Speech recognition for patients with Aphasia

Erik van der Caaij, René Uhliar, Jesse van Noordt, Koray Poyraz and Jeroen Vuurens
Data Science kb-74, Aphasia Research Group
The Hague University of Applied Sciences
The Hague, The Netherlands

Abstract

This paper discusses the possibility to translate audio to phonemes with existing speech to text recognition systems and own development of audio to phonemes system in aid of the therapy of Aphasia patients. Overviews of Sphinx, Kaldi, Google Speech to Text and Sequence to Sequence system are given, which are mostly used for development of speech recognition systems. We then show how Sequence to Sequence is used to create an audio to phoneme system.

Keywords
Speech to Text, Aphasia and Dutch phonemes

1 Introduction

Aphasia is a language disorder caused by a brain injury in one of the language regions of the brain, e.g. trauma or a brain tumor (“What is Aphasia?”, z.d.). As a result, affected patients may exhibit problems in expressing themselves in natural language or understanding language. Several varieties of Aphasia are known to exist, e.g. Broca’s, Wernicke’s, Conduction and more. Aphasia patients get confused when dealing with semantically related words, to make phonetic mistakes. Patients also show different levels of awareness of their language disorder.

Treatment of this disease may (partially) restore the language abilities of patients, although patients rarely recover fully from Aphasia. Traditionally, treatment includes sessions with speech therapists. During treatment, the patients often experience stress as a result of their inability to communicate properly, which may have a negative effect on the efficiency and effectiveness of their treatment.

The Aphasia project studies the possibility of computer-aided treatment of this condition. The intuition is that the stress experienced by Aphasia patients is partially caused by their inability to communicate with another human being. Replacing the other human being with a machine may help to lower the stress experienced in their attempts to rehabilitate their

language capabilities. This study focuses on automated translation of speech into a sequence of phonemes to enable experiments with computer-aided analysis and treatment of patients that exhibit phonetic problems.

Our research question is: How can we transcribe audio to phonemes?

The remainder of this paper is structured as follows: In the second chapter you will find the related work such as literature that we used and the tools and libraries we used. In the third chapter you will find the experiments we have done and show the results that came out of it. In the fourth chapter you will find the results that we have obtained in this project. In the last chapter you will find the conclusion and recommendations for future research.

2 Related work

In the previous chapter we have already explained how Aphasia is related to our project. In this chapter we show important papers in the domain of aphasia related research and audio to text translation.

Some papers in the domain of aphasia related speech research were written by Roelant Ossewaarde. Ossewaarde his research is focused on categorizing patients based on their speech and automated detection of pauses in the speech of aphasia patients (Ossewaarde R. J.-2., 2017) and computerized assessment of the acoustics of progressive aphasia (Ossewaarde R. J., 2016).

Current papers which focus on translating speech to text from aphasia patients are focused on translating audio into complete words. When translating the audio, preparation methods are used on the audio to gain information. In Figure 1, an extraction pipeline is shown which extracts MFCC features (Santosh K.Gaikwad, 2010) (Fayek, 2016). The papers give a good overview of what feature extraction techniques can be applied on audio signals (Santosh K.Gaikwad, 2010) (Fayek, 2016). The techniques and the order in which they are applied to transform the audio data to Mel-frequency Cepstral Coefficient, are shown in the diagram in figure 1.

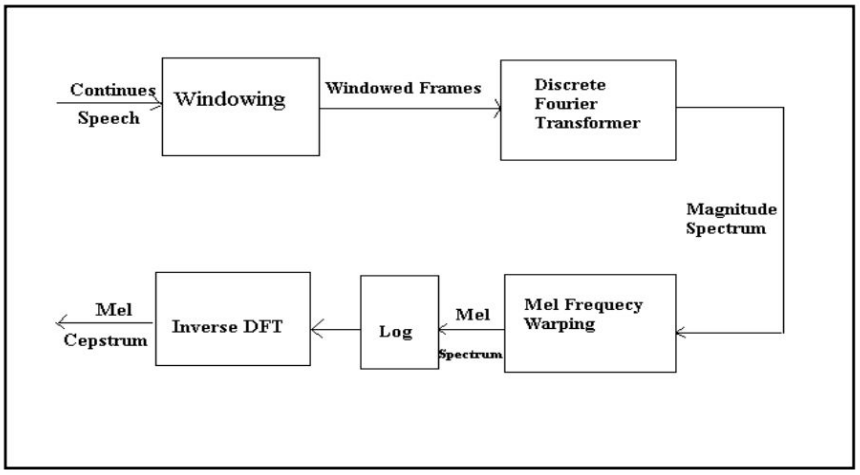

Figure .1: Feature extraction diagram of transforming audio data to Mel-frequency cepstrum.

3 Approach

3.1 Existing approaches

To answer our main research question, we want to get a Speech to text recognition solution that can translate audio to textual phonemes. We do not want the system to correct the incorrect words.

Our first question is whether there are existing tools that can translate audio to text. Google speech to text is the one of the best existing Speech to text recognition systems (“Cloud Speech-to-Text”, z.d.) So we tried to use the Google speech recognition system, to see what existing recognition systems can and what they cannot do. The existing systems cannot predict the incorrect nonsense words that we need to predict for the Aphasia patients. We then tried a Python Library speech_recognition (Zhang (Uberi), z.d.) to see whether this library can predict the nonsense words.

We found PocketSphinx as one of the state-of-the-art libraries in Speech recognition (Shmyrev, z.d.). PocketSphinx is a system that requires three models (Acoustic model, Language model and dictionary). We downloaded these three models for the Dutch language and trained it. We have noticed that PocketSphinx uses a dictionary to predict words. If the word does not exist in the dictionary, same as the Google speech recognition system, PocketSphinx cannot predict this word. So we cannot use PocketSphinx to predict nonsense words as well. We looked further for a state-of-the-art library and we found the Kaldi toolkit.

Kaldi is a state-of-the-art Speech recognition system (“Kaldi ASR”, z.d.), which can be adjusted for a certain purpose. Our purpose is to predict phonemes from correct and incorrect spoken words. Kaldi comes with a pretrained English model, but it can be used for the Dutch language as well. We tried Kaldi with Dutch language by recording voice samples and used this in the Kaldi model. There is a Kaldi tutorial that explains how to implement a new language such as Dutch (“Kaldi: Kaldi for Dummies tutorial”, z.d.). This tutorial led to many errors and we could not finish this experiment in the time that we had for this research.

The existing systems are not able to predict incorrect nonsense words. More research has to be done, specifically on the Kaldi toolkit, because Kaldi looks promising for this goal.

3.2 Generating Dutch speech with annotated phonemes dataset

The goal of generating Dutch phonemes dataset is to train a Dutch phoneme classifier. Learning a Dutch phoneme classifier requires a dataset of Dutch speech with annotated phonemes. We do not have a dataset of Dutch speech with annotated phonemes, so we tried to create our own Dutch phoneme dataset with a phoneme boundary classifier. For this experiment we used the Dutch Corpus Gesproken Nederlands data with more speakers. These data contain the following properties: spoken text, the related wav audio file and a text

file with the begin and end time of a spoken word in the audio file. Thereafter we did create a phoneme boundary generator which creates phoneme boundary dataset for training a phoneme boundary classifier. What this generator does, is merging the last N (e.g. N = 5ms) milliseconds of a word with the begin N (e.g. N = 5ms) milliseconds of the next word. This concatenation is called the “true sub region”. For the “false sub region” the generator slices each e.g. 10ms in a e.g. 50ms parent region. By creating a false sub region, we distinguish true sub region in our dataset. This allows the phoneme boundary classifier to distinguish between true and false for recognizing the boundary. Table 1 shows the result of the evaluation of the following trained models: Random Forest, MLP (Multi-layer Perceptron) and Bi-LSTM. The result gives an overview of training accuracy, validation accuracy, recall (class 0 and 1), precision (class 0 and 1) and f1-score (class 0 and 1). Class 0 stands for false subregion which is not the phoneme boundary and class 1 stands for true subregion which is the phoneme boundary in an audio signal.

3.3 LSTM (Sequence to Sequence)

In recent years the most competitive systems to translate text between languages use a sequence-to-sequence deep learning architecture (Savage, 2018). Sequence to sequence considers the input as a sequence over time, and also predicts the output as a sequence. This is often used in the language domain, but also applies to the translation of other time-series.

In this study we experimented with several existing architectures the have shown to be effective for translating text from one language to another. We preprocessed the raw audio into MFCC features to extract the verbal information. This is considered as a standard preprocessing in speech processing (Santosh K.Gaikwad, 2010). In the first step an LSTM encoder learns a vector from the MFCC feed containing the information that is useful to transcribe the audio. In the second step another LSTM is used to decode this vector into a sequence of phonemes a model of this process is shown in figure 2.

We are trying to use beam search and attention model aside with this LSTM models to get better results.

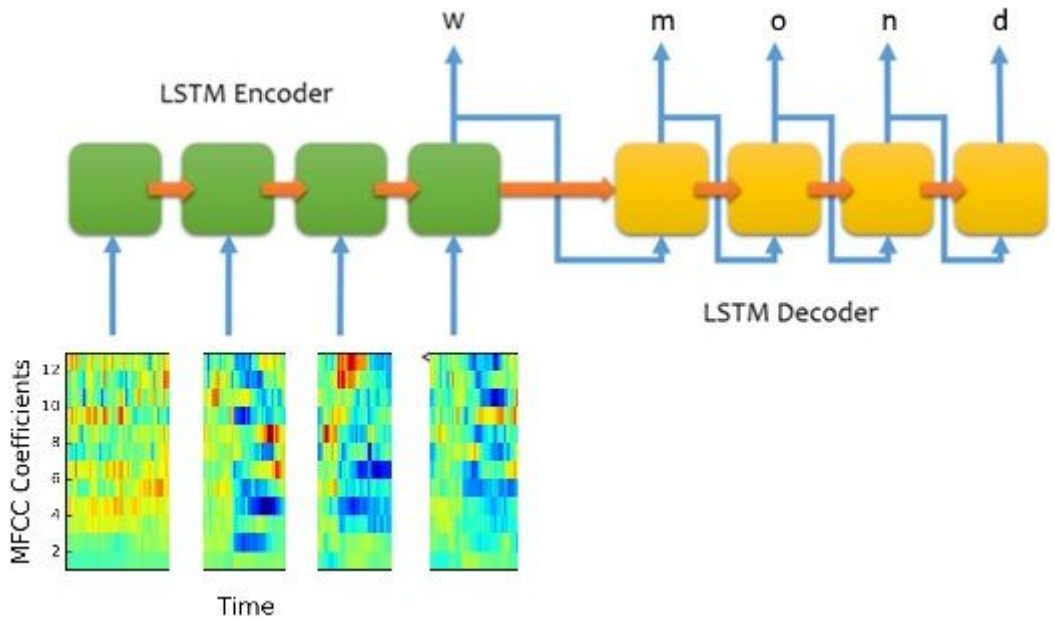

Figure .2: This figure shows an overview of the process where the MFCC input features are being encoded in the LSTM encoder and represented as vectors to the LSTM decoder where they are being decoded to phonemes.

4 Experiments

In this chapter we will explain the experiments which we have done. We created speech to text applications, created a phoneme prediction model and a sequence to sequence solution.

4.1 CGN(Corpus Gesproken Nederlands)

The project of Corpus Gesproken Nederlands is a database in which current spoken Dutch speech is stored. Using this database, we made a selection of audio files we used to create our dataset. The selection we made were only manually verified data. So the data that has been verified by humans whether it is correct. There are various folders in the Corpus Gesproken Nederlands dataset going from comp-a until comp-o. From this subsection of folders we did not include the category of comp-d, because of the incorrect sample rate. We also have excluded the following folders comp-n, comp-l, comp-m, comp-i and comp-a because of insufficient audio quality. From the remaining data we selected the belonging wav files and the text files with the orthographic description.(“Project informatie Corpus Gesproken Nederlands”, z.d.)

4.2 Existing speech to text solutions

For the automatic translation of speech to text there are several solutions readily available. In this section, a few solutions are being described which have been used to see whether one of the existing speech recognition solution will suffice for our project goal.

4.2.1 Google Speech-To-Text API

This project was started with a GSTT (Google Speech-To-Text) experiment. GSTT is an API (Application Programming Interface), which allows developers to implement speech to text solutions quickly and easily.

Python library speech_recognition has been used for this purpose. speech_recognition is a high level ASR library. It allows the developer to, among others configure Google speech to text API to be used for ASR.

First, GSTT has been used to predict spoken English words and sentences and later on to predict Dutch words and sentences. A microphone laptop has been used. It is also important to notice that it depends on the microphone how well the GSTT recognizes words.

4.2.2 Sphinx (PocketSphinx)

For this experiment, a minimalistic speech recognition program was built. A dictionary of just 10 Dutch words has been used. The program predicted around 8 out of 10 words correctly. Later, a dictionary file of around 1.4 million Dutch words has been used. Simple words e.g. “Hallo” were predicted correctly, but more difficult words e.g. “Experiment” were less frequently predicted correctly.

An another kind of experiment was an LSTM-model where MFCC features were given as input from spoken audio and words were expected as output. The MFCC features represented these words.

4.3 G2p-seq2seq toolkit (depends-on PocketSphinx)

PocketSphinx works on a dictionary with words and phonemes. The dictionary in Sphinx already contained around 1.4 million unique Dutch words and phonemes which were considered enough for a first test. The model was trained with the words and phonemes to recognize which phonemes belong to a word.
After the model has been trained, predictions could be made on new Dutch words which are not contained in the dictionary. The model predicted the phonemes that belong to the words. G2P-seq2seq model was used to generate the phonemes of a word.

G2p-seq2seq can be found on GitHub (nurtas-m, 2016). The G2p-seq2seq repository contains clear instructions on how to install and use this toolkit. This toolkit already comes with a pre-trained English word to phoneme prediction model. The instructions can be found in the README.md file on GitHub. The process from README.md file has been repeated to train a Dutch predictive model.

It is important that TensorFlow 1.8.0 or higher and Tensor2Tensor 1.6.6 or higher are installed. These external libraries are required, else the G2p-seq2seq tool will not run. We ourselves ran into multiple issues when these versions were lower, so this is a must.

4.4 Sequence to Sequence(MFCC to phonemes)

In this experiment we used LSTM model to transcribe audio to text. We used an example where English sentences were converted into French sentences by using an LSTM model. We did some tests to see if this method also works on Dutch sentences and MFCC features of an audio signal.

For the first test, we used Dutch sentences as input and output layers. After the first test was complete we wanted to see if the inputs were MFCC features. We wanted to verify whether this would work as well. So in the second test we used MFCC features as input and Dutch words as output.

In the third test we used the MFCC features as input of an audio signal and letters as output. After this test was complete we saw that the model was overfitting a lot, so we used multiple techniques such as dropouts to minimize overfitting.

To accomplish our goal we wanted to use MFCC features as input of an audio signal and the phonemes from the words of the MFCC audio signal as output. This test is not finished yet and this is some future work to do.

For the evaluation, we used the Corpus Gesproken Nederlands dataset. We specifically used the manually checked dataset from CGN and we used the maps that are manually verified(comp-k,o,j,h,o,f,e) to learn the network and used (comp-b,c) for cross validation. We measure the model's effectiveness using the accuracy metrics and the Adam optimizer.

5 Results

5.1 Phoneme Boundary Classifier

In table 1, the results of the models that we have trained on the phoneme boundary dataset are as described in section 3.2.

From the score table we can see that the model MLP (Multi-layer Perceptron) has the best results. MLP scores the highest in the validation accuracy, precision class 1 and second best on recall class 1. These results are marked with green color.

Model	Random Forest	Multi-layer Perceptron	Bi-LSTM
Training acc.	**0.56**	**0.62**	**0.75**
Validation acc.	**0.56**	**0.62**	**0.58**
Recall (class 0)	**0.54**	**0.67**	**0.62**
Recall (class 1)	**0.59**	**0.57**	**0.51**
Precision (class 0)	**0.57**	**0.61**	**0.64**

Precision (class 1)	**0.56**	**0.63**	**0.49**
F1 score class 0	**0.55**	**0.63**	**0.62**
F1 score class 1	**0.57**	**0.60**	**0.50**

Table 1: Evaluation metrics for learning a phoneme boundary classifier.

Because of the noisy dataset, we chose not to proceed with the development of a phoneme boundary classifier.

5.2 Sequence to Sequence(MFCC to phonemes)

The results from the English to French sentences were that almost every sentence was correctly predicted. The loss was also very low for this test.

For the Dutch sentence to Dutch sentence test, the results were about 70% correctly predicted. On a training set which contains 7000 sentences.

We also tried MFCC features to Dutch Phonemes and the accuracy is 91% as shown in figure 3. It is trained in four epochs with 100.000 samples. The batch size is 150, the dropout is 0.5 and the latent dim is 100.

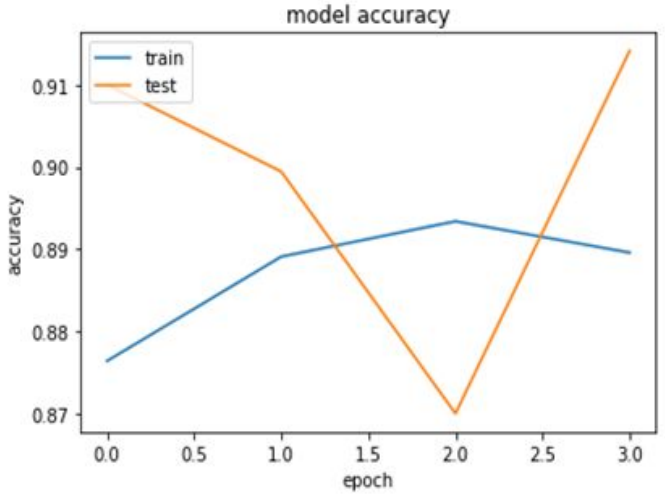

Figure .3: Accuracy results of train and validation for the MFCC features to phonemes. In the result we can see that the lines are not far from each other between 1.5 and 2.5 epochs which means it is not overfitting or underfitting.

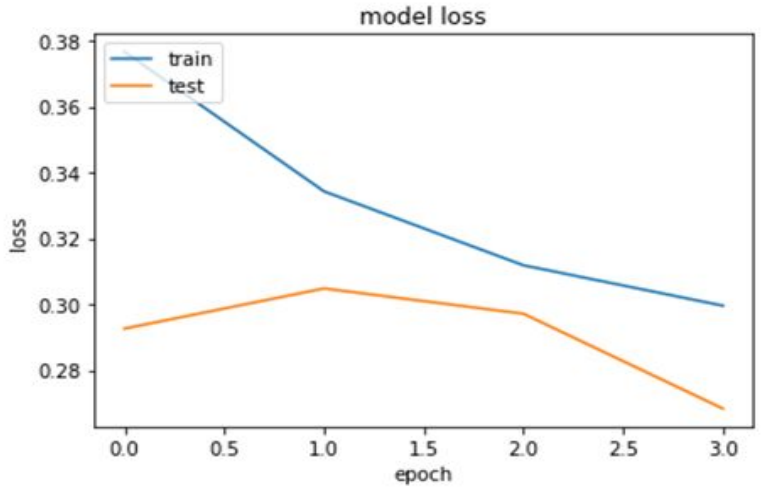

Figure .4: LSTM (Sequence to sequence) model loss between train and validation for the MFCC features to phonemes. In this plot it seems like the training process is going towards overfitting after 2 epochs because the distance between those lines increases.

```
('f|oo|r|\n', 'f|e|l|oo|r|\n')
('i|n|\n', 'i|n|\n')
('d|e|\n', 'd|e|\n')
('d|r|s|t|e|\n', 'ee|r|s|t|e|\n')
('d|o|n|d|e|\n', 'r|o|n|d|e|\n')
```

Figure .5: LSTM (Sequence to sequence) output on the validation set for the MFCC features to phonemes. In this figure we see the target on right side of the comma and the prediction on the left.

Currently we are trying to get better results using an attention model.

6 Conclusion

In this paper we looked into the possibility to translate audio to phonemes in aid of the therapy of Aphasia patients. Existing Speech to Text systems appeared unfit for this purpose because they fail to recognize out-of-vocabulary words. In this study we attempted to overcome this problem by learning a translator for audio to phonemes. We obtained the best results using a LSTM encoder-decoder architecture. However, the results of this model are not yet sufficiently accurate for use with real patients. Our preliminary experiments using error correction on the output are promising.

To learn a system to transcribe audio with phonemes requires a dataset with annotated phonemes. Results we obtained in this study may improve when using data from which the system can learn specific phonemes.

References

Dave, N. (2013). Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition, *1*, 5.

Dahl, D. A., Linebarger, M. C., & Berndt, R. S. (2008, January). Improving automatic speech recognition of aphasic speech through the use of a processing prosthesis, 14.

Ossewaarde, R., Jalvingh, F., Jonkers, R., & Bastiaanse, Y. (2016). Computerized assessment of the acoustics of primary progressive aphasia.. Poster session presented at Science of Aphasia conference 17, Venice, Italy.

Ossewaarde, R., Jonkers, R., Jalvingh, F., & Bastiaanse, Y. (2017). Automated detection of unfilled pauses in speech of healthy and brain-damaged individuals. Abstract from 5th International Conference on Statistical Language and Speech Processing SLSP 2017, Le Mans, France.

Clinic, Cleveland. "What Is Aphasia?" Cleveland Clinic, 2017, my.clevelandclinic.org/health/articles/5502-aphasia.

Cloud, Google. "Cloud Speech-to-Text: Spraakherkenning | Cloud Speech-to-Text API | Google Cloud." *Google*, Google, cloud.google.com/speech-to-text/?hl=nl.

Haytham Fayek, M. "Speech Processing for Machine Learning: Filter Banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between." *Haytham Fayek*, Haytham Fayek, 21 Apr. 2016, haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html.

Cmusphinx, cmusphinx. "Cmusphinx/g2p-seq2seq." *GitHub*, 7 Aug. 2018, github.com/cmusphinx/g2p-seq2seq.

Povey, Daniel. "Introduction." *Kaldi ASR*, kaldi-asr.org/doc/kaldi_for_dummies.html.

Povey, Daniel. "Kaldi ASR." *Kaldi ASR*, kaldi-asr.org/.

Oostdijk, N. "Achtergrond En Motivatie." *Project Informatie*, lands.let.ru.nl/cgn/doc_Dutch/topics/project/pro_info.htm#intro.

Santosh, K.Gaikwad & Bharti, W.Gawali & Yannawar, Pravin. (2010). A Review on Speech Recognition Technique. International Journal of Computer Applications. 10. 10.5120/1462-1976.

Savage, Kristin. "Using AI And ML For Translation Solutions - DZone AI." *Dzone.com*, 14 Aug. 2018, dzone.com/articles/using-ai-and-machine-learning-for-translation-solu.

Shmyrev, Nickolay. "Building an Application with sphinx4." *CMUSphinx Open Source Speech Recognition*, cmusphinx.github.io/wiki/tutorialsphinx4/#using-sphinx4-in-your-projects.

Uberi. "Uberi/speech_recognition." *GitHub*, 31 Oct. 2018, github.com/Uberi/speech_recognition#readme.