated representations of the audio data. We then apply the discrete cosine transformation (DCT) to decorrelate the fliterbank features before mean normalizing the data, which results in our MFCCs. Since we now have models that are able to make use of correlated features, we can exclude the DCT and mean normalize the filterbanks directly. Since the DCT is a linear transformation, it can remove some of the nonlinearities in the audio data, which can cause loss of information. This can lead to both performance benefits and faster extraction of audio features since filterbanks take one less step to

compute. For these reasons, filterbanks have increased in popularity as audio features with the domination of deep learning models in the fields of speech and language recognition.

Once we have the data in a form that can be input to a model, more challenges arise. Another common challenge faced with audio data is the large variance of speech patterns that can be used to create the same sounds. A human will recognize the words Happy Birthday whether they are coming from an American male speaking with a Boston accent or a woman from London speaking with a heavy British accent. For a computer this is more difficult. The words may be the same, but the sound waves that the model is taking as input are not identical. Breaking this down in more detail, in general speakers repeating the same words are distinguished by two features, acoustic differences in the voice and pronunciation [6].

Building on the challenge listed in the paragraph above, once we have dealt with acoustic differences and accents by normalizing the data, we have an almost infinitely large set of words that can be input to the model. It is impossible to have every possible word in a database to map to a language. Even if we had a languages entire dictionary, slang words will arise that are not captured by the official documentation of the language. In this case, we must rely on the system to capture the context the slang is in and be able to classify the language while disregarding its inability to identify the slang phrase. In theory, the system should be able to have enough information around the slang to hypothetically toss out the phrase it doesnt understand and still perform well enough to classify the language.

Once we have broken down the speech far enough to convert it to text, we must determine which language that text is written in, thus completing our goal of classifying the spoken language. There have been numerous studies in creating lexicon-free speech recognition systems, but the bulk of spoken language recognition systems use lexicons to map the text to a specific language. Thus introduces the final challenge in our process of building a large enough lexicon for each language to map the text to and classify. As with the previous challenge, it is nearly impossible to encapsulate every possible word in a language. Therefore we must adapt our lexicon to optimally map the text. One such method of doing this is taking both an official documentation for each language and creating a dictionary database to map to, but a more robust method would be to take commonly spoken phrases from the language and include that in addition to our dictionary. This will allow us to differentiate between languages that use similar word structures because it will add a second pass classifying phrases associated with a language rather than individual words [8]. Hence if a group of words can be attributed to multiple languages, the further breakdown of the pattern of the words can, in theory, further break down the classification among the possible candidates.

## 4 Tutorial: X-vectors

X-vectors were developed for speaker recognition with the goal of producing embeddings that generalize to speakers not seen in the training data [12]. X-vector embeddings are extracted from a deep neural network trained on the extracted audio features. The architecture is shown in Table 1 [9].

**Table 1.** Architecture of DNN for extracting x-vectors

| Layer | Layer context | Total context | Weight dimensions |
|---|---|---|---|
| frame1 | $[t-2, t+2]$ | 5 | $5F \times 512$ |
| frame2 | $\{t-2, t, t+2\}$ | 9 | $1536 \times 512$ |
| frame3 | $\{t-3, t, t+3\}$ | 15 | $1536 \times 512$ |
| frame4 | $\{t\}$ | 15 | $512 \times 512$ |
| frame5 | $\{t\}$ | 15 | $512 \times 1500$ |
| stats pooling | $[0, T)$ | $T$ | $1500T \times 3000$ |
| segment6 | $\{0\}$ | $T$ | $3000 \times 512$ |
| segment7 | $\{0\}$ | $T$ | $512 \times 512$ |
| softmax | $\{0\}$ | $T$ | $512 \times L$ |

The input to the DNN is a segment consisting of $T$ speech frames. Each speech frame $t$ has $F$ features, which can be MFCCs, filterbanks, or BNFs [9, 10]. The first five layers operate on the frame level, then they are pooled and operated on as a whole segment. Each speech frame $t$ is input into the first layer along with the previous two frames and next two frames, for a context of five frames for each $t$. The total number of features inputted per frame is thus $5F$. The first layer (usually) expands the input into 512 dimensions. A ReLU activation is applied, and this is also the activation function for every other layer except the softmax layer. The frame1 layer outputs for $t-2$, $t$, and $t+2$ are spliced together to form a $512 \times 3 = 1536$-dimensional vector with a total context of nine frames. A similar process is used for the frame3 layer, taking the $t-3$, $t$, and $t+3$ outputs from frame2, for a total context of fifteen frames. The frame4 layer adds no additional context. The frame5 layer expands the output into a 1500-dimensional vector.

Next, a statistical pooling layer takes each 1500-dimensional vector from all $T$ speech frames and calculates the mean and standard deviation of each dimension, resulting in a 3000-dimensional output representing the entire segment. The segment6 layer takes this vector and reduces it to a 512-dimensional linear combination, representing the full context $T$. Another segment layer calculates nonlinearities in the data, which are finally used to predict one of $L$ languages after a softmax activation.

The outputs of either segment6 or segment7 can be extracted pre-activation and used as embeddings, but segment6 embeddings were found to be preferable

and came to be known as x-vectors [10, 12]. X-vector-based classification models have been found to outperform state-of-the-art i-vector models in a wide variety of applications [13].

X-vectors can be augmented with various types of background noise, such as babble, music, noise, and reverberation, to artificially increase the amount and diversity of training data [10]. This has been shown to significantly improve performance in certain tasks including speaker recognition and language recognition [9, 13].

## 5 Methodology

We began by gathering data from several open-source datasets. We then loaded the data into a Python script and extracted MFCCs and filterbanks. We split the data into training, test, and validation sets, making sure there was no speaker overlap and balancing for sex as much as possible. We trained x-vectors on our training set and used the training weights to find embeddings for our validation and test sets. We selected a few different models to compare for classification based on recent research. We then tuned and trained each model minimizing multiclass crossentropy loss and compared them based on equal error rate (EER).

### 5.1 Data collection

We collected data from a variety of open-source multilingual datasets. Datasets included Common Voice, Omniglot, RhinoSpike, VoxForge, and such-and-such dataset. Table 2 shows the datasets used, the number of hours of recordings, and the number of languages.

**Table 2.** Datasets used

| Dataset | Hours of recordings | Number of languages |
|---|---|---|
| Common Voice | x | 40 |
| Omniglot | y | 122 |
| RhinoSpike | z | 86 |
| VoxForge | a | 19 |
| Such-and-such | b | c |

### 5.2 Model Selection

A number of effective speaker recognition and language recognition algorithms have been developed in recent years. We decided to compare a few different models to determine which one worked best on our open-source data. The models we used were PLDA, NPLDA, CNN, LSTM, and such-and-such.

### 5.3 Data Preparation and Model Usage

In order to use audio data in our models, we needed to convert it from audio formats such as MP3 or WAV to numerical representations. We used VAD to separate the speech frames from the non-speech frames. We extracted both MFCCs and filterbanks to compare their performance. We then split our data into training, validation, and test sets. In order to prevent data leakage, we made sure that there was no speaker overlap between these sets as much as possible. We also aimed for an even balance of sexes between each set.

We trained an x-vector network, as described earlier, on our training set. We then use our trained weights to calculate x-vectors for validation and test sets as well. We used our x-vector embeddings as features in several different models, minimizing the categorical crossentropy loss. We used equal error rate (EER) as our evaluation metric. EER is defined as the point where miss rate and false alarm rate are equal and is a standard metric in evaluating language recognition models. Our results are shown in Table 3.

**Table 3.** Model comparison

| Model | EER |
|---|---|
| PLDA | x |
| CNN | y |
| LSTM | z |

## 6 Conclusion

We found that such-and-such model performed the best on our open-source data. We achieved a so-and-so EER, which compares such-and-such way to recent language recognition projects.

## 7 References


1. H. Li, B. Ma and K. A. Lee, "Spoken Language Recognition: From Fundamentals to Practice," in Proceedings of the IEEE, vol. 101, no. 5, pp. 1136-1159, May 2013.
2. Arvaniti, Amalia & Ross, Tristie. (2010). Rhythm classes and speech perception.
3. Sak, H. & Senior, Andrew & Beaufays, F.. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. 338-342.

4. Drugman, T, Stylianou, Y, Kida, Y & Akamime, M, Voice Activity Detection: Merging Source and Filter-Based Information, IEEE Signal Processing Letters, vol. 23, no. 2, pp. 252256, 2015.
5. Logan B., Mel Frequency Cepstral Coefficients for Music Modeling, Proceeding of the International Symposium on Music Information Retrieval (ISMIR) 2000.
6. M. R. Hasan, M. Jamil, M. G. Rabbani, and M. S. Rahman, "Speaker identification using Mel frequency cepstral coefficients," 3rd international Conference on Electrical & Computer Engineering ICECE 2004, 28-30 December 2004.
7. Kardava, I, Antidze, J & Gulua N, Solving the Problem of the Accents for Speech Recognition Systems, International Journal of Signal Processing Systems. 4. 235-238, 2015.
8. Dahan, J.G., Gupta, V, Method and apparatus for performing speech recognition utilizing a supplementary lexicon of frequently used orthographies, US 6018708A, United States Patent and Trademark Office, January 25th, 2000.
9. Snyder, D., Garcia-Romero, D., McCree, A., Sell, G., Povey, D., & Khudanpur, S. (2018, June). Spoken Language Recognition using X-vectors. In Odyssey (pp. 105-111).
10. Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., & Khudanpur, S. (2018, April). X-vectors: Robust dnn embeddings for speaker recognition. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 5329-5333). IEEE.
11. Matejka, P., Zhang, L., Ng, T., Glembek, O., Ma, J. Z., Zhang, B., & Mallidi, S. H. (2014, June). Neural Network Bottleneck Features for Language Identification. In Odyssey.
12. Snyder, D., Garcia-Romero, D., Povey, D., & Khudanpur, S. (2017, August). Deep Neural Network Embeddings for Text-Independent Speaker Verification. In Interspeech (pp. 999-1003).
13. D. Raj, D. Snyder, D. Povey and S. Khudanpur, "Probing the Information Encoded in X-Vectors," 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), SG, Singapore, 2019, pp. 726-733, doi: 10.1109/ASRU46091.2019.9003979.
14. Mozilla Common Voice, https://voice.mozilla.org/en/datasets
15. Omniglot, https://omniglot.com/soundfiles/
16. RhinoSpike, https://rhinospike.com/
17. VoxForge, http://www.voxforge.org/