Milestone+Report

March 26, 2018

1 Abstract

Sitcoms can serve as a good corpus for language studies despite their relatively smaller corpus size due to more exaggerated language inherent in the format. Sitcom characters typically fall into distinguishable archetypes and character development is typically minimal. Sitcom language will tend to try and replicate current vernacular in order to sound naturalistic. Because of these features, we can predict speaking characters, performing a very basic form of author detection with small corpus size, and also model language drift.

We chose particularly the sitcom, *Friends* because we enjoyed the show, but also because it is long running and fulfills the two key features of sitcoms described above without completely abandoning naturalistic language.

We use an LSTM model to predict speaking character and use a snapshot language model to classify linguistic drift. The LSTM model gives a much better than random chance of correctly predicting character.

2 Introduction

Our underlying assumption is that sitcoms are a good case study for natural language for two main reasons. First is distinctive characters with recoginizable, often exaggerated speaking styles and verbal tics (often in the form of character catchphrases). This may enable a form of author detection using both a smaller corpus of words, and smaller sample of text. In the real world, there may be many author detection tasks that fall somewhere between trying to determine which Federalist Paper was written by Alexander Hamilton, and trying to determine whether it was Samantha, Carrie, Charlotte, or Miranda who said a particular line. Second, sitcoms will try to adhere to the current zeitgeist and popular culture, incorporating newer pieces of popular slang and phraseology into dialogue, though often in a hamfisted manner, when the characters were younger, or ironically, if the characters were older. We believe that because of this, sitcom dialogue can serve as a model for language drift in wider society.

Friends was one of the most popular tv series' in the world during its run of 226 episodes from 1994 to 2004. Because of its long run, distinctive, but not hyperbolic characters, and snappy, but still believable dialogue, we believe that the show is perfect for this type of study.

The ten year run of the show allows for some study of linguistic drift, as we believe the writers intended for the character dialogue to sound current. The distinct characters allows us to perform a character identification task. In some ways, the two tasks may interefere with each other as change in language might be due to character development rather than language drift in broader society. Character development can also make the character classification tasks more difficult.

3 Background

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts proposed a framework for studying linguistic drift in their 2013 paper, *No Country for Old Member*. In it, they studied two online communities with a total of more than 4 million posts of a median of more than 50 words, each. They created a snapshot language model for each month in the community by creating a bigram language model with Katz backoff smoothing, and then comparing each post to the model to find how surprising it's language was with the language of the community of the time by calculating corss-entropy. The paper also examined the frequency of certain types of words over time.

Dario Bertero and Pascale Fung, in their 2016 paper, *A Long Short-Term Memory Framework for Predicting Humor in Dialogues* succesfully used an LSTM model to predict the setup-punchline structure of sitcom dialogue. They also experimented with a convolutional neural network using character trigrams, word2vec, and word tokens as input features.

4 Methods

4.1 Character identification

- Modeling using word2vec, DNN, CNN and LSTM models, the folders and scripts are shown as below. We are still looking to further improve the accuracy, if it's possible.
- The baseline accuracy would be random guessing, which has an accuracy of 1/6, or 16.67%. If we get a higher accuracy in our modeling, the model is (at least somewhat) useful!
- We have attched part of the codes in this report, the complete version of all codes will be uploaded into this repository: https://github.com/aysafanxm/w266-final-project

4.2 Linguistic Drift

Our corpus is considerably smaller than in Danescu et. al, with only 28,876 lines of dialogue. Instead of using computing a monthly snapshot language model, we will instead measure drift season by season.

4.2.1 Folders

- Model folder: word2vec model and DNN model
- Cnnmodel folder: CNN model
- Rnnmodel folder: LSTM model

4.2.2 Scripts

- *handle_json.py* - The original lines.json file contains all lines from the show "Friends", which was in JSON format, we converted the character names into numbers and write them into data/feature_raw.txt. We also pick only the 6 main characters' lines (6 main characters: Ross, Rachel, Joey, Chandler, Monica and Phoebe).

- *extract_label_and_sentence.py* - Extract labels from data/feature_raw.txt and write them into data/label.txt, also extract the segmantations into data/sentence.txt.

- *extract_feature.py* - Train word vectors using word2vec (4 dimensions) and calculate the feature vectors of each sentence (take the average of the word vectors in each sentence), then write feature vectors into data/feature.txt. This process is mainly for DNN training because CNN and LSTM use embedding which doesn't train word vector the same way.
- *main_word2vec.py* - DNN with 3 hidden layers. The neuron numbers of each layer is 40, 20 and 10, respectively. The input dimension is 4 (4 features) and the output dimension is 6 (6 characters). The first 2 layers' activation function is sigmoid and the last layer's is softmax. The learning rate is 0.0001 and there are 1000 iterations. Note that there is a parameter, is_train, in the model, if is_train is True, it starts to train a new model, otherwise it takes the trained model.
- *main.py* - Similar to main_word2vec.py, but it is DNN with embedding.
- *data_helpers.py* - Helps to batch process the data
- *cnn_model.py* - CNN with an embedding layer (100 dimensions word vectos), a CNN layer, a pool layer and a softmax layer to output the probability of each label.
- *textCNN.py* - It takes the cnn_model.py to train or test the lines data. It takes 90% of the lines for training and 10% of them for testing. *The accuracy of CNN is 28%~30%.* The learn rate is 0.0001.
- *textRNN.py* - RNN with an embedding layer, a bi-lstm layer, a concat layer, a fully connected layer and a softmax layer. It takes 90% of the lines for training and 10% of them for testing. *The accuracy of RNN is 39%~40%.* The learn rate is 0.0001.

5 Results and discussion

The baseline we used for the classification is random guessing. Since there are 6 characters, the probability of correctly guessing a classification is 1/6 which is 16.67%. The accuracies for our modelings are as below:

- word2vec model and DNN model (with accuracy of 18%~19%)
- CNN model (with acuracy of 28%~30%)
- LSTM model (with accuracy of 39%~40%)

Apparently, LSTM model has the best accuracy for character classification. It is not as high as we expected, but it is still (much) higher than random guessing. The main issue of the low accuracy is that we only have fewer than 20,000 lines of sentences for the 6 main characters, and there is no way to gather more data because the show is over. Also, the lines from a show are a good representative of the natural language, but definitely not the natural enough: for example, people tend to use the same sentences over and over again in reality, but a show can't have too many same lines for a character or the audience gets bored. We are satisfied with the results so far.

6 Next Steps...

We intend to further examine the LSTM model to determine if there is a pattern to to inaccurate prediction, and see if we can improve the character identification model.

We will fully implement the Snapshot Language Model to measure linguistic drift and measure the degree to which each character differs form the baseline for every season. In addition, we will examine n-grams for interesting patterns across and between seasons.

We suspect that we may have issues with sample size for measuring linguistic drift. To compensate, we will incorporate data from other popular sitcoms that ran not too long before or after *Friends* ended. While this obviosuly will not be applicable to character identification, we can instead refine the linguistic drift models and measure drift against a baseline of a large set of television programs.