Lab session 3

Summary

Today we will run the `Aligner`. This tool takes a .wav signal file and an accompanying .txt text file with the orthographic transliteration of what was said in the speech signal. It then looks up the pronunciation of each word in the .txt file in a dictionary (Beware, for unknown words, it must generate pronunciations automatically, and they can be very wrong). With this expected sequence of phones, it then generates label files for phones, syllables, and words, indicating the most likely position of each word, syllable, and phone, considering the acoustic signal.

Please note that the `Aligner` does not check the sequence of phones for plausibility, nor does it have any means of identifying where the acoustics do not match the phones well. It just gives the most likely alignment, blindly assuming that the expected sequence of phones is correct. This way of generating an annotation is called **forced alignment**.

We will use the Aligner today, and analyze the resulting label files both using Linux shell commands and `praat`. Then we will experiment a bit with the Aligner.

Running the `Aligner`

Unfortunately the Aligner is only installed on the IMS Linux machines. So in order to do this lab, you will either have to use one of the machines in the pool, or **if you use your own computer, you will have to log onto one of the Linux machines using a Terminal. You will then have to copy the files you generate back and forth between the IMS machines and the computer you use, because we will use your local `Praat` to look at the results**.

Preparing the files

To run the `Aligner`, we need a speech signal plus its orthographic transcription in a separate file. I provide two files as an example, `english.example.wav` and `english.example.txt`. You can find them in

```
/mount/studenten/MethodsCL/2019/Speech/Lab3
```

on the IMS Linux machines, or on ILIAS.

Open a terminal, and copy `english.example.wav` file and its corresponding `.txt` file from the above directory to some directory where you have write permissions. Change to this directory in the shell using the `cd` command.

Where to find the Aligner for today

The `Aligner` can currently deal with English and German, and usually it's available by invoking Alignphones or Alignwords. Unfortunately for English it is set up to produce phoneme label files with British English SAMPA output, while we discussed American English ARPA phonemes in class. So for this lab, I'm offering a modified version of the Aligner which will produce ARPA. (However, you can see in the output while the Aligner is running that the pronunciations were originally created in SAMPA).

The ARPA variant of the Aligner can be found here

```
  /mount/studenten/MethodsCL/2019/Speech/Lab3/tools
```

Generating phones label files

To generate the phones label files, run

```
  /mount/studenten/MethodsCL/2019/Speech/Lab3/tools/Alignphones english.example.wav
```

This will, after printing a lot of messages about its progress, generate a file `english.example.mfc`, which you can think of as representing the spectral properties of the speech signal.

Beware, the `Aligner` generates the `.mfc` file only if it does not exist yet, and the way it is generated differs between English and German. **If you change the .wav file in any way (e.g. cut out a part) or if you have run the unmodified version of the Aligner, which has German as the language, instead of the one found above, make sure that you delete the `english.example.mfc` file before you run the `Aligner` again**, because otherwise it will still use the wrong `english.example.mfc` file.

The main result of the above command is several label files in ESPS format, only one of which are of interest today: the phones label file, which is called `english.example.phones`.

It is in ESPS label file format. This is different from the `praat` TextGrid format, and much simpler. ESPS is still very wide-spread. The Boston radio news corpus, from which I took the utterances for all three lab sessions, also provides its annotations in that format.

Use `cat` or `less` to inspect the `.phones` label file and make sure you understand the format: there is a short header, which is ended by "#". The `Aligner` generates just an empty header, so the # will be in the first line. Then we have three columns in each line: the time stamp corresponding to the end of the label, a color (an integer), which would matter only if you were to display the labels in a very old software called xwaves, and the label itself.

Thus we have one line for each phoneme in case of the `.phones` label file, plus the header. This makes it very easy to quickly count the number of phones, or the number of phones of a certain type using very basic shell commands!

Analyzing the ESPS label file using Linux

You can for instance use `grep` on the result to extract only lines that contain a label (i.e., not the ones from the header). Here are some examples of how `grep` can be used, for more go back to the Linux tutorial we did earlier:

- `grep "abc"`
  would extract all lines that contain the string abc
- `grep "[0-9]$"`
  would extract all lines that end with a digit
- `grep " \(a\|o\)"`
  would extract all lines which have a space followed by either a or o

Try to find a regular expression that will extract only the lines containing labels. There are lots of ways to do this in this example. You can assume that the Aligner always produces an empty header, and that no phoneme label contains a # sign.

Once you have the correct regular expression to extract only the lines containing phone labels, use it in a pipe with `wc` to count the number of lines in your label file[1]. How many phones does this news story contain? In other words, how many phones does your `.phones` file contain?

The ESPS label format together with Linux commands provides a very simple and efficient way to do simple corpus statistics: Try to find a regular expression that will give you only unvoiced stop consonants (e.g., only lines containing p, t, k). Use `grep` and `wc` to determine how many unvoiced stops there are in your .phones file. In the same way, you could do statistics over an entire speech corpus by running `grep` on all label files from that corpus in one go.

grep "[[:lower:]]" english.example.phones | wc -l

1 i.e. `grep "bla" | wc -l`

Generating words label files

To generate the words label files, do

```
/mount/studenten/MethodsCL/2019/Speech/Lab3/tools/Alignwords english.example.wav
```

The notes from above concerning output messages and the `.mfc` file apply again.

The command generates a file called `english.example.words`, which contains the word labels. You can again have a look at it using `less` or `cat`.

Checking the results using praat

If you are working at your own laptop, copy the two label files (phones and words) and the .wav file to your local computer.

Start `praat` and load the .wav file in the usual way. You can then load the two label files all at once using

```
Open -> Read from special tier file... -> Read interval tier from Xwaves...
```

This creates two objects in the object window. Select them both together by holding the Ctrl or Strg key, and hit the button

```
Into TextGrid
```

Then select the resulting TextGrid and sound together, and do `View & Edit`.

Inspect the TextGrid.

Some experiments with the `Aligner`

Correcting the .txt file

The example contains an error: the speakers says U Mass Boston, but the `.txt` files contains University Mass Boston. Check the annotation around this area, then correct the `.txt` file, re-run the `Aligner`, and check again.

Modify the .txt file

To remind yourself of the fact that the `Aligner` blindly assumes that it must find all the phones specified, add some words to the `.txt` file that the speaker does not say, and re-run the `Aligner`. Check around the area where you added the words and see for how long the labels are affected before the labels match again.

Delete some words, and check the outcome.