Module Code: CS3TM20

Assignment report Title: Text Mining Coursework

Student Number (e.g. 25098635): 27011203

Date (when the work completed): 08/04/2020

Actual hrs spent for the assignment: 30

Assignment evaluation (3 key points):

* Was a fun and interesting assignment, solidified concepts
* Coursework spec required lots of extra explanation, was quite ambiguous
* Good amount of online help available

Git link to code:

https://github.com/grilla99/Python-NLP

**Content outline**

**1. Task #1: NLP analysis [65 marks]**

**2. Task #2: Prediction Challenge [35 marks]**

**References and citations [10 marks]Task 1 – NLP Analysis**

* 1. Single Models

M1: Baseline Model: Lexicon Level [1][2]

* Tweet normalisation. Convert all to lowercase, remove URLS, remove usernames, remove the # in #hashtag and remove repeated characters by tokenizing words
* Uni-gram tf-idf weighting for each term
* Input tweet example:
* Being put back on hold for what has now been an HOUR is completely unacceptable.,negative
* Output: IDF – weights for each term used in the corpus provided and accuracy results

Calendar

Description automatically generatedText

Description automatically generated with medium confidence

* A 66% accuracy for this baseline model. The tf-idf weighting accurately represented sentiment well as can be seen for the emojis on the left hand side, they are all obvious indicators of sentiment and the high weighting shows that the algorithm has represented this well.

M2: Word Lemmatization – Morphological Level [3][4]

* Tweet normalisation. Convert all to lowercase, remove URLS, remove usernames, remove the # in #hashtag and remove repeated characters by tokenizing words.
* Use word net lemmatizer to get lemma of all words
* Use Count Vectoriser and TF – IDF (Bag of Words) of SK for feature extraction.
* Train a model using this data with linear models implementing stochastic gradient descent
* Input tweet example:
* I'm #MakingLoveOutofNothingAtAll on my #brandloveaffair to #LAX https://t.co/kdHRUF54sW
* Text processing complete, result: ["'m makingloveoutofnothingatall brandloveaffair lax" 'positive']
* A picture containing calendar

  Description automatically generated
* A marginal improvement on the baseline model, was expecting a greater increase. Seems to keep the meaning / morpheme of a word better than the stemmer as the stemmer was producing some strange ‘stems’ which didn’t appear to make any logical sense. As can be seen in the results quite often experienced false negative. Also experienced difficulty in identifying neutral tweets correctly, but this is known to be a difficult area of sentiment analysis.

M3: Word Stemming – Morphological Level [3][4]

* Tweet normalisation. Convert all to lowercase, remove URLS, remove usernames, remove the # in #hashtag and remove repeated characters by tokenizing words
* Use the Porter stemmer to get stem of all words
* Use Count Vectoriser and TF – IDF (Bag of Words) of SK for feature extraction.
* Train a model using this data with linear models implementing stochastic gradient descent
* Input tweet example:
* @USAirways The automated message isn't helpful and it's impossible to speak with a human right now. Desperately need our luggage :(
* Text processing complete, result: ["autom messag n't help 's imposs speak human right desper need luggag"
* A picture containing calendar

  Description automatically generated

M4: Word2Vec – Lexicon and Semantic Level [5,6,7,8,9]

* Tweet normalisation. Convert all to lowercase, remove URLS, remove usernames, remove the # in #hashtag and remove repeated characters by tokenizing words
* Use the Porter Stemmer to get stem of words
* Input data to ‘get\_word2vec’ function that I created
* 200 features, 15 epochs, builds a vocabulary and outputs the length
* Train the model, evaluate using a decision tree classifier as opposed to SGC

A picture containing calendar

Description automatically generated

M5: Part of Speech Tagging – Syntax Level and Sentiment Level [10, 11, 12]

* Tweet normalisation. Convert all to lowercase, remove URLs, remove usernames, remove the # in #hashtag and remove repeated characters by tokenizing words
* Use my pos\_tagging function which takes an array of tweets, performs part of speech tagging and then returns a new array which contains words and their relevant tags.
* This gets passed to a custom constituency parser which uses is\_adjective, is\_adverb, is\_noun, is\_verb and is\_valid to check if a token is valid.
* Uses the nltk synset to determine a score for each string (after a score for each word) to denote its sentiment
* Return a positive negative or neutral prediction
* At time of report writing was unable to get the data to work with the classification report function, please see Github for implementation in the SentimentAnalysis.py.

Bibliography

[1] – https://medium.com/analytics-vidhya/fundamentals-of-bag-of-words-and-tf-idf-9846d301ff22

[2] – https://www.mygreatlearning.com/blog/bag-of-words/

[3] – <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>

[4] - <https://www.datacamp.com/community/tutorials/stemming-lemmatization-python>

[5] - <https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-11-cnn-word2vec-41f5e28eda74>

[6] - <https://towardsdatascience.com/predicting-tweet-sentiment-with-word2vec-embeddings-67aace9b019d>

[7] - <https://www.kaggle.com/nitin194/twitter-sentiment-analysis-word2vec-doc2vec>

[8] - <https://github.com/akanshajainn/Sentiment-Analysis-Twitter-word2vec-keras>

[9] - <https://medium.com/swlh/sentiment-classification-using-word-embeddings-word2vec-aedf28fbb8ca>

[10] - <https://tanzu.vmware.com/content/blog/twitter-nlp-example-how-to-scale-part-of-speech-tagging-with-mpp-part-1>

[11] - <https://stats.stackexchange.com/questions/251310/tagging-of-tweets-using-nltk>

[12] - https://medium.com/@gianpaul.r/tokenization-and-parts-of-speech-pos-tagging-in-pythons-nltk-library-2d30f70af13b

Dataset: sentiment analysis dataset (dataset.zip, on blackboard). It includes a training

set (train.csv), a development set (dev.csv), and a test set (test.csv). Each tweet has a

sentiment label (Positive, Negative, Neutral). In the test.csv file, the correct sentiment

label of each tweet has been replaced with a symbol “?”.

* Apply NLP analysis methods of each linguistic level:
* Morphology (Lemmatization, review different types of stemmer see which is better)
  + Formed of morphemes.
  + Two types of morphemes, lexical and grammatical.
  + Stemming is the simplest form of morphological processing.
  + Stems are the base of an English word, which can be surrounded by secondary morphemes called affixes.
  + Stemming involves reducing a word to its lemma.
  + http://www.nltk.org/howto/tokenize.html
* Lexicon (Lexical analysis is dividing the whole chunk of text into paragraphs, sentences and words)
* Syntax ( Grammatical structure of sentences, helps to identify sentence structure and relationship between entities)
* Semantics ( <http://www.nltk.org/howto/semantics.html> )
* To process the input text and extract features (positive, negative, neutral)?
* Then use Logistic Regression as the classifier ( <https://www.nltk.org/api/nltk.classify.html> )
* Discuss results and effectiveness of each method for **sentiment prediction** based on the development set
* Give references and citations to the model you used
* Use td-idf weighted uni-gram bag-of-words model as baseline model

{ Text normalization:

* Tokenizing (segmenting) words 🡪 Splitting into words
* Normalizing word formats 🡪 Case folding (all to lowercase? Case may be useful for sentiment, stemming)
* Segmenting sentences