Amazon Fine Food Review Analysis

Introduction:

The following presentation is about classifying the reviews from the given dataset into positive review or negative review based on ratings given by the users. The dataset consists of reviews of fine foods from Amazon. Amazon evaluations are frequently the most visible consumer product reviews. As a frequent Amazon customer, I was intrigued by the idea of visualizing the structure of a vast collection of Amazon reviews in order to become a more informed consumer and reviewer.

Dataset Description:

This dataset contains Amazon reviews of exquisite meals. The data spans more than a decade, with all 500,000 reviews up to October 2012 included. Product and user information, ratings, and a plaintext review are all included in reviews. We also have feedback from all of Amazon's other categories.

Preprocessing the Dataset:

Many duplicate entries were discovered in the reviews data. As a result, duplicates had to be removed in order to obtain unbiased results for data analysis. After removing the duplicates, it is required to preprocess the data before making the prediction model. The following steps are followed to preprocess the data:

i. Begin by removing the html tags
ii. Remove any punctuations or limited set of special characters like , or . or # etc.
iii. Check if the word is made up of english letters and is not alpha-numeric
iv. Check to see if the length of the word is greater than 2 (as it was researched that there is no adjective in 2-letters)
v. Convert the word to lowercase

vi. Remove “Stopwords”

vii. Finally Snowball Stemming the word (it was observed to be better than Porter Stemming)

After completing the above steps, we collect the words used to describe positive and negative reviews.

Transforming the text data into numerical vectors:

As machine learning model uses only the numerical vectors, we need to convert the text features into numerical vectors. In order to achieve this, I am using a Text Featurization Techniques which are:

i. Bag of words (BOW): BOW is a method to extract features from text documents. These features can be used for training machine learning algorithms. It creates a vocabulary of all the unique words occurring in all the documents in the training set.

ii. Tf-idf: "Term Frequency — Inverse Document Frequency" is abbreviated as TF-IDF. This is a strategy for quantifying a word in a document; we assign a weight to each word that represents its value in the document and corpus. In the fields of information retrieval and text mining, this method is commonly employed.

iii. Word2vec: This is a state-of-the-art algorithm that takes into account the semantic meaning of the term. When we enter a word, it is converted to vectors. It also automatically learns relationships from the text. Dense vectors are the result of the word2vec model. A big text corpus is required for the Word2vec model to work.

iv. Average Word2Vec: It is same as word2vec. But, here we use average of all word vectors in a sentence. This average vector will represent the sentence vector.

v. Tf-idf Word2Vec: We will calculate the tf-idf value of each word first, then multiply by the word2vec value of each word and pick the mean.

Building a Model:

I am considering two different models which are:

i. KNN using Brute Force

ii. KNN using k-d tree algorithm

The supervised machine learning algorithm k-nearest neighbors (KNN) is a simple, easy-to-implement technique that may be used to address both classification and regression issues. The

KNN algorithm believes that objects that are similar are close together. To put it another way, related items are close together.

K-d tree is a space-partitioning data structure for organizing points in a *k*-dimensional space. K-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches) and creating point clouds. *k*-d trees are a special case of binary space partitioning trees.

Brute Force Algorithms are exactly what they sound like – straightforward methods of solving a problem that rely on sheer computing power and trying every possibility rather than advanced techniques to improve efficiency.

Model Performance:

After classifying both models with all the text featurization techniques, the model K-d tree with average word2vec technique has the highest accuracy of 84% when compared with the rest. And K-d tree proves to be better model than brute force.