Assignment - 1

1. Given the following raw text data: Raw Text: "I looooovvvve this proooooduct!!! It's AmAAAAzingggg!!! #a@hAppy/customer" The cleaned text data is as follows: Cleaned Text: "i love this product amazing happy customer" Please list the text data cleaning methods used here. Do you know any other text-cleaning methods? Please list at least one text cleaning method. What is the purpose of text data cleaning?

The text cleaning methods that have been used to clean the raw text given are explained below.

1. **Repetition letters in words:** The words “love”, “amazing” and “product,” which contain repeated letters, are reduced to their simplest forms.
2. **Special Character cleaning:** Special characters like “@” and “#” are removed from the text.
3. **Upcase/downcase:** All the letters in the text are converted to lowercase. For example, the A’s in the “AmAAAAzing” and “I” are converted to lowercase.
4. **Punctuation Sign:** The punctuation signs like “?”, “!” and “;” are removed from the text. For example, the exclamation marks are removed after the word “AmAAAAzing” and “proooooduct”.
5. **Stop Words:** The stop words are the words that are commonly used and can be removed from the sentence, they do not change the meaning of the sentence, and this will not change the prediction power. They can be removed before data pre-processing. For example, “it” and “a” have been removed from the sentence.

Other text cleaning methods:

1. **Stemming**: It is the process of stripping prefixes and suffixes from words to return them to their original form. The resulting stem might not always be a meaningful word or might not always mean what the original word meant. For example, the words like “jumping” are optimized to “jump”.
2. **Lemmatization**: Words are simplified or reduced to their dictionary form, or lemma, through lemmatization. The resulting lemma is always a phrase that has a dictionary definition. For example, the word “went” is reduced to “go”.

Purpose of Text Cleaning:

Text data cleaning, sometimes referred to as text preprocessing, is the process of converting unclean, unstandardized text data into a format that may be used more successfully for analysis, modeling, and other natural language processing (NLP) applications.

It is important to preprocess the text, this helps in standardizing the text and reducing noise which can increase the model's ability to forecast the results accurately.

2. What is the likelihood ratio test for phrase detection? How does it work?

Firstly, hypothesis testing is a method that helps is used to divide large amounts of data into "yes" or "no" conclusions. The data samples are collected and modeled from random distributions and the outcomes are unstable.

The likelihood ratio was proved successful for collocation extraction or phrase detection.

Likelihood ratio test for phrase detection:

The Likelihood ratio test is a statistical model that is used for phrase detection, that identifies the frequently used phrases by calculating the probability or likelihood of the words forming that phrase.

For each given pair of words, the method assesses two hypotheses on the dataset.

Null Hypothesis: word 1 appears independently from word 2 (There is no connection between seeing word 1 and seeing word 2)

Alternative Hypothesis: Two words form a common phrase.

Hence, the likelihood ratio test for phrase detection asks the following question:

“Is it possible that the observed word occurrences in each text corpus were produced by a model in which the probability of the two words was entangled or in which they occurred independently?”

We can express the null hypothesis Hnull (independent) as $P(w2 \mid w1) = P(w2 \mid \text{not } w1)$, and the alternate hypothesis Halternate (not independent) as $P(w2 \mid w1) \neq P(w2 \mid \text{not } w1)$.

The final statistic is the log of the ratio between the two:

$\log \lambda = \log L(\text{Data; Hnull}) / L(\text{Data; Halternate})$.

The probability of observing the word frequencies in the dataset under the independent or not independent model for the word pair is represented by the likelihood function L(Data; H).

Now, we make one more assumption about the generation process of the data. We assume that the data is generated using the binomial distribution model, where special words are inserted if the coin outcome is head, or a random word is inserted otherwise.

The algorithm for applying likelihood ratio test analysis to identify frequent words is as follows:

1. Determine the probability that each singleton word will occur using P(w).

2. Determine the conditional pairwise word occurrence probability P(w2 | w1) for each unique bigram.

3. Determine the log of the likelihood ratio for each unique bigram.

4. Sort the bigrams according to the likelihood ratio.

5. Choose the bigrams with the lowest likelihood ratio values as features.

3. How are data vectors and feature vectors represented in bag-of-words featurization for text documents?

4. What is quantile binning? Why do we need it?

A method of "binning," also known as "quantizing," divides counts into various "bins" that omit count values. We must choose the width of each bin before quantizing the data.

There are typically two types:

1. Fixed Width binning: Simply taking data and arranging it into buckets within a specified range. The range can be customized according to the requirements.

2. Adaptive binning: It analyses data more on the bounds of quantiles or deciles. An effective method for adaptive binning is quantile-based binning.

Quantile Binning:

According to the distribution of the data, a technique called quantile binning is employed to convert continuous numerical data into discrete bins.

Quantiles are used to divide the data into equal parts. For example, a median split the data into two halves, with the first half less than the median and the next half higher than the median. Similarly, quartiles split the data into quarters. The deciles split the data into tenths.

Example: Binning counts by quantiles

 # Continue example 2-3 with large_counts

>>> import pandas as pd

 # Map the counts to quartiles

>>> pd.qcut(large_counts, 4, labels=False)

array([1, 2, 3, 0, 0, 1, 1, 2, 2, 3, 3, 0, 0, 2, 1, 0, 3], dtype=int64)

 # Compute the quantiles themselves

>>> large_counts_series = pd.Series(large_counts)

>>> large_counts_series.quantile([0.25, 0.5, 0.75])

0.25 122.0

0.50 926.0

0.75 8286.0

dtype: float64

Need for Quantile Binning:

1. It identifies several patterns in continuous variables that are simple to decipher and comprehend.
2. It splits the data into intervals of equal size so that each bin has roughly the same number of observations.
3. To manage skewed distributions and lessen the influence of outliers, we need quantile binning.
4. We can present the original continuous data in a more manageable and understandable form by grouping the data into bins.
5. When utilizing categorical or discrete input machine learning techniques, it can be extremely helpful.

5. What are the limitations of bag-of-words in capturing semantic meaning in text data, and how can these limitations be alleviated?

A bag of words is a concept that shows the word counts of the words that appear in a text document. It is referred to as a "bag" of words because any details regarding the arrangement or structure of the words within the document are not given, it only focuses on the existence of the words in the document.

A written document is transformed into a vector of counts during bag-of-words (BoW) featurization. Vector has items for most of the vocabulary terms. A sentence counts by the number of that word if it appears in the vocabulary section. If the word is not present, the count is zero. The Bow feature only shows the number of times a Word appears in Text; it does not care much about the sequence of the words.

Limitations of bag-of-words in capturing semantic meaning in text data:

1. A statement's semantic significance might be lost if it is condensed into a single word. By breaking the whole sentence into tiny parts, there is a chance of losing the actual intention and meaning of the sentence.
   For instance, "not bad" can also signify "decent" or "good" in a figurative sense. On the other hand, when "not" and "bad" are combined, a floating negation and a terrible

attitude are the results. This is one of the limitations of the bag-of-words in capturing the semantics of the text.
2. Even though the bag-of-words heuristic is straightforward and practical, it is crucial to remember that it is not a reliable indicator of the text's semantic meaning.
3. Word order is lost when using a bag-of-words representation, which might result in the loss of crucial contextual information.
4. As the number of distinct words in the corpus increases, the size of the feature space also increases, potentially resulting in high-dimensional and sparse representations.
5. Bag-of-words ignores the relationships between words and their meanings in favor of treating each word as a separate characteristic.
   With the singletons "toy" and "dog," the meaning is lost because "toy dog" and "dog toy" can indicate completely different things.

Methods to Alleviate the Limitations:

1. Bag-of-n-Grams: N-grams capture sequences of n-words rather than individual words, helping to maintain some local word order information. It somehow retains the original structural sequence of the text.
2. Term frequency-inverse document frequency: We can use TF-IDF, which assigns weights to all the words in the text document based on the frequency of the words.
3. Language models: New developments in natural language processing, such as transformer-based models like BERT or GPT, can better capture the semantic meaning and contextual meaning of the text.

6. How are word count statistics used in tasks such as document classification and information retrieval, and why are word-level features effective indicators of topic content?

7. What is the purpose of using the log transform in supervised learning, and how is it evaluated?
   Firstly, log transformation is a statistical technique that is used to transform the variables by applying a logarithmic function to all the variables.

Purpose of Log Transformation in Supervised Learning

1. The log transform is often applied to the target variable in supervised learning tasks when the target variable is skewed, or its distribution is not approximately normal.
2. It also reduces the impact of extreme values and makes the distribution more symmetric.
3. By taking the logarithm of the target variable, the scale of the data is compressed, and the values are spread out. This transformation can help in stabilizing the variance and improving the linearity between the features and the target variable.
4. It can be particularly useful when the target variable spans several orders of magnitude or exhibits a long tail in the distribution.

5. In supervised learning, some feature-target relationships may not necessarily be linear. By calculating the logarithm of the target variable, we may be able to transform the relationship into a linear one or make it more suited to linear modeling methods.
6. If we have a very broad range of data, smaller numbers can be overwhelmed by larger ones. By using the log of each variable, it is possible to improve the comprehension of the visualization.

Evaluation of Log Transformation

1. Firstly, we can evaluate the models with and without log transformation. We can compare the model metrics and performance indicators to assess the models with and without log transformation.
2. Secondly, The evaluation of the log transform can be performed by comparing the statistical properties of the transformed data with the original data. This includes assessing the normality of the distribution, homoscedasticity, and the relationships with other variables.
3. Thirdly, The models are evaluated using the R-squared score, which indicates how well a trained regression model predicts new data. A good model will have a high R-squared value. The perfect model obtains the highest possible score of one. A negative score is possible, and a defective model can have an arbitrarily low negative score. Cross-validation gives us not only a score estimate but also a variance, allowing us to determine whether the differences between the two models are significant. Apart from the R-squared score, other evaluation metrics can also be used to like F1-score, recall, precision or mean squared error for a trained model.

8. How does the Box-Cox transform compare to the log transform in compressing the tail of the distribution?

9. What are scaling/normalization and transformation? What is the biggest difference between scaling/normalization and transformation?