Assignment 05 Date: 14th March, 2018, Wed Last Date of submission: 2nd April, 2018, Wed

Title: Using semantics (lexical semantics and word embedding) 20 Marks = 5*4
[updated on 16/03]

Build a text Categorization application (predicting review rating from review text) using linear and non-linear models after vectorising text and attempt to refine it further by using lexical as well as distributional semantics.

Do NOT worry about the final accuracy of the categorization application. If it does not meet your expectation, it only proves that semantics is a hard problem to solve (;-

Goal of this assignment is to get started with a real use case of semantics. The idea is to give you a taste of real life problem and solution building.

You are expected to play with text data (reviews), do some text pre-processing, use semantics & vectorised text, and then build some text classification model using binary classifier such as SVM and a non-linear model using ANN (while trying various optimization options).

For the sake of saving time, quickly freeze on particular options of ANN optimisation, stick to the same for all sub cases. 5 cases that you need to deal with:

(a) Come up with vectorised representation of text as features. You should limit your vocabulary to avoid unnecessary computation. With these vectorised representation of text as features, you should do text categorization. Note that we are NOT using semantic aspect of text here.
(b) Try to improve the above method by "EXTRACTING" LESSSR NUMBER OF FEATURES from the vectorised representation in previous step using matrix factorization method such as SVD. Ensure that you are capturing enough of variances (more than 90% ?). Note that still we are NOT really using semantic aspect of text here though we are trying hard to get more meaningful features by dimensionality reduction.
(c) Our assumption is emotion and sentiments have a direct bearing on the star rating. Derive features for each review text using Sentiment and Emotion Lexicon. The features are [sentiment, anger, fear, anticipation, trust, surprise, sadness, joy, disgust]. The sentiment column values can be sum of sentence sentiments for a review (+1, -1, 0 per sentence)and you can use textblob api as that is trained on review data. For emotion related columns, get the value using NRC Emotion lexicon. With these 9 "derived" features, you should be able to predict the rating nicely! We are using lexical semantics here.
(d) Come up with vectorised representation of every review by using only the emotion bearing words (using NRC Emotion lexicon) as we may have a better feature-set this way. This is essentially a repeat of the 1st case and the 3rd case but enclosing the lexical semantics into a vectorised model (not really distributional semantics)
(e) Derive the representative vectors for a review by using Word2Vec word-embedding. You can create your own Word2Vec distribution but your feature vector for a review should be derived from individual vectors representing each content word for that review. So you can come up with a single vector for each review as an average or sum of all the word vectors of the content words in that particular review. You can even use doc2vec which is derived from word2vec. Note that here we are using Distributional Semantics (word embedding to be specific) to derive features.

Your deliverable will be the **comparison of the performances of these two types of categorization models** for **each of the 5 approaches above**. **In total, there are 10 sets of measurements.**

Instructions

1. The review dataset is in CSV file dataset.csv
2. You will see that there are review ratings (stars) of 1-5 for each text review. You are advised to pick up the dataset only for rating 1 and 5 as that makes it a binary categorization problem. Optionally, you can also make star 1,2 -> 0 and 4,5-> 1
3. If the dataset is imbalanced, you can optionally choose same number of reviews of both the classes.
4. Your X is the text reviews and y is 'stars' for categorization problem. Ignore the other attributes for this assignment.
5. Have a pre-processing code and convert every review text as a list of tokens
6. Next, you need to do linear and non-linear modelling after doing further pre-processing in the five cases as outlined below. Getting features and pre-processing are the hard parts whereas fitting models is the easy part (;-

Doing Linear modelling

- **Use ScikitLearn library unless you are very comfortable with something else**

Doing non-linear modelling

You are advised to use Keras. If you are comfortable with tensorflow, you can use that too. Once you have pre-processed the data and ready with the X and y, you can fit in Keras or Tensorflow code

(a) You are advised to try out different options and freeze on a network configuration that you will try out in all the five cases.
(b) The optimization options can be i.e. no of hidden layers, drop out, decreasing learning rate etc.

Case 1

- Use ScikitLearn to vectorise the X (preferably tf idf)
- Find the shape of the sparse matrix so forms (its size will be the size of vocabulary). You can moderate this by using a parameter which specifies the minimum count so that your vocabulary is around 15000. Otherwise the vocabulary can be in excess of 25000.
- Now we have the pre-processed X and y
- Do train test split (0.25, 0.75)
- Train the model with cross validation (k = 10)
- Evaluate the model & print mean accuracy, std deviation (of accuracy), classification report, confusion matrix and ROC curve

Case 2

- Investigate using matrix factorization such as SVD or PCA (check TruncatedSVD) that captures more than 90% of the variances but lesser vector size. This way you

lessen the breadth of X (with lesser number of attributes). You may be able to bring down the vector size to around 2000. Take your own decision.

- Now we have the pre-processed X and y
- Repeat the steps mentioned in the previous case. Has your accuracy increased?

Case 3

- Investigate TextBlob API. It is straightforward to classify a review text using this and also to classify the text subjective or objective text. Essentially, this gives sentiment classification into +1, -1, 0 for a sentence. This gives a feature column
- Examine the NRC Emotion Lexicon (NRC-emotion-lexicon-wordlevel-alphabetized-v0.92.txt). Understand the format from the net. Using this, you can convert every review text into 8 emotion features.
- Repeat the steps in case 1 (you do not need to further limit the vocabulary here or use matrix factorization)

Case 4

- Modify your pre-processing routine so that each review text is now processed as a list of tokens that are emotion carrying or sentiment carrying using the lexicon above. Make your decision.
- Repeat the steps in case 1 (you do not need to further limit the vocabulary here or use matrix factorization)

Case 5

- Using all the review texts that you have, create your own Word2Vec distribution using Gensim API. You can also use downloadable Google Word2Vec distribution. That may work better.
- Using this word embedding, now determine the vector for each review text as the sum or average of vectors (word embedding) for all content words. You can also use doc2vec. We are here trying pure distributional semantics.
- Repeat the steps

Required background & helper code

1. **Natural Language Processing 13:** classroom material explaining Information Extraction and IE application such as sentiment, emotion mining and emotion lexicons.
2. **Natural Language Processing 12:** classroom material explaining distributional semantics, word embedding etc.
3. Classroom material explaining typical steps in ML with text : **MachineLearningWithText.pdf**
4. Scikitlearn API http://scikit-learn.org/stable/
5. Keras blog https://blog.keras.io/
6. You can **refer the notebooks made available in Google Drive**
    - **Google Drive -> code_used_in_class ->** apart from internet resources and use the class discussions.
    - You can use the helper code on calculating sentiment, emotion, converting text to vector, using Word2Vec, ML using SciKitLearn, some Keras code for ANN.
    - You are expected to at least sew together everything.