Bar-Ilan University

Undergraduate Project

Discriminative Phoneme Alignment for Pronunciation Feedback

Einav Saad (203571435)
Hadas Cohen (204053268)
Nitzan Zeira (204208714)

Supervised by Dr. Joseph Keshet

August 2016

1 Overview

This project proposes a tool that provides visual feedback on the quality of pronunciation in a foreign language. We build an Android application in which the learner of a foreign language is prompted to pronounce a word out of a list, his voice is recorded and analyzed, and he is given feedback on his pronunciation per syllable or letter. For example, a learner of American English is asked to pronounced the word aluminum that should be pronounced canonically as [ah l uw m ah n ah m]. If the learner pronounced the word as [ah l uw m ae n ux m], he will be prompted visually that two letters aluminum were mispronounced.
There are several pronunciation feedback systems that were developed in companies, and are based on phoneme alignment ("forced alignment") algorithms. Given a speech utterance along with its phonetic content, phoneme alignment algorithms find the start time of each phoneme. The phoneme aligner also outputs a confidence for each of the phonemes and a global confidence in its prediction.
Most phoneme alignment algorithms are based on the hidden Markov Model (HMM) algorithm (Brugnara et al., 1993). The HMM is trained as a phoneme recognizer and is used as a forced aligner by restricting ("forcing") its dynamic programming to a given phoneme sequence.
In this project, a new phoneme alignment algorithm which is not based on HMM is explored (Keshet et al., 2007; McAllester, Hazan & Keshet, 2007). The algorithm was examined and changed so as to maximize the measure of performance used to evaluate alignments, rather than to just maximize probabilities, as traditionally done in HMM-based algorithms for speech recognition. The goal of this project is to re-train the algorithm to output reliable pronunciation scores.

2 Project Description

The flow of our system is as follows:

1. In the application we built, a word or a phrase to be pronounced is chosen by the user from a readymade list, where the canonical phonetic content of this word or phrase is known.

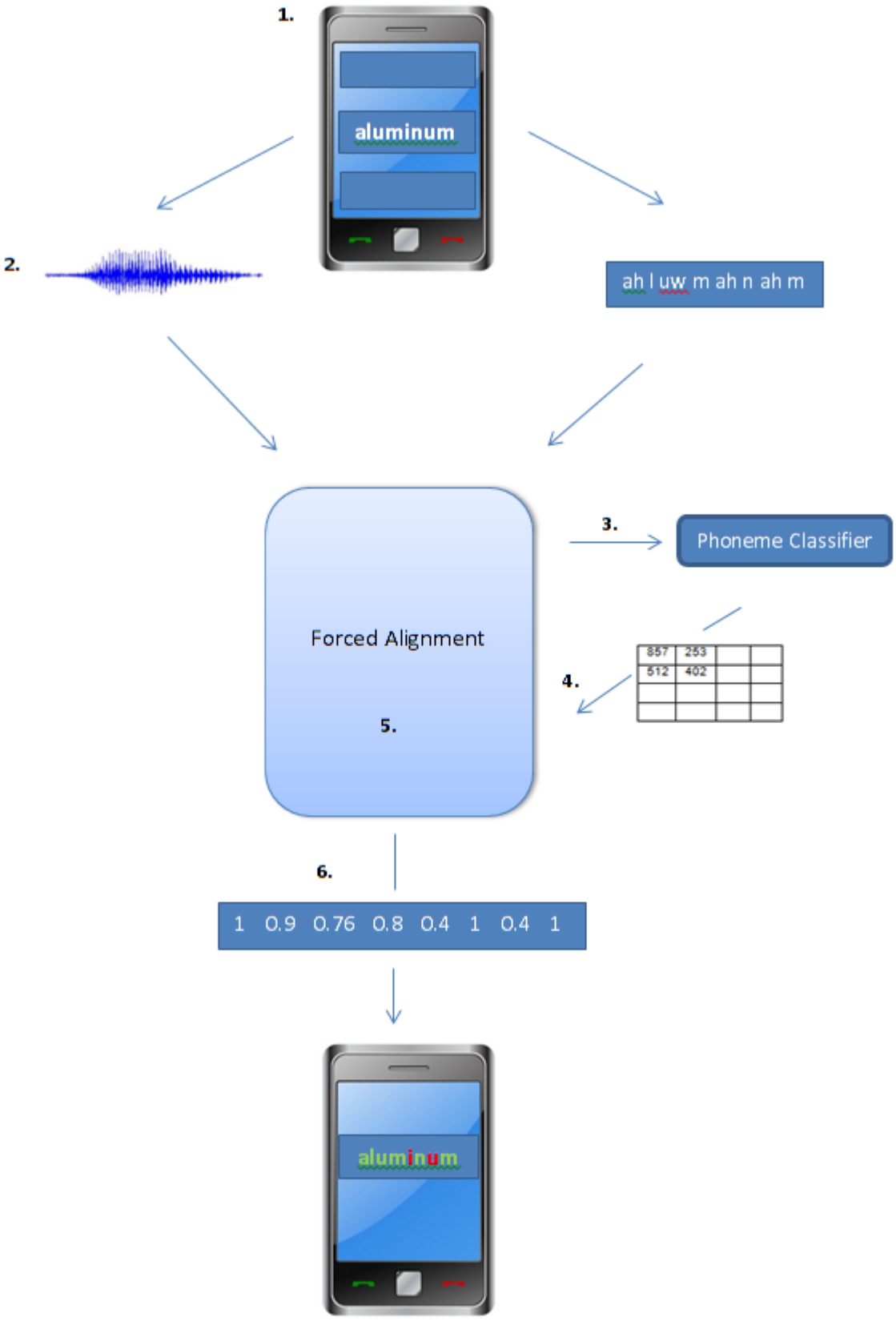
2. The user's voice is recorded and along with its canonical phonetic content it is served as input to a phoneme alignment algorithm on the server side.

3. The algorithm uses a phoneme classifier module to provide scores for each phoneme at each time frame of 10 milliseconds.

4. It finds the most probable start time (frame) of each phoneme by calculating the "path", the partition of the speech into blocks of frames, which maximizes the total sum of scores.

5. For each frame the algorithm compares between the score given to the desired phoneme, which was given as input and was supposed to be spoken in this time frame, and the maximum score given among all the phonemes, which represents the phoneme the user most likely said. The final scores for each phoneme is the ratio between these two scores, which ranges from 0 to 1 and defines the distance between the user's and the desired pronunciation.

6. The server then sends these final scores to the client application, and it presents the mispronounced phonemes (those with lower scores) visually.


1.
aluminum
2.
ah l uw m ah n ah m
Forced Alignment
5.
3.
Phoneme Classifier
4.
857 253
512 402
6.
1 0.9 0.76 0.8 0.4 1 0.4 1
aluminum

3 Implementation

See our GitHub repository containing all of the project code, along with a user guide for using our android application.

4 Milestones

4.1 Technical Setup

Moving the original code and modules from the current working environment on Mac Darwin operating system into our working environment, where we would carry out the research and code changes, on Linux Ubuntu operating system. We run into many technical problems, which required inquiry about the differences between the operating systems, appropriate recompilation of the modules and binaries, and OS-depended changes in the code.

4.2 Background Learning of the Algorithm

Understanding the forced alignment algorithm, by learning the appropriate paper:
- A Large Margin Algorithm for Speech-to-Phoneme and Music-to-Score Alignment – *By Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer and Dan Chazan*
The original algorithm outputs alignment of the speech into the given known phonemes of the spoken word, giving each phoneme a start time, thus dividing the speech into time blocks of frames for each phoneme. As a side effect, it uses scores for each time block which represent the quality of the desired phoneme recognition and directly implies the quality of the user's pronunciation of this phoneme. The problem was that the nature of the scores mentioned above was not very clear. They were distributed in non restricted range, they didn't behave uniformly for different kinds of speeches (such as whole sentences VS single words, American English native speakers VS Hebrew native speakers, etc.), and were not reliable enough to use them as is to make the desired final prediction of the user's pronunciation quality.

4.3 Adapting the Code to Our Needs

Changing the current code implementing the algorithm, so we could get the scores for every possible phoneme, and not only for the word's phonemes received in the input. Doing this allowed us to examine the quality of the scores, and their behavior according to different inputs.

4.4 Creating Data for Experiments

Given the TIMIT dataset containing many speech utterances and their phonetic content, we performed various manipulations such as switching between syllables/phonemes in the same sentence, cutting out single words with their corresponding phonemes or removing / adding silent frames in the beginning/ending of an utterance, in order to achieve utterances in various pronunciation levels.

5 Experiments

5.1 Understanding the algorithm behavior

With the data we created we carried out experiments to figure out the behavior of the algorithm scores. We mainly examined whether it indeed gives good scores for correctly pronounced words\sentences, and bad scores for those which were badly pronounced.

Results

The results informed us with two issues:

- The scores were not always uniform, and an objective look on a standalone score can't determine whether it's good or bad.
- Sometimes a higher score was given to a different phoneme, say 'a', than to the one which was supposed to be said, say 'b', although the user's speech does sound good and similar to 'b', and seemingly has no reason to be interpreted as 'a'.

Conclusions and Hypotheses

These issues raised questions on whether the algorithm is influenced by other speech features such as:

- Duration of the utterance.
- A whole sentence or a single word.
- Silent frames at the beginning/ending of an utterance – do they increase or decrease the quality of the phoneme recognition and hence the final scores, and do cutting them out improves the algorithm performance.

To examine the hypotheses above we have carried out additional experiments.

5.2 Normalization Approach

We have researched for different ways to normalize the scores, in order to make them more robust to the speech differences mentioned above, and mainly to transfer them into a restricted scale which we could use to determine a threshold for classifying the pronunciation quality. E.g. For a scale between 1 - 5 the score is higher as the quality gets better.
We have learned the following paper:
- **Automatic scoring of pronunciation quality** – *By Leonardo Neumeyer, Horacio Franco, Vassilios Digalakis and Mitchel Weintraub*
which suggests different methods to normalize the score of an utterance into a scale of 0-1.
After consultations about the normalization methods suggested, we applied the following:

```
For each time block i representing a spoken phoneme:
    For each possible phoneme j:
```

$$scores(i,j) = \frac{e^{scores(i.j)}}{\sum_{l=1}^{k} e^{scores(i.l)}}$$

Results and Conclusions

We found that:

- Normalization does not significantly improve the scores quality.
- Reduction to a smaller scale makes it more difficult to realize a proper threshold between "bad" and "good" scores, as the scale shrinks.

5.3 Types of Speech Input

- We tried to measure the score of a word when it's part of the sentence VS when it stands by itself.
  Results and Conclusions
  We discovered that when the utterance is only a single word the algorithm does not influenced by the word's context inside the sentence, and does perform more accurately.

- We tried to measure the score of a speech utterance with silent frames at its beginning/ending and respectively add the 'sil' signals to its phonetic content, VS when cutting the silent frames out.
  Results and Conclusions
  We found that the algorithm perform slightly better when cutting off the silent frames.

5.4 Types of Final Scores Output

We examined various approaches for determining the most accurate final scores the algorithm should output for the user's phonemes pronunciation:

- Output the original score calculated by the phoneme classifier module, which regards to the right phonemes for the word, as given in the input.
  Conclusions:
  Not so good, as we mentioned above. Non uniform scores, that do not match to a restricted scale which allows clear classifying of 'good' and 'bad' pronunciation quality.

- Outputthe phoneme having the maximal score given by the phoneme classifier, for each block of frames in the utterance, and present it as the output to the user – as to tell him he sounded more like this phoneme than the desired correct one.
  Conclusions:
  Non measurable output which does not exactly meets the original goal of pointing the user towards how far he is from correct pronunciation.

- Output the difference between the two scores mentioned above: the score of the correct phoneme and the maximal score which implies the phoneme the user most likely said.
  Conclusions:
  Better, since addresses the problems with the previous approaches. But still not good enough, because it ignores the scores' original scale (since it only gives their difference), thus not always representing the severity of the difference. In addition, does not solve the scaling problem – do not give a clear range and threshold for the pronunciation quality.

- Output the ratio between the two scores mentioned.
  Conclusions:
  So far the best and most accurate solution, which addresses all the problems we ran into.

6 Main Struggles

- When the practical part of the project, the android application, was already functioning we could use it to collect data of our own, and measured the algorithm performance on our own records. We discovered then that the scores it outputted were different from the ones we were used to so far, when measuring data from American English native speakers.
  We raised a hypothesis that the reason for the difference is that the current phoneme classifier was trained only on English native speakers data, and could have interpreted wrongly the slight differences in the Israeli accent.
  As we tried to confirm this hypothesis we:

1. Created corpus of data from our own records – to represent Hebrew native speakers' pronunciation.
2. Extracted new data of English native speakers' pronunciation from youtube.
3. Compare the algorithm performance upon them.

At first it seemed that our hypothesis is correct, since the scores did behave differently for both kinds of data.
But a deeper examination of the scores, along with Dr. Keshet estimations brought us to a new conclusion that what is needed is re-calibration of the thresholds we used for classifying 'good' and 'bad' pronunciation.
We also figured out it's best if we add another 'medium' level to the thresholds, for scores that are not strictly fall into neither of the definitions of completely 'good' or 'bad'.

- The android application we built is based on communication with a server running the forced alignment algorithm. It sends the recorded utterance along with its phonetic content and receives the scores for each phoneme.
  During the development we used a local server running on our Linux virtual machines, but for the final project delivery we obviously needed a proper server with a static IP address which will be running continuously.
  With the help of Dr. Keshet we received a server to work on from the university. We ran into several technical problems with it, which mostly resulted from the fact it was running a windows operating system, and the use of Cygwin, which couldn't run the algorithm modules and binaries properly as is and required multiple adaptations, re-compilation and structure changes.
  Another issue was that we had problems with accessing it from outside of the university network.
  While trying to handle these issues, we searched and tried various alternatives. We decided the best way for now is to use Google Cloud Platform services to host a virtual machine running Ubuntu, where we copied and installed all that is needed to run our server.

7 Future Thoughts

1. For future progressions in general; if one wants to apply the algorithm on a more specific types of speakers, he can always train the phonem classifier on a this specific type of speech data in order to achieve better and more accurate results.
   (a) With the purpose of applying the application for Hebrew native speakers learning English, one can use a better phoneme classifier module, which is more robust to Isreali speech, thus can output more accurate scores and enhance the total algorithm performance.
   (b) In addition, the application, and also the algorithm generally, can be adapted for the use of teaching deaf to speak appropriate English, with correct pronunciation.
2. **Readymade records of the words:** A good and useful feature in the application could be addition of an option for the user to hear every word as it should be pronounced.This would require a qualitative and uniform data, where all utterances are pronounced clearly,and in the same audio format.
3. **Personal accounts:** A feature in the application that allows multiple users to use it, each having a personal account.
4. **User speech recognition:** Enhance the previous feature by adding a possibility to recognize a user with speech recognition algorithm, that might be based on the user's specific pronunciation nuances.
5. **Scalability:** Addition of option for easily replace the phoneme system type.This would require a data corpus of words with their phonemes in the new system, and a new phoneme classifier module.