Code-mixing Detection Using Acoustic Features

Xinyi Wu & Claude Zhang
LING 575 Project Report

Motivation

Code-mixing(CM) is an interesting and underdeveloped field due to the limitation of CM data and its variousness in phenomenons. There are usually three types of models to detect a code-mixing speech: acoustic models, pronunciation models, and language models.
Most of the code-mixing detection model are build on language models (Vu & Schultz 2014, Yilmaz & Leeuwen 2018). We are interested in finding out whether using the acoustic features of an utterance can predict if code-mixing is happening.

Corpus

Eppler Corpus (1999) is corpus created and annotated by Eva Eppler, who interviewed a community of Austrian Jewish refugees from Nazi occupied Austria who settled in Northwest London in the late 1930s. Subjects are 10 people varied in ages and genders in that community. They were audio-recorded in one-to-one or one-to-multiple sociolinguistic interviews/oral history collections. The total length of the interviews are around 650 min. The dominant language in this corpus is German, but English plays about 30 percent in total in the corpus.

Full transcripts were made of sound files using the CHAT/LIDES transcription systems, extended to deal with code-mixed data.
Here is a short excerpt of the transcription (*afred1.cha*):

**ALA:* *[- eng] I made one application die@s [x 3] wie@s heißt@s die@s bank@s ? 1165186_1175124*
%mor: *pro:sub|I v|make&PAST det:num|one n|application L2|die L2|wie L2|heißt L2|die L2|bank ?*
%gra: *1|2|SUBJ 2|0|ROOT 3|4|QUANT 4|2|OBJ 5|4|OBJ 6|5|CONJ 7|6|COORD 8|7|CONJ 9|8|COORD 10|2|PUNCT*
**ALA:* *nicht Zentralbank &-um (..) die große Bank ? 1175124_1181068*
%mor: *adv|nicht n:prop|Zentralbank det|die&f&sg&nom&acc adjm|groß-e n|Bank&f&sg&nom&acc ?*

All the English sentences are marked with *[- eng]*, *@s* is an indication of code-mixing. We use these two properties to transform the data into three groups: English(eng), German(ger), and Code-mixing(mix).

Segmentation Tool

MAUS(The Munich Automatic Segmentation System) uses a hybrid approach consisting of statistical classification of the signal (HMM) and probabilistic rule based components (statistically derived from a corpus). It support 21 languages therefore we can choose if we want to segment a sentence based on German or English language.
Here is a short excerpt from a segmented sentence:

intervals [1]:
xmin = 0.000000
xmax = 0.688345
text = ""
intervals [2]:
xmin = 0.688345
xmax = 0.878322
text = "x-x-x"
intervals [3]:
xmin = 0.878322
xmax = 1.198322
text = "hundreds"

Implementation

1. Transform the provided transcripts into sentences chunks. Forms we need:
    a. [start_time] [end_time] [wav index] for cutting the original big audio files;
    b. Just a sentence named by its index, for inputting into MAUS to segment;
    c. An annotated file with splited sentences to keep track of the outputs.
2. Cut the original audio files (whole conversation) into sentences based on the transformed transcripts.
3. Input the sentence audio files and its (transformed again) transcripts into MAUS, get the timemarks of the each word in the segmented sentences.
4. Extract audio features from the sentences and train using gradient boosting by pyAudioAnalysis.
    a. We only keep audios with length greater than 4 seconds. One reason is that the used features have different degrees of sensitivity to audios with different lengths, so the length of audios should be better kept less dramatic. The other reason is that most short turns are monolingual, but for this project, we are more interested in the phenomenon of code-mixing.
    b. The resulting dataset contains 316 mixed turns, 1343 German ones, and 596 English ones.
5. The dataset is then further divided into 80% training and 20% test. To get a balanced dataset, we bootstrap within each class for training and test respectively.
    a. The numbers of samples are set to be the mean of counts across classes. Finally, we have 601 samples in each class for the training part, and 150 samples for test.

Experiment

6. The experiment is designed to contain two parts. First is to train the classifier on the sentences audio files. And in this part we uses both a two-class classification of German-English, and a three-class classification of mixed-German-English.
    a. We are interested not only in the classification of mixed-German-English audios, but also in to what extent the mixed audios bring difficulties to problem.
7. The second part is to train the classifier on the word-level audio files. We want to see if a acoustic model could classify between German-English and mixed-German-English.

Performance Measures

8. We use train/text accuracy and confusion matrix to evaluate the performance of the model.

Results

Sentence level:

Gradient Boosting

German-English Classification

Test Accuracy: 65.67%

	German	English
German	111	39
English	64	86

Mixed-German-English Classification

Test Accuracy: 51.78%

	Mixed	German	English
Mixed	61	62	27
German	16	97	37
English	21	54	75

Random Forest

German-English Classification

Test Accuracy: 59.7%

	German	English
German	109	41
English	80	70

Mixed-German-English Classification

Test Accuracy: 46%

	Mixed	German	English
Mixed	51	80	19
German	29	91	30
English	15	70	65

Discussion and Challenges

We didn't finish the word-level classification due to the long runtime for training. But the results will be contained in the google drive that we store the data once the program finishes running.

As for sentence-level classification, we can see from the accuracy that the performance of the proposed model is mediocre. The accuracy is not high even for the two-class classification problem. From the confusion matrix we can see that the classifiers tend to choose German over the other two options. This might caused by the dominant language of most of the subjects are German, or by more chances of inserting German words into English sentences when code-mixing. A better normalization method might be needed and evaluated on the data.

In general the data preparation step & waiting for the model to train was painful (Thought we narrowed the scope enough but nope). Claude wrote the code for cutting data and bootstrapped the MAUS segmenting and Xinyi wrote the code for training and evaluation. Due to the large size of the data it's hard to pass around the data quickly between the two of us. We waste some time on form communication and that.

Some problems might lead to the low accuracy:

- I didn't find a proper cutting tools in the beginning, apparently ffmpeg is a commonly used tool but can only cut the audios into seconds instead of milliseconds. Later I found a library called pydub that can do the work but we decided to just use it segmenting each sentence audio file into words.
- As always, a larger dataset would probably works better (and train longer too).
- The mixed-language conversation adds difficulty in (auto) segmentation. MAUS cut phoneme based on one selected language, we chose German when segmenting the code-mixing sentences since it's the dominant language of the subjects.

For future work, it would be worthy to rerun using audio files that have other intervals of length, e.g., [0~4 sec],[4~8 sec],[8~]. Different from other type of classification on an utterance (e.g., emotion detection), cutting audio files into a same length of window frame seems unreasonable

since code-mixing is not something that happens continuously in a sentence (but maybe so in a short/pre-processed utterance?)
Another direction of future work might be looking into the difference between German-mixing-English-code and English-mixing-German-code.

Overall this exploration is not in vein, acoustic features are useful when predicting code-mixing. Incorporating more linguistics features would be another direction of future work, especially on this corpus, since its pretty well-annotated.

Reference & websites

Eppler, Eva. 1999. 'Word order in German-English mixed discourse', UCL Working Papers in Linguistics 11, 285-309. https://biling.talkbank.org/access/Eppler.html

Giannakopoulos T (2015) pyAudioAnalysis: An Open-Source Python Library for Audio Signal Analysis. PLOS ONE 10(12): e0144610. https://doi.org/10.1371/journal.pone.0144610

Niesler, T., & Willett, D. (2006). Language Identification and Multilingual Speech Recognition Using Discriminatively Trained Acoustic Models

Schiel, F. (n.d.). The Munich Automatic Segmentation System MAUS. Retrieved from https://www.phonetik.uni-muenchen.de/forschung/Verbmobil/VM14.7eng.html

Yankayis, M.: Feature Extraction Mel Frequency Cepstral Coefficient (MFCC). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.701.6802&rep=rep1&type=pdf.

Kua, J.M.K. & Thiruvaran, Tharmarajah & Nosrati, Hadis & E, Ambikairajah & Epps, Julien. (2010). Investigation of Spectral Centroid Magnitude and Frequency for Speaker Recognition. https://github.com/tyiannak/pyAudioAnalysis
https://www.phonetik.uni-muenchen.de/forschung/Verbmobil/VM14.7eng.html

Vu, N. T., & Schultz, T. (2014). *Exploration of the Impact of Maximum Entropy in Recurrent Neural Network Language Models for Code-Switching Speech*. Retrieved from http://aclweb.org/anthology/W14-3904

Yilmaz, E., Heuvel, H.V., & Leeuwen, D.A. (2018). Code-Switching Detection with Data-Augmented Acoustic and Language Models. CoRR, abs/1808.00521.