Pre-proposal: 2.4.3. Start Planning Your Capstone - DenizS [Dec20]

8.6.2 Capstone Project Proposal - DenizS [Dec 20]

1. Introduction and Motivation

Hundreds of millions of people speak a mixture of two languages, also known as code-switching. One notable example in terms of adoption is Spanglish, the mixture of English and Spanish. While tools have been used for modeling English and Spanish separately, there is currently no cloud service that I know of that leverages the data from a mixture. The overarching theme of this project will be modeling Spanglish.

2. Data

I am going to use corpora that are part of LinCE. These contain curated and annotated tweets. The entire data set is downloadable once you agree not to share it. It contains 4 corpora for Spanglish with curated annotations: One for per-token language identification (LID), one for part-of-speech (POS) tagging, one for named entity recognition (NER) with beginning and continuation labels, and another for tweet-level sentiment analysis (SA). Each corpus is divided into train, dev, and test subsets using clever stratification. All 4 corpora contain the identified language on a per-token level. Each corpus contains data from 10K+ tweets[1]. Some more details are in 03_5.2 - Data for Capstone - DenizS [Dec 20] part 1.

3. What To Build

To concretize the abstract undertaking of modeling a mixture language, my Capstone Project will involve building out the following artifacts. My approach will likely involve "traditional" ML and deep learning, and may also leverage cloud services backed by single-language models.

1. A **spelling correction tool** for Spanglish to be deployed as an API endpoint. Given a document, it will return 0 or more suggestions for correction. This can be done based on the training data that we have without the labels, so it would have the spirit of unsupervised learning. This is a per-token classification task, though the process may involve regression under the hood where we represent concepts in, say, a 300-dimensional vector space over the reals.

[1] Just the portions set aside for training have 21K+, 27K+, 33K+, and 12K+ tweets respectively, though these are not unique across the corpora.

    - To evaluate this system, we can take tweets and swap out words for their misspellings, and see how the system performs. We can use an objective function that penalizes missing misspellings and false alarms.
2. A **POS tagger** for Spanglish to be deployed as an API endpoint. For every token it will predict a POS tag. This can be trained and evaluated using the POS corpus. This is a supervised learning task. This is a per-token classification problem since there is a small number of predefined parts of speech.
    - We can evaluate the system in standard ways using the dev and test portions of the POS corpus.
3. An **SA tool** for Spanglish to be deployed as an API endpoint. This is a supervised learning task. While we can consider this a classification problem with labels negative, neutral, and positive as in our training data, it is more natural to treat this as an ordinal regression problem with the three labels corresponding to real values.
    - We can evaluate the system in standard ways using the dev and test portions of the SA corpus. 8.6.1 Benchmarking - DenizS [Dec20] pertains to this task.
4. [Stretch goal 1] An **offensive language classifier** on top of the SA tool to be deployed as an API. We can find lists of offensive words in English and in Spanish. To complement this bottom-up approach, we can send a portion of the data in a stratified manner to AMT to label it as offensive or not. Doing this at the document level, we have a binary classification task. We could also think of this as an ordinal regression task.
    - *How will we evaluate this system?* By setting aside a portion of the AMT labeled data as test data, we can have a decent enough system.
    - Caveat: I don't know what proportion of the (negative-sentiment) Lince tweets is offensive. If it's very small, I might end up needing to draw on other data sets.
5. [Stretch goal 2] A **chatbot** that draws on some of the above capabilities to interpret user input in Spanglish to navigate a menu.[2] The system would 'speak' in English, at times injecting Spanish phrases in parentheses to mirror the user's input.
    - *How will we evaluate this system?* We may want to do a user study.

4. Resource Estimation (more in progress than the other sections)

The Spanglish corpora have a total of 2.1M lines (roughly tokens), 4.8M words (including labels) and 24.9M Bytes. The whole thing easily fits into my laptop's RAM.

Out of the roughly 1.8M tokens, there are 110K unique tokens[3]. The latter can be further reduced if we treat emojis differently; Lince considers every sequence of emojis as one token.

[2] Not sure how I would deploy this, as I don't have the front-end skills to make something like this on either the web or as a phone app. But I might be able to leverage existing "chatbot as a service"s and somehow plug in my language model.

[3]
```
cat *spaeng/* | cut -f 1 | grep -v '^# sent_enum' | grep -v '^$' | \
     sort [| uniq] | wc -l
```