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

**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

* 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

M2: Word Lemmatization – Morphological Level

* 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']
* Calendar

  Description automatically generated

M3: Word Stemming – Morphological Level

* 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"
* Calendar

  Description automatically generated

M4: Word2Vec – Lexicon and Semantic Level

* 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

* 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.

References

[1] –

[2] –

[3] – fasd

**Baseline Model:**

(With stemming, first value ignores text after full stop – investigate)

**Lexicon Model:**

TF-IDF Weighted Unigram Bag of Words Model 🡪 What is the computational meaning of individual words in context

Word2Vec as an alternative to TF-IDF

Word2Vec – Results

Word2Vec – Write Up

**Syntax Model:**

NLTK.POS tagging

Constituency parsing, chunking

**Semantic Model:**

Word Embedding ( Follow tutorial – uses word 2 vec)

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