Data Acquisition

Text String and documents have been provided with the assignment and are being used as such. The iris data set which is widely available is loaded and used using sklearn’s dataset library for Q4.

NLP Overview

Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken and written -- referred to as natural language. It is a component of artificial intelligence (AI).

NLP has existed for more than 50 years and has roots in the field of linguistics. It has a variety of real-world applications in several fields, including medical research, search engines and business intelligence.

Q1 – Tokenization Methodology

For Q1, we are using NLTK library for various tokenizing the given sentence. Various tokenization we used are as follows:

1. Word Tokenization - Word tokenization is the process of splitting a large sample of text into words. This is a requirement in natural language processing tasks where each word needs to be captured and subjected to further analysis like classifying and counting them for a particular sentiment etc. We used word\_tokenize() function to tokenize to individual words.
2. Sentence Tokenization - Sentence tokenization is the process of splitting text into individual sentences. The sentence tokenizer will not split an individual word, so the offending text, in replacement form, is preserved intact during the tokenization process. After generating the individual sentences, the reverse substitutions are made, which restores original text in a set of improved sentences. We used sent\_tokenize() function to tokenize the sentences.
3. Tweet Tokenization - Twitter is a social platform that many interesting tweets are posted every day. Tweets are more difficult to tokenize compared to formal text because they may include smileys and emojis, etc. Therefore, tweet tokenizer is used to tokenize the tweet sentences. It can preserve the emojis and come with many handy options. We used TweetTokenizer() to tokenize as per tweets.
4. MWE Tokenizer – It takes a string and merges multi-word expressions into single tokens, using a lexicon of MWEs. For example, name “Johanna Beach” is person’s name, and it is separated into two tokens. We can avoid this and merge them in one expression using MWETokenizer() function providing the required words to be considered as one.
5. Regexp Tokenizer – This tokenizer splits a string into substrings using a regular expression. For example, getting the tokens out of the alphabetic sequences, money expressions, or any other non-whitespace. In our assignment, we identified the numbers in the string using RegexpTokenizer() function.
6. TextBlob Word Tokenize - TextBlob is a Python library for processing textual data. Using it we can easily perform many common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. In our scope, we used blob.words() and blob.sentences() function to get word tokenize and sentence tokenize.
7. List noun\_phrases - With the help of TextBlob.noun\_phrases() method, we got the noun phrases of the sentences by using TextBlob.noun\_phrases() method. After that we printed the nouns found in the one by one.
8. Determine sentiment.polarity - Sentiment polarity for an element defines the orientation of the expressed sentiment, i.e., it determines if the text expresses the positive, negative, or neutral sentiment of the user about the entity in consideration. We used SentimentIntensityAnalyzer() to get the polarity scores of the string given.

Business Understanding

Businesses use massive quantities of unstructured, text-heavy data and need a way to efficiently process it. A lot of the information created online and stored in databases is natural human language, and until recently, businesses could not effectively analyze this data. This is where natural language processing is useful.

With the improvements in deep learning and machine learning methods, algorithms can effectively interpret them. These improvements expand the breadth and depth of data that can be analyzed for business goals.

NLP Tokenization

By tokenizing, you can conveniently split up text by word or by sentence. This will allow you to work with smaller pieces of text that are still relatively coherent and meaningful even outside of the context of the rest of the text.

It’s the first step in turning unstructured data into structured data, which is easier to analyze. When you’re analyzing text, you’ll be tokenizing by word and tokenizing by sentence. For tokenizing, we are using NLTK library.

Q2 – Bag of Words Methodology

A bag of words is a representation of text that describes the occurrence of words within a document. We just keep track of word counts and disregard the grammatical details and the word order. It is called a “bag” of words because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

Step 1: Convert the above sentences in lower case as the case of the word does not hold any information.

Step 2: Remove special characters and stop words from the text. Stop words are the words that do not contain much information about text like ‘is’, ‘a’, etc.

Step 3: Go through all the words in the text and make a list of all the words in our model vocabulary.

Q3 – TF-IDF Methodology

Term Frequency Inverse Document Frequency (TFIDF) analysis is one of the simple and robust methods to understand the context of a text. Term Frequency and Inverse Document Frequency is used to find the related content and important words and phrases in a larger text. Implementing TF-IDF analysis is very easy using Python. Computers cannot understand the meaning of a text, but they can understand numbers. The words can be converted to numbers so that the relationship between them can be understood.

1. Term Frequency

The term is frequency measure of a word w in a document (text) d. It is equal to the number of instances of word w in document d divided by the total number of words in document d. Term frequency serves as a metric to determine a word’s occurrence in a document as compared to the total number of words in a document. The denominator is always the same.

TF = (number of instances of word w in document d)/ (total number of words)

1. Inverse Document Frequency (IDF)

This parameter gives a numeric value of the importance of a word. Inverse Document frequency of word w is defined as the total number of documents (N) in a text corpus D, divided by the number of documents containing w.

IDF = log (total number of documents in text corpus/number of documents containing w)

1. Term Frequency Inverse Document Frequency (TF-IDF)

The product of TF and IDF is the TF-IDF. TF-IDF is usually one of the best metrics to determine if a term is significant to a text. It represents the importance of a word in a particular document.

TF-IDF = TF \* IDF

Q4 – Data Processing Techniques Methodology

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. We are doing below data processing techniques:

1. Decimal Scaling
2. Decimal scaling is a data normalization technique. In this technique, we move the decimal point of values of the attribute. This movement of decimal points totally depends on the maximum value among all values in the attribute.
3. A value v of attribute A can be normalized by the following formula:

Normalized value of attribute = (vi / 10j)

1. Min-Max Normalization
2. We used MinMaxScaler() for making Min-Max normalization of our X dataset.
3. MinMaxScaler transforms features by scaling each feature to a given range. It scales and translates each feature individually such that it is in the given range on the training set, e.g., between zero and one. Formula:

X\_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

X\_scaled = X\_std \* (max - min) + min

1. Z-score Normalization
2. We used StandardScaler() for making z-score normalization of our X dataset.
3. StandardScaler standardizes features by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as: z = (x - u) / s. Use this if the data distribution is normal.

Importance of Tokenization in NLP

Before processing a natural language, we want to identify the words that constitute a string of characters. That’s why tokenization is a foundational step in NLP. This process is important because the meaning of the text can be interpreted through analysis of the words present in the text. Tokenization is the process of breaking apart original text into individual pieces(tokens) for further analysis. Tokens are the pieces of the original text; they are not broken down into a base form. Therefore, appropriate tokenization technique is used to get best results out of the natural language.

Word level or sentence level tokenization or any other form of tokenization level is chosen depending on the requirements and source of the text data.

Data Processing Techniques Comparison

Min-Max Normalization has advantage over z-score normalization when the data set doesn’t have outliers. And, it guarantees all features will have the exact same scale. Z-score normalization handles outliers better but does not produce normalized data with the exact same scale. Decimal Scaling normalization benefits are connected to information base, as the information bases become lesser in size, the methods or any function will go through the information very quick and more limited in this way improving reaction time and speed. However, on comparison with z-score and min-max normalization, decimal normalization is not bringing any further benefits for non-homogeneous dataset. Therefore, z-score and min-max normalization are more appropriate for practical purposes. However, if your dataset has a lot of outliers then use z-score otherwise use min-max normalization.

Conclusion

A model in NLP is a probabilistic statistical model that determines the probability of a given sequence of words occurring in a sentence based on the previous words. These models have probability distribution over sequences of words. To bring out the best of the model, we need continuous improvements to the token level(word or sentence or any other) as per the requirement. Choosing TF-IDF or word of bag, also depend on document size. Data pre-processing is also vital to convert the data to a normalized form, which greatly affects the improvement of the NLP model. All these choices on methodology can improve or degrade the NLP model. Hence, the correct combination of these have become very essential for the NLP model output success.

Bag of Words and TF-IDF Comparison

Bag of Words just creates a set of vectors containing the count of word occurrences in the document, while the TF-IDF model contains information on the more important words and the less important ones as well. One of the major limitations of bag of words is that if we deploy bag of words to generate vectors for large documents, the vectors would be of large size and would also have too many NULL values leading to the creation of sparse vectors. Also, bag of words doesn’t bring in any information on the meaning of the text. On the other end, TF-IDF is more comprehensive to get value of tokens proportionally increasing the number of times a word appears in the document but is counterbalanced by the number if document in which it is present. Limitation of TF-IDF is that vectorization it creates doesn’t bring contextual meaning of the words.

Therefore, we can say that for small documents we can use Bag of Words, but we should use TF-IDF when we have large documents.