Abstract |

Insights gained from sentiment analysis (SA) have allowed businesses and users an opportunity to gain a deeper understanding of how communities perceive products, brands, topics, and/or services. Reddit is an American social news aggregation, web content rating, and discussion website and as of Jan 2020 according to Alexa Internet, it ranks as the fifth most visited website in the Canada and 18th in the world. For users, one advantage to using reddit is that all topics are aggregated in a sub community (known as a subreddit) where individuals may post and comment on topics of specific interest to them. By employing a generalized sentiment analysis script, we aim to mine the text to develop a generalised model that will analyse submissions and comments so that we can gain insights on what is currently trending in any subreddit in real-time. Currently there are 2 popular approaches for analyzing text, the first utilizes a bag of words method and applies TF-IDF while the second employs a shallow 2-layer neural network that are trained to reconstruct the contexts of words known as words2vec. For the scope of this project, our methodology is as follows; 1) Utilise Reddit’s api to scrape subreddit submission titles and comments 2) pre-process (stop words, symbols, apostrophes, lemmatize ) and tokenize text 3) upload the data to a MongoDB database 4) Apply TF-IDF, words2vec, and sentiment analysis models on the pre-processed text 5) Deploy streaming version to get live updates of subreddits.

Introduction |

With the dramatic shift and prevalence of social media in modern society, text analytics can be a powerful tool for companies to take strategic action based on valuable insights on common themes and trends found on the internet. To make text analytics the most efficient, organisations can use text analytics software, leveraging machine learning and natural language processing algorithms to find meaning in enormous amounts of text. Reddit is an American social news aggregation, web content rating, and discussion website and as of Jan 2020 according to Alexa Internet, it ranks as the fifth most visited website in the Canada and 18th in the world. For users, one advantage to using reddit is that all topics are aggregated in a sub community (known as a subreddit) where individuals may post and comment on topics of specific interest to them. Therefore, there is a wealth of commentary, opinions, and knowledge that can be mined from the submissions and comments of reddit that can have the potential to provide valuable insight to any company and/or topic of interest. For the scope of this project, our methodology is as follows; 1) Utilise Reddit’s api to scrape subreddit submission titles and comments 2) pre-process (stop words, symbols, apostrophes, lemmatize ) and tokenize text 3) upload the data to a MongoDB database 4) Apply TF-IDF, words2vec, and sentiment analysis models on the pre-processed text 5) Deploy streaming version to get live updates of subreddits.

Literature Review

1) Řehůřek, R. (2019, November 1). Gensim Topic Modelling for Humans Core Concepts. Retrieved from https://radimrehurek.com/gensim/index.html

This webpage documents the free python package known as gensim and explores key concepts about its usage and required understanding of terms such as documents (text), corpus (collection of 0documents), vector (mathematical representation of corpus, and models (algorithm employed to transform vectors from one representation to another. It also provides a walkthrough and example for users to get started using genism as well as providing api references.

2) Jurafsky, D. and Martin, J.H., “Vector Semantics and Embeddings,” in Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd ed. Stanford University, UK: Online, 2019, pp. 94–122.

The chapter “vector semantics and embeddings” explains 5 main topics which include (1) the various methods to represent words, (2) ways to measure similarity, (3) TF-IDF model, (4) Word2vec, and (5) the properties of embeddings. Additionally, the concept of information retrieval is introduced which is used to find documents in a collection that best matches a query from a term-document matrix. For its application, TF-IDF is a useful model to give weight to terms in a given vector, and it is dependent on the term frequency and its frequency of appearance in documents, such that words that are useful for describing documents from rest of collection are given more weight, while those that occur frequently are less. Therefore, TF-IDF is good for weighting co-occurrence matrices for IR.

In addition to representing words in a term-document matrix, word-word matrices enable the identification of n-grams around a target word and by using cosine similarity it is possible to determine if a word is similar to another pair based on its frequency. Furthermore, to identify collocations it is practical to use pointwise mutual information to calculate the probability of 2 words (x and y) occurring compared to what we would expect if they were to appear independently.

Alternatively, rather than asking how often each word occurs near a target word as with TF-IDF. It is possible to ask using the Word2vec model, how likely a word is to appear a target word. This model is a binary classification trained on a logistic regression classifier which uses and algorithm known as skip-gram. Skip-gram is a shallow neural network that treats the target word and each neighboring word as a positive example while randomly sampling other words to get negative samples. This method uses a logistic regression model to train classifier to distinguish the 2 cases and uses the regression weights as embeddings.

3) Hutto, C.J. & Gilbert, Eric. (2015). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014.

Highlights the development, evaluation, and validation of VADER (for Valence Aware Dictionary for sEntiment Reasoning). Sentiment analysis was previously heavily reliant on pre-existing manually constructed lexicons where each lexical feature was labelled manually according to their semantic orientation. The article identifies 3 sentiment oriented (polarity-based) lexicons, which include LIWC, General Inquirer (GI), and Hu & Liu and notes that the common disadvantage for using them is that they are unable to discern sentiment intensity of words and an inability to parse sentiment-bearing lexical items such as slang or emoticons from social text. The authors construct and empirically validate a gold standard list of lexical features (along with their associated sentiment intensity measures) which are specifically attuned to sentiment in microblog-like contexts.

4) TextMiner (2014, January 17). Text Mining Online: NLP. Retrieved from https://textminingonline.com/

TextMiner provides an in depth walk through using python’s famous natural language processing package known as NLTK. The website contains a series of articles that introduce key concepts in the preprocessing of text such as word tokenization, parts-of-speech-tagging, and stemming and lemmatization all within the python environment.

5) MonkeyLearn. (2020). Text Analysis. Retrieved from https://monkeylearn.com/text-analysis/

Text analysis plays an important role in how the people and businesses gain valuable insights on an information saturated world. Monkeylearn’s article introduces the topic outlining the various methods of text analysis which includes basic techniques such as word frequency, collocation, and concordance. Furthermore, the authors also describe a wide array of advanced methods such as text classification, sentiment analysis, topic analysis, intent detection, text extraction, keyword extraction, entity recognition, word sense disambiguation, text clustering.

6) Turney, P.D. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL 2002

The author introduces a simple unsupervised learning algorithm primarily for rating reviews as a thumbs up or down (recommended or not recommended). This algorithm takes reviews and extracts sentences that contain adjectives or adverbs, then estimates its semantic orientation through pointwise mutual information – information retrieval and classifies the review based on the average semantic orientation of the phrases.

**Dataset**

The dataset used is an on-demand stream of subreddit submission titles and comments obtained from reddit. Using PRAW: The Python Reddit API Wrapper, we can pull submission titles and all comments of “n” hottest submissions from the front page and store the data in a mongoDB database (schema-less NoSQL document database).

Within the database, we can have multiple collections each representing a submission within the subreddit and all the comments stored as documents specific to that submission. Furthermore, we have an additional collection called “[subreddit name]\_overview” which includes information about all submissions we scraped which includes their title, score, id, url, number of comments, and content body. Lastly, we have 3 collections that are similar in that they contain an aggregation of comments from all submissions and includes information such as the message, submission title, and timestamp. However they differ in that one contains text that has not been preprocessed, while another has text that has gone through the preprocessing step (refer to approaches section for details on preprocessing) but is not tokenized, and the third has both been tokenized and preprocessed.

For the scope of this project, we intend to scrape 2 subreddits as an example and take the top 20 hottest (due to computing and time constraints) submissions and its comments.

**Approach**

1. Build Corpus
   1. Using PRAW, select subreddit and scrape current N hottest submissions/comments
   2. Store submissions and comments into MongoDB.
2. Text Preprocessing
   1. Extract and aggregate all comments from MongoDB into one data-set
   2. Remove symbols, spaces, punctation, and digits (except <’>)
   3. Lower case all comments
   4. Tokenize words
   5. Find and replace words in our “apostrophe” dictionary which are a set of words that can be converted from their contracted form. Additionally, any shorthand and abbreviations can be corrected here.
   6. Apply parts-of-speech (POS) tagging to all words
   7. Lemmatized tokenized text based on their POS tag
   8. Remove stop words, the resulting dataset is now tokenized and preprocessed. Add features, submission title (the submission the comment was extracted from) and timestamp (date/time it was posted). Upload new dataset into MongoDB.
   9. Join all words with a space where there used to be a comma, to un-tokenize the comments. Add features, submission title (the submission the comment was extracted from) and timestamp (date/time it was posted). Upload new dataset into MongoDB.
3. Build Features (Word Frequency)
   1. Using pre-processed and tokenized dataset, we count the frequency of each word and filter words that appear less than 2 times under the assumption, words that appear once are unimportant.
   2. Export unique words and word count to a .csv
4. Build Script to generate bigrams and trigrams from entire corpus and query subset
   1. Load dataset (either full tokenized corpus, or tokenized query subset)
   2. Using the NLTK package we use TrigramCollocationFinder and BigramCollocationFinder on our list of words
   3. Generate frequency table based on the results of the trigram/bigram finder.
   4. Find bigrams-trigrams based on pointwise mutual information to calculate the probability of 2 words (x and y) occurring
   5. Generate PMI table of top bigram/trigram candidates
   6. Find bigrams-trigrams based on Chi squared test to calculate the likelihood that words are independent
   7. Generate Chi squared table of top bigram/trigram candidates
5. Information Retrieval (Word embeddings with TF-IDF)
   1. Using genism, convert corpus into bag-of-words vector (BOW-V)
   2. Create a TF-IDF model to transform BOW-V into a vector space where the frequency counts are weighted according to the relative rarity of each word in the corpus.
   3. To prepare for similarity queries, we index the transformed TF-IDF model.
   4. Receive query from user, tokenize, and convert into (BOW-V) then query the similarity of our BOW-V document against every document in the corpus.
   5. Returns the top documents containing our query and we generate and export a dataframe with the document ID, similarity score, preprocessed comment, and unprocessed comment. This is now a subset of our data-set that is specifically related to our query.
6. Build Sentiment Analysis Model from entire corpus and query subset
   1. Load dataset (either full tokenized corpus, or tokenized query subset)
   2. Using vaderSentiment for python, we build a sentiment analyser
   3. Run sentiment analyser on each message to determine what the breakdown of each comment is rated as and the overall sentiment of the comment.
   4. Based on documentation provided by Hutto, C.J. & Gilbert, Eric, compound scores dictate whether a comment is considered positive, neutral, or negative. Using these thresholds for positive sentiment: (compound score >= 0.05), neutral sentiment : (compound score > -0.05) and (compound score < 0.05), and negative sentiment : (compound score <= -0.05).
   5. When run on our query subset, we have identified what the sentiment for each comment related to our query is.
   6. (Optional) When run on our entire corpus, we create training data to develop our own sentiment analyser using machine learning algorithms.
7. Word Similarities Using Word2vec.
   1. Load our preprocessed and tokenized corpus and using genism, we import their Word2Vec function.
   2. We apply Word2Vec on our messages and query our transformed model
   3. Returns words that are most likely to return with our target word