Sophia Davis
NLP final project write-up
for 11/25/13

Language Recognizer

Files:

recognizer_util.py -- helper methods for other files

for 'codebook' method:

- recognizer_codebook.py
- train_recognizer_codebook.py

for 'simple' method:

- recognizer.py
- train_recognizer.py

result generation:

- script_2speakers.py
- script_3speakers.py
- script_4speakers.py

PDF's of script output

.wav files of cleaned excerpts from UN speeches

I. Problem

Humans can easily distinguish between spoken languages with fairly high accuracy, even if they are not fluent in the language that they are hearing. The goal of my project was to determine how accurately a computer could predict the source language of a spoken recording.

II. Corpora

Data Selection:

I downloaded oral translations of speeches in Russian, French, and English from the September 24th session of the UN General Assembly (http://gadebate.un.org/). My initial data came from two speeches: one by President Obama and one by Fredrik Reinfeldt, the president of Sweden. Later, I also downloaded speeches by Cristina Fernández, president of Argentina, and Viktor Yanukovych, president of the Ukraine. Each speech was translated by a different person, so my data consisted of recordings from four different speakers of each language.

After downloading the speeches in mp3 format, I converted them to uncompressed integer wav files using VLC. The speeches were of varying quality: volume, amount of background noise, and length of pauses all differed significantly from speaker to speaker. Using Audacity, I selected three sentences from each translation of each speech. Initially, I intended to select exact translations of three different sentences. I eventually gave up on this and opted to select three higher quality sentences from each speaker, regardless of which sentences these translations corresponded to.

Data Cleaning:
In an attempt to standardize my corpus, I used Audacity to perform basic data clean-up. First, I removed large pauses (somewhat arbitrarily: I hit zoom twice and removed spaces bigger than my pinkie). Then, I used Audacity's 'Noise Removal' and 'Normalization' functions (with default settings). Although this did make speech volume more uniform across speakers and reduced background noise, it also changed the quality of the voices: they became more robotic and tinny (especially speakers who originally spoke very quietly). Background noise was still very loud on some recordings.

There are other potential problems which I did not attempt to fix. For example, several speakers spoke with accents, meaning that language models were perhaps trained on non-standard versions of each language. The recordings also varied in length from about five seconds to as long as 20-30 seconds. This was because I initially intended to do an analysis of the change in pitch in one sentence between languages, so I wanted each recording to consist of only one complete sentence, as opposed to being a set length. I am not sure how the quality of these recordings affected the performance of my program.

Training/Test Set Generation:
Two-Speaker Model:
I formed my initial models with data from two speeches (Obama and the PM of Sweden). I had three recordings from each of two speakers for all three languages. By training models on one recording from each speech for all three languages, there were $3^7 = 2187$ possible different training sets. I wrote a script that generated every possible permutation and randomly selected a certain number. For each of the chosen training sets, the script trained a model, and then tested this model on all recordings not used in training.

Four-Speaker Model:
Next, I added data from the two remaining speeches, resulting in a total of four speakers representing each language. First, I formed a model that trained on one recording from each of the four speakers for all three languages. As before, I wrote a script to calculate all different permutations and randomly select a certain number. For each of the chosen training sets, the script trained a model and then tested on all recordings not used in training.

Unfamiliar-Speaker Model:
Finally, from each randomly selected training set (with four speakers), I randomly selected three of the four recordings from each language. A model was trained on these recordings, and tested only on the fourth recording (from each language) which had not been used to train the model, i.e. the model had not been trained on the voice of the speaker of the test file.

III. Methodology

Basics:
The first step for processing recordings was the same for recordings used in both training and test sets. I used the python_speech_features library (http://python-speech-features.readthedocs.org/en/latest/) to calculate the energy and the twelve independent cepstral coefficients on each 25 ms window of a recording (window step of 10 ms). From these, I calculated the delta and double-delta values (a measure in change in energy and in each

cepstral coefficient value over windows). This resulted in one 39-value vector for each window.

Simple Method:
For my initial model, I simply averaged the MFCC vectors from all windows in a recording. This method assumes that since the sounds of English, French, and Russian are different, the averages of these sounds will also be different.

I trained models on multiple recordings from each language by averaging the results from all recordings from one language, which produced one 39-value MFCC vector for each language.

Similarly, I found the average MFCC vector of the test recording. Then, I calculated the Euclidean distance between this vector and the three average MFCC vectors representing each different language in the trained model. The language in the model which produced the pairing with the lowest Euclidean distance was returned as my program's source language guess.

'Codebook' Method:
My next attempt was also based on the 39-value MFCC vectors. After calculating the MFCC vector on each window, I determined the frequency at which the maximum sound intensity (first formant, or F_1) occurred on that window. I only considered MFCC vectors from windows where the first formant was within the range of human speech, hoping that this would help reduce the potential confounding factor of background noise (I defined this range as 200 to 3500 Hz, based on http://en.wikipedia.org/wiki/Formant. The French Wiki article listed an even smaller range, and the Russian Wiki article neglected to give any numbers. Thanks Russia. So I assumed this range would also suffice to capture the frequency range of spoken Russian.)

I divided this range (on the Mel-scale -- which seemed to work better than linear scale) into a certain number of "F_1-bins" (5, 7, or 9), and only averaged MFCC vectors from windows with F_1 values in the same bin, thus forming a 'codebook' of 5, 7, or 9 MFCC vectors. My assumption was that by grouping information from windows with similar F_1 values, I would capture information about certain vowel sounds in each language, and thus be able to compare individual language features. This is in contrast to the simple method, which, by taking the overall average and lumping information about all sounds together, might have reduced variation between individual sounds in each language.

To compare a test recording with language data in the model, I used the same F_1-bin scale as used for training to group MFCC values, and calculated the average MFCC vector in each bin. Then, I found the Euclidean distance between the average MFCC vectors of corresponding bins in the test recording and each language in the model.

I used two different methods to predict the source language of the test recording. First, for each language in the model, I found the average Euclidean distance over all corresponding F_1-bins, and returned the language pairing with the smallest average distance as a source language guess. I also tried looking at each language in the model, comparing vectors at each corresponding bin, and simply returning the language which produced the single lowest overall Euclidean distance between corresponding bins.

The 'codebook' method took significantly more computation power, and these models were thus much slower to run.

IV. Problems

My main problem was the dubious quality of my corpus (background noise, pause removal, accents, differing length of recordings). This occurred because I could not find any reliably constructed corpus containing examples of speech from several different languages, and formed and cleaned my own corpus.

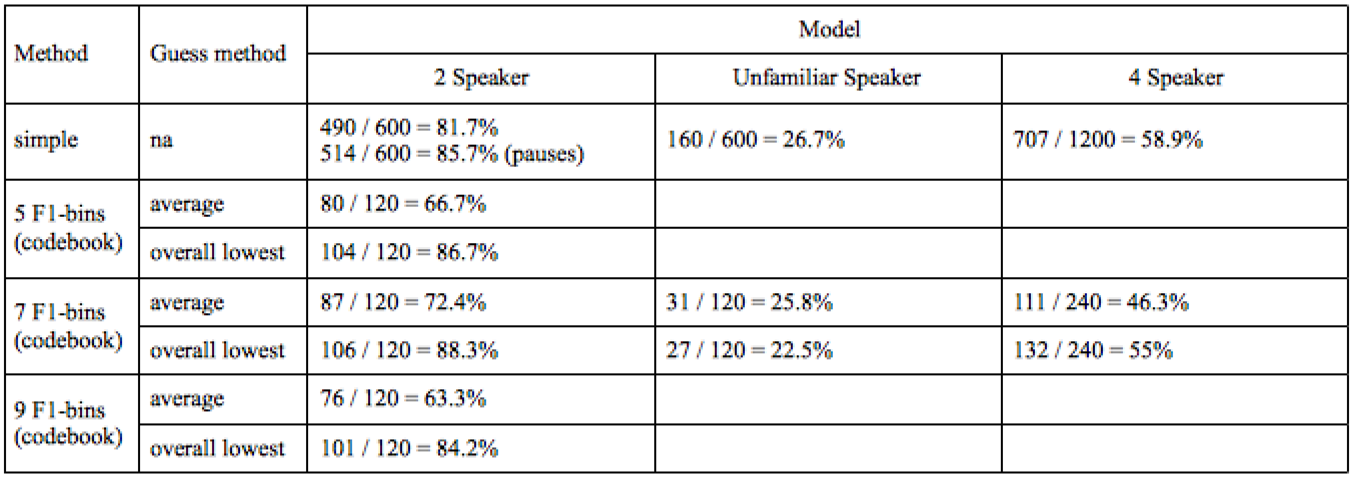
Another problem, especially for the ‘codebook’ method, was the processing time. I had to re-loop through each 25 ms window to find the frequency of the first formant. If I had been able to write a function to calculate MFCC values myself, I would have been able to find the first formant and MFCC vector at each window in one iteration. However, to simplify my project, I was using a library to calculate MFCC vectors, which took only an array of sound samples as input. Thankfully, the library itself was very speedy.

Furthermore, I had many variations of training sets to consider. Training and testing on all permutations of the training set for the two-speaker model using the simple method would have taken over 45 hours. The codebook method was not only slower, but also introduced even more variables (number of bins and method for returning sample language guess) that needed to be tested. I was forced to only run a fairly small number of permutations of each method and variation.

V. Results

For the simple method, success rates were calculated by training on 50 different random permutations of training data (two- and four-speaker models) or 200 random permutations (for the unfamiliar-speaker model -- less test files were available for each training set). The running time of the ‘codebook’ method was significantly longer. There were also more variations to test: changing the number of F_1 bins used to create the codebook or using a different way to return source language guess resulted in different success rates. As such, only one fifth as many permutations were run for the codebook method.

Overall results:
I have included more specific results in an appendix. The output from my scripts are included as PDF files in my assignment submission.



Method	Guess method	Model		
		2 Speaker	Unfamiliar Speaker	4 Speaker
simple	na	490 / 600 = 81.7% 514 / 600 = 85.7% (pauses)	160 / 600 = 26.7%	707 / 1200 = 58.9%
5 F1-bins (codebook)	average	80 / 120 = 66.7%		
	overall lowest	104 / 120 = 86.7%		
7 F1-bins (codebook)	average	87 / 120 = 72.4%	31 / 120 = 25.8%	111 / 240 = 46.3%
	overall lowest	106 / 120 = 88.3%	27 / 120 = 22.5%	132 / 240 = 55%
9 F1-bins (codebook)	average	76 / 120 = 63.3%		
	overall lowest	101 / 120 = 84.2%		

For models trained and tested on only two speakers per language, success rates were surprisingly high. Results ranged from 81.7% accuracy (simple method) to 88.3% (codebook method with 7 bins, returning the overall lowest bin distance). For two-speaker models, the codebook model was more successful. This is hopefully because it succeeded at pinpointing more specific differences between languages, and compared the languages on the level of individual sounds. The fact that two-speaker models were most successful when the lowest overall distance between frequency bin MFCC vectors (as opposed to average distance) was used as a source language guess also supports this idea (by averaging distance between bins, I was possibly reducing the variation in differences between the languages). However, the unfamiliar-speaker model (tested on voices that had not been used for training) was slightly more successful with average difference between bin MFCC vectors.

For two-speaker models, the codebook method seemed to produce better results with 7 bins than with 5 or 9 bins, for both methods of determining source language guess. I do not know exactly why this is the case, but I hope it has to do with capturing differences in the frequencies of common vowels in Russian, French, and English, and cannot be attributed to random chance. Because the unfamiliar-speaker and four-speaker models took so much longer to run than the two-speaker models, I only tested these models with the 7-bin version of the codebook method.

I was dismayed to see that the simple method performed better on recordings from which I had not removed pauses (85.7%), than on recordings from which pauses had been removed (81.7%). I suspect this is because these models merely enable the program to “recognize” speakers’ voices. Some of the translators paused much more than others (these pauses seemed more indicative of personal speaking style than language). This probably lowered the average MFCC values of their speech significantly and made it even easier for the program to identify their voice. On the other hand, perhaps the amount of time a speaker pauses is related to the language they speak, and I reduced the accuracy of my project by removing these pauses.

In any case, my hypothesis was that if my program was indeed simply recognizing speakers, by adding data from more speakers, the success rate would decrease. Sure enough, for models trained on four different speakers per language, when tested on recordings by those same speakers, success rates decreased for all methods. Surprisingly, all tested variations of the codebook method were worse than the simple method when data from more speakers were added.

Unfamiliar-speaker models (trained on three voices from each language and tested on a fourth voice) performed worst of all. This makes sense: three voices is clearly not enough training data in order to identify any random speaker of a language.

VI. Further Improvements

Most obviously, if I had more computational resources and time, I could run more permutations of each type of method. However, there is nothing in my results to suggest that, for example, the low success rates of the unfamiliar-speaker model were due to a 'bad' batch of training sets -- results were very similar for all models.

Ideally, I could find a clean, normalized corpus of conversational phrases in different languages from many different speakers. This would help my project be more practically applicable. People usually already know what language politicians are speaking (often indicated by the flag next to their face), but this type of a program could be actually useful if it could reliably identify source language based on a recording of any speaker. Similarly, finding a corpus with data from more languages would help my project be more applicable to real-world situations. It would also be interesting to see how training and testing on a wider variety of languages affected my success rate. I suspect that the models would do worse differentiating between related languages, such as Russian and Czech, than on very different languages, like English and Chinese.

I might be able to increase success rate as well. By tweaking settings (perhaps smoothing the intensities on each window before calculating the frequency of the maximum intensity), my models might become more sensitive to differences in spoken languages. There are also other ways to compare languages, even based on my simple codebook idea, for example, by tallying the number of bins pairings with lowest distance for each language, or only averaging the lowest two or three distances.

Then I'm going to sell it to the NSA.

VII. Appendix

Detailed Results from Select Models:

The column headers of each chart indicate the actual source of the recording. The row headers indicate what the recognizer program returned as a guess at source language. Correct guesses are colored green.

a. Simple method, two-speaker model: 490 / 600 trials correct = 81.7%

Data: two speeches

Training: one recording from each speech

Testing: all recordings not used to train model

Results from 50 random permutations of training sets

script_2speakers.py, codebook = False

	French (200)	Russian (200)	English (200)
French	192 = 96%	41 = 20.5%	7 = 3.5%
Russian	8 = 4%	135 = 67.5%	30 = 15%
English	0	24 = 12%	163 = 81.5%

b. Simple method, two-speaker model, with pauses: 514 / 600 trials correct = 85.7%

Data: two speeches

Training: one recording from each speech

Testing: all recordings not used to train model (recordings NOT cleaned of pauses)

Results from 50 random permutations of training sets

script_2speakers.py, codebook = False

	French (200)	Russian (200)	English (200)
French	186 = 93%	26 = 13%	0 = 0%
Russian	14 = 7%	147 = 73.5%	19 = 9.5%
English	0 = 0%	27 = 13.5%	181 = 90.5%

c. Simple method, four-speaker model: 707 / 1200 trials correct = 58.9%

Data: four speeches

Training: one recording from each speech

Testing: all recordings not used to train model

Results from 50 random permutations of training sets

script_4speakers.py, codebook = False

	French (400)	Russian (400)	English (400)
French	216 = 54%	19 = 4.8%	22 = 5.5%
Russian	55 = 13.8%	187 = 46.7%	74 = 18.5%
English	32.2%	194 = 48.5%	304 = 76%

d. Simple method, unfamiliar-speaker model: 160 / 600 trials correct = 26.7%

Data: three speeches per language (chosen randomly)

Training: one recording from each speech

Testing: all recordings from speaker whose voice was not used to train model

Results from 200 random permutations of training sets

script_3speakers.py, codebook = False

	French (200)	Russian (200)	English (200)
French	72 = 36%	67 = 33.5%	64 = 32%
Russian	59 = 29.5%	43 = 21.5%	91 = 45.5%
English	69 = 34.5%	90 = 45%	45 = 22.5%

e. Codebook method, two-speaker model, 7 bins, ‘average’ guess method: 87 / 120 trials correct = 72.4%

Data: two speeches per language
Training: one recording from each speech
Testing: all recordings not used to train model
Results from 10 random permutations of training sets
script_2speakers.py, codebook = True
average = True, n_bins = 7

	French (40)	Russian (40)	English (40)
French	31 = 77.5%	11 = 27.5%	10 = 25%
Russian	8 = 20%	28 = 70%	2 = 5%
English	1 = 2.5%	1 = 2.5%	28 = 70%

f. Codebook method, two-speaker model, 7 bins, ‘lowest distance’ guess method: 106 / 120 trials correct = 88.3%

Data: two speeches per language
Training: one recording from each speech
Testing: all recordings not used to train model
Results from 10 random permutations of training sets
script_2speakers.py, codebook = True
Average = False, n_bins = 7

	French (40)	Russian (40)	English (40)
French	40 = 100%	8 = 20%	0 = 0%
Russian	0 = 0%	27 = 67.5%	1 = 2.5%
English	0 = 0%	5 = 12.5%	39 = 97.5%

g. Codebook method, four-speaker model, 7 bins, ‘average’ guess method: 111 / 240 trials correct = 46.3%

Data: four speeches
Training: one recording from each speech
Testing: all recordings not used to train model
Results from 10 random permutations of training sets
script_4speakers.py, codebook = True
Average = True, n_bins = 7

	French (80)	Russian (80)	English (80)
French	28 = 35%	18 = 22.4%	7 = 8.8%
Russian	29 = 36.3%	31 = 38.8%	21 = 25.3%
English	23 = 28.8%	31 = 38.8	52 = 65%

h. Codebook method, four-speaker model, 7 bins, ‘lowest distance’ guess method: 132 / 240 trials correct = 55%

Data: four speeches
Training: one recording from each speech
Testing: all recordings not used to train model
Results from 10 random permutations of training sets
script_4speakers.py, codebook = False
Average = False, n_bins = 7

	French (80)	Russian (80)	English (80)
French	40	4	2
Russian	10	30	16
English	30	46	62

i. Codebook method, unfamiliar-speaker model, 7 bins, ‘average’ guess method: 31 / 120 trials correct = 25.8%

Data: three speeches per language
Training: one recording from each speech
Testing: all recordings from speaker whose voice was not used to train model
Results from 40 random permutations of training sets
script_3speakers.py, codebook = True
Average = True, n_bins = 7

	French (40)	Russian (40)	English (40)
French	10 = 25%	14 = 35%	13 = 32.5%
Russian	18 = 45%	5 = 12.5%	11 = 27.5%
English	12 = 30%	21 = 52.5%	16 = 40%

j. Codebook method, unfamiliar-speaker model, 7 bins, ‘lowest distance’ guess method: 27 / 120 trials correct = 22.5%

Data: three speeches per language
Training: one recording from each speech
Testing: all recordings from speaker whose voice was not used to train model
Results from 40 random permutations of training sets
script_3speakers.py, codebook = True
Average = False, n_bins = 7

	French (40)	Russian (40)	English (40)
French	14 = 35%	16 = 40%	9 = 22.5%
Russian	16 = 40%	4 = 10%	22 = 55%
English	10 = 25%	20 = 50%	9 = 22.5%