Word Representation

By:

Bayat Vahabodin

Contents

Introduction 1
Data Preparation 1
Feature Extraction 2
  Bag of Words (BoW) 2
  N-gram 2
  Term Frequency-Inverse Document Frequency (TF-IDF) 3
  Continuous Bag of Words (CBOW) 3
  Skip-gram 3
Modeling 3
Evaluation 4
Conclusion 5

Tables

Table 1: Accuracy of results 4

Introduction

Word representation in natural language processing refers to the process of transforming words into a numerical form that can be used as input to machine learning models. This is necessary because machine learning algorithms work with numerical data and cannot handle raw text directly.

To classify Amazon reviews into five categories (1-5), different textual features can be used to represent the words in the reviews. Here are some common techniques for word representation:

Data Preparation

We loaded Amazon reviews dataset from a CSV file named "Reviews.csv" into a Pandas DataFrame. It then uses the train_test_split() function from Scikit-learn library to randomly split the amazon_data DataFrame into two subsets, namely train_data and test_data, with a 80/20 split ratio. The random_state parameter ensures that the randomization used for the splitting is deterministic, so that the same split can be reproduced if the code is run again with the same random_state value.

We used the Natural Language Toolkit (nltk) library to perform text preprocessing on Amazon reviews data for the purpose of text classification. We downloaded necessary data and libraries using the nltk.download function.

At first, we download and install three datasets: punkt, stopwords, and wordnet. These datasets are necessary for performing text preprocessing tasks such as tokenization, removing stop words, and lemmatization.

Next, we defined two important objects: stop_words and lemmatizer. stop_words is a set of common words that are often irrelevant to the meaning of a sentence and can be safely removed without losing important information. lemmatizer is an object from the WordNetLemmatizer class, which is used to reduce words to their base or dictionary form.

We defined a function to apply text preprocessing techniques to the input text. The function performs the following tasks:

- ✓ Converts text to lowercase to standardize the text.
- ✓ Removes punctuation from the text using the translate function and the string.punctuation constant.
- ✓ Tokenizes the text into a list of individual words using the word_tokenize function from the nltk library.
- ✓ Removes stop words from the text using the stop_words set defined earlier.
- ✓ Lemmatizes each word in the text using the lemmatizer object defined earlier.
- ✓ Joins the words back into a string.

The function returns the preprocessed text as a string.

Finally, the clean_text column is added to both the train_data and test_data by applying the preprocess_text function to the Text column of each dataframe using the apply method. The resulting clean_text column contains the preprocessed text for each review.

Feature Extraction

Bag of Words (BoW)

In this technique, the occurrence of each word in the review is counted and a vector is created where each element corresponds to a unique word in the review. The value of each element in the vector is the count of the corresponding word in the review. The resulting vector is typically very high-dimensional, with the dimensionality equal to the size of the vocabulary.

N-gram

An N-gram is a sequence of N contiguous words in a sentence or text. N-grams can be used to capture the context in which a word appears by creating a vector for each N-gram that occurs in the text. The resulting vectors can be used as features for text classification.

Term Frequency-Inverse Document Frequency (TF-IDF)

This technique is similar to BoW, but it takes into account the importance of each word in the review and across the entire corpus of reviews. The TF-IDF score of each word is calculated by multiplying its frequency in the review (TF) by its inverse frequency in the corpus (IDF). The resulting vector is typically lower-dimensional than BoW and can help reduce the impact of commonly used words in the classification task.

Continuous Bag of Words (CBOW)

This is a neural network-based model for learning word embeddings. In CBOW, the goal is to predict the center word of a context window of surrounding words. The context window can be defined as a fixed number of words to the left and right of the center word. The resulting word embeddings can be used as features for text classification.

Skip-gram

Skip-gram is another neural network-based model for learning word embeddings. In skip-gram, the goal is to predict the surrounding words given the center word. The resulting word embeddings can also be used as features for text classification.

Modeling

Logistic regression

Logistic regression is a popular machine learning algorithm used for binary classification tasks. It is also commonly used for multi-class classification tasks.

The goal of logistic regression is to find the relationship between the input features and the output variable, which is a categorical variable in our case (the five categories).

Support Vector Machines

SVM widely used for classification tasks, including text classification. The goal of SVMs is to find a hyperplane that maximizes the margin between the different classes. The margin is the distance between the hyperplane and the closest data points from each class. The closest data points from each class are called support vectors, and they determine the location and orientation of the hyperplane.

Evaluation

It is important to evaluate the performance of the model to ensure that it is accurately classifying the data. In the context of text classification for Amazon reviews, it is crucial to evaluate the performance of the model on both the training dataset and the test set.

To evaluate the performance of the model on the training dataset, we used confusion matrix, accuracy, and F1-score metrics. Accuracy measures the proportion of correctly classified reviews, while F1-score measures the balance between precision and recall. Precision measures the proportion of true positives among all positive predictions, while recall measures the proportion of true positives among all actual positives. By evaluating the model on the training dataset, we can get an idea of how well the model is performing on the data that it has been trained on.

However, evaluating the performance of the model on the training dataset alone is not enough. We also need to evaluate the model on the test set to see how well it performs on new unseen data. To do this, we can use the same metrics that we used to evaluate the performance on the training dataset. By evaluating the model on the test set, we can get an idea of how well the model is able to generalize to new data.

The below table shows the results of evaluating different text representation techniques using Logistic Regression (LR) and Support Vector Machines (SVM) algorithms on the training and test datasets.

Table 1: Accuracy of results

Feature	LR Train	LR Test	SVM Train	SVM Test
BOW	87.38%	71.5%	63.12%	68.5%
N-Gram	96.38%	69%	100%	68.5%
TF-IDF	100%	71.5%	63.12%	68.5%
CBOW	63.12%	68.5%	63.12%	68.5%
Skip Gram	63.12%	68.5%	63.12%	68.5%

Conclusion

Based on the results in the table, there are some differences in the performance of Logistic Regression (LR) and Support Vector Machines (SVM) algorithms on the training and test datasets across different text representation techniques.

For LR, the performance on the training dataset is generally better than on the test dataset. For instance, on the BOW and n-gram techniques, the LR model achieved an accuracy of 87.38% and 96.38%, respectively, on the training dataset, while the accuracy dropped to 71.5% and 69%, respectively, on the test dataset. Similarly, for the TF-IDF technique, the LR model achieved a perfect accuracy of 100% on the training dataset, but its accuracy dropped to 71.5% on the test dataset. This indicates that the LR model may be overfitting the training data.

For SVM, the performance on the training dataset is also generally better than on the test dataset, except for the n-gram technique where the SVM model achieved a perfect accuracy of 100% on the training dataset and an accuracy of 68.5% on the test dataset. For other techniques, such as BOW, TF-IDF, CBOW, and Skip Gram, the SVM model achieved an accuracy ranging from 63.12% to 68.5% on both the training and test datasets. This suggests that the SVM model is not overfitting the training data as much as the LR model.

In summary, while both LR and SVM achieved different levels of accuracy on the training and test datasets across different text representation techniques, SVM generally performed better than LR in terms of generalizing to new unseen data.