Machine Learning

Week 3 Assignment

Overview

You will use three different kinds of Natural Language Processing techniques to perform sentiment analysis on song lyrics.

User Stories

You will complete the following requirements:

1) Build feature vectors for song lyrics in `MasterSongList.json` using the following NLP techniques: * Bag Of Words * TF-IDF * Doc2Vec

2) Create classifiers that can differentiate between at least two different moods based on each of the above feature representations

3) Use concepts learned from previous weeks to try and get as high a score as possible

3) Evaluate the techniques that work the best and discuss why you think this is the case

The following advanced user stories are optional. You're not required to do these, but you will learn more from doing them:

- Use n-grams in the Bag of Words and TF-IDF representations by setting `ngram_range` in `CountVectorizer` and `TfidfVectorizer`. (You can learn about n-grams here).
- Try filtering out lyrics that are not in English.
- Explore `Doc2VecHelperFunctions.ipynb`, and try to choose different parameters for `model_vector_size`, `model_epoch_range` etc. and see how it affects your model or if you can get a better score. You can try using `GridSearchCV` to do this. (**Remember** that training the Doc2Vec model takes a lot of time!)
- Try using `KMeans` Clustering to plot your data from `CountVectorizer`/`TfidfVectorizer`/`Doc2Vec` and see if you can find any interesting clusters (**Note** This one is difficult! But here is a hint: you will first need to reduce the number of dimensions in your data using a technique like Principal Component Analysis - `PCA` in Python).

Setup

You have to install `gensim`. This is the Python library that contains the implementation of `Doc2Vec` that we will use. You can install it by typing `easy_install -U gensim` at your Terminal / Command Prompt. If needed, follow instructions at the gensim website.

Hints / Walkthrough

1) The first thing to do is to make a new dataframe that contains only those songs that actually have `lyrics_features`. This will help you map your `lyrics_features` *vectors* to your songs later on. Make sure to reset the index on this dataframe.

2) Next, you want to build a list/array containing the `lyrics_features` that you are going to be using for your NLP processing. Each item in this list must be a single `string`, containing the song lyrics for a given song. Make sure the string is correctly 'cleaned' for NLP (i.e. lower case, no stop words, etc.). Most of the lyrics are already quite clean.

- **Bag Of Words & TF-IDF**
  For `CountVectorizer` and `TfidfVectorizer`, you only need to fit them to the `lyrics_features` of those moods. For example, if you are working with `celebratory` and `sad`, then build your bag of words and tf-idf features using the `lyrics_features` from only the `celebratory` and `sad` songs. So it's probably best to build a dataframe that only contains the songs of the moods you wish to analyze, and then construct your lyrics list from that dataframe.

- **Doc2Vec**
  For `Doc2Vec`, you should use *all* the songs that have `lyrics_features`. This is because `Doc2Vec` actually builds a vocabulary, and so the more training data it has, the better. So it is best to use the dataframe you created in Step 1.

  You can generate a Doc2Vec model using the given Python Notebook `Doc2VecHelperFunctions.ipynb`. Use `%run Doc2VecHelperFunctions.ipynb` to import the notebook. You can then call `convert_lyrics_to_d2v`, passing it your lyrics list. This will generate 2 files: `all_song_lyrics.txt`, and `song_lyrics.d2v`. `song_lyrics.d2v` is your Doc2Vec model. You can load it via `model = Doc2Vec.load('./song_lyrics.d2v')`.

  Once you have `song_lyrics.d2v`, remember that the features you need to use must be indexed using the same index number as in your dataframe. **This is very important!** Because you reset the index and mapped every single song in your new dataframe in Step 1, you should be able to access the vector for your song in your model by simply using the same index as in the dataframe.

3) Finally, try different classifiers for each of your NLP models to differentiate between your chosen moods (egs: `happy` and `sad` songs) using *only* your `lyrics_features` vectors as your feature data! Try and get as high a score as possible, while using at least 500 songs for a given mood.

Tips & Notes

- Use moods that are 'opposite'. For example, try `happy` and `sad`, or `celebratory` and `sad`. Because a song can have multiple moods, and we are trying to differentiate between 2 moods, the classification will work best on moods that don't have overlapping data.
- Balance your data for better results (**Hint:** Did you know that the `RandomForestClassifier` has a `class_weight` parameter that you can set to `balanced`? Check it out! See if there is a difference in your score using it vs. manually balancing your data via `.sample`)

- Use concepts learned from previous weeks! This includes (but is not limited to) different kinds of Classifiers (`KNearestNeighbors`, `LogisticRegression`, `SVC`, `RandomForest)`, and you can try new ones that we have not covered too! Most of the syntax is always the same (egs: `.fit`, `.predict`) so you don't need to understand how a new classifier works exactly, just to try it out. Also try using `GridSearchCV`, `SelectKBest`, `SelectFromModel`, `RFE` etc. to try and improve your score.