Movie Review Classification

Name - Saksham Nayyar

George mason University

# Abstract

The problem statement revolves around determining the sentiment of movie reviews as positive or negative. The primary objective is to develop an efficient and accurate K-Nearest Neighbor Classifier to predict the sentiment for 25000 reviews for movies provided in the test file.

To achieve this goal, I followed a structured approach. I began by preprocessing the training and testing datasets, cleaning and tokenizing the text while removing stop words and punctuation. I then employed the TF-IDF vectorization technique to convert the text data into numerical features, capturing the importance of words in the reviews. Subsequently, I utilized the K-Nearest Neighbors (KNN) classifier for sentiment classification.

Through rigorous experimentation and cross-validation, we fine-tuned the model's parameters, including the number of neighbors (k), and the choice of distance metric. The findings revealed that a KNN classifier with a distance-weighted strategy yielded the best results. We achieved a classification accuracy of around 80% on the test dataset.

# Program Instructions

To run the program and reproduce the results presented in the report, follow these instructions:

* Download file HW1-G01462522.py to the folder containing training and testing data file.
* Change the names of training dataset file to (train\_new.txt) and testing dataset file to (test\_new.txt).
* Execute python file (HW1-G01462522.py), after the execution a new file will get created in the same folder containing the results of the testing dataset (format.dat).
* Now you can compare the result in format.dat file to the ground truth to determine accuracy.

Table of Contents

[Abstract 2](#_Toc146967300)

[Program Instructions 2](#_Toc146967301)

[Introduction 4](#_Toc146967302)

[Methodology 5](#_Toc146967303)

[KNN Classification 5](#_Toc146967304)

[Text Pre-processing 5](#_Toc146967305)

[Approach and Experiments 6](#_Toc146967306)

[TF-IDF vectorizer 6](#_Toc146967307)

[K Nearest Neighbors 7](#_Toc146967308)

[Weight parameter of KNN algorithm 7](#_Toc146967309)

[Results 8](#_Toc146967310)

# Introduction

This report presents a study on sentiment analysis, focusing on the task of classifying movie reviews into positive and negative sentiments.

Analyzing these reviews can provide valuable insights for filmmakers, studios, and movie enthusiasts. Sentiment analysis, or opinion mining, is the automated process of determining sentiment polarity (e.g., positive, negative, or neutral) from text data. By applying sentiment analysis to movie reviews, we can gauge audience reactions and make data-driven decisions.

The main objectives of this report are as follows:

1. Implement the Nearest Neighbor Classification Algorithm.
2. Handle Text Data (Reviews of movies)
3. Design and Engineer Features from Text Data.
4. Choose the Best Model i.e., Parameters of a Nearest Neighbor Selection, Features and Similarity Functions

The scope of this project includes the following:

1. Data Preprocessing: Cleaning and transforming textual data to prepare it for machine learning model.
2. Model Development: Building and training a machine learning model for sentiment analysis.
3. Model Optimization: Fine-tuning model parameters using cross-validation to improve accuracy.
4. Efficiency Analysis: Evaluating the algorithm's runtime and exploring dimensionality reduction techniques for improved efficiency.

# Methodology

## KNN Classification

KNN is a non-parametric, instance-based, and lazy learning algorithm. Unlike many other algorithms that involve explicit training phases, KNN makes predictions based on the entire dataset. It operates on the principle that data points with similar features tend to have similar labels.

KNN algorithm comprises of three major things:

1. Training Data: The set of records.
2. Distance metric: To compute distance between records, Euclidean, Manhattan or cosine similarity.
3. K: The value of k, the number of nearest neighbors. This is the most important parameter to tune the model for optimal performance.

How KNN works:

It basically works by computing distance to other training records and then identifying the k nearest neighbors. Using the class label of the nearest neighbor to determine the class label of the unknown record.

In this project, I have employed KNN classifier for sentiment analysis of movie reviews.

## Text Pre-processing

Text cleaning and preprocessing are fundamental steps in machine learning (ML) projects. This section provides an overview of text cleaning and preprocessing techniques, highlighting the importance of TF-IDF (Term Frequency-Inverse Document Frequency) and stemming, which plays a crucial role in our sentiment analysis project.

Text data in its raw form often contains irrelevant information, and inconsistencies that hinder our model’s performance. Text cleaning and preprocessing aim to transform raw text into a structured and cleaner format, making it suitable for feature extraction and analysis.

Steps to clean data:

1. **Lowercasing:** Converting all the text to lowercase makes sure that our model “Hello” and “hello” are treated the same.
2. **Removing HTML Tags**: These are mostly present in data when we scrape data from the internet. And we need to remove these as they do not contain any sentimental information.
3. **Removing Punctuation**: Special characters and punctuation marks need to be removed as they do not contain any semantic information.
4. **Removing Stopwords**: Words like “us”, “this”, “are” do not contain any semantic information and need to be removed.
5. **Stemming**: Stemming is the process of reducing the words to their root or base form. It converts words like “running”,” run” and” runs” to “run” and make sure they all are treated the same way.
6. **TF-IDF Vectorization (Term frequency – Inverse document frequency)**: It changes the text to a matrix, into a numerical representation so that it can be fed to the machine learning model. TF-IDF score is the product of TF and IDF.
   * **TF (Term Frequency)** tells us how frequently a word appears in a document.
   * **IDF (Inverse Document Frequency)** measures the importance of a word across all documents.

**In this project**, I have used above mentioned text cleaning and pre-processing techniques to prepare the data for sentiment analysis.

# Approach and Experiments

In this section, I will define my approach to selecting parameters and features for the task. I used cross-validation to fine tune my model parameters. The following parameters were considered:

## TF-IDF vectorizer

I have used TF-IDF vectorization technique for text data preprocessing. I experimented with different TF-IDF parameters such as:

* **max\_df (Maximum Document Frequency):** It is used to exclude the words that have high document frequency.

For e.g. In this project, max\_df=0.7, It means that words that appear in more than 70% of the documents will be ignored because they are too common to be useful.

* **min\_df (Minimum Document Frequency):** It is used to the words that have low document frequency.

For e.g. In this project, min\_df=0.004, It means that words that appear in less than 0.4% of the documents will be ignored as these are rare words that might be adding noise to the model.

* **ngram\_range**:It determines the range of word combinations to consider when vectorizing text. For example.

In this project, ngram\_range= (1,2), It means both unigrams and bigrams will be considered because sometimes we have not or a negative word in front of a positive word which negates it, but we could not capture this in our model if take only unigrams.

* **max\_feature:** It is the maximum number of features to be considered during text vectorization. This parameter is used to reduce the dimensionality of the feature space.

In this project I experimented with different values of max\_feature and then calculated cross-validation score as shown below. After plotting this graph, I decided to take max\_features = 1500.

**A graph with a line

Description automatically generated**

Fixed the value of K = 100, then ran the model for different values of max\_features.  
Plotted the graph between accuracy(y-axis) and max\_features(x-axis).

## K Nearest Neighbors

I experimented with various values of K to find the optimal number of neighbors. The results are presented in the following graph.

Fixed the value of all other parameters, then ran a for loop for different values of K (k\_values = range (1,500,15).  
Plotted the graph between accuracy(y-axis) and k\_values(x-axis).

## Weight parameter of KNN algorithm

It controls how the KNN algorithm assign weights to the neighboring data point. I experimented with both values ‘uniform’ and ‘distance’, and We observed that using ‘distance’ led to increased accuracy as shown below. The accuracy shown below is from cross-validation.

|  |  |
| --- | --- |
| Weights | Accuracy (%) |
| “uniform" | 79 |
| “distance" | 80 |

# Results

In this section, we present the results of our data mining experiments:

* **Optimal parameters:** Based on our cross-validation experiments, the optimal parameters for the K-nearest Neighbors classifier are K = 150, weights = ‘distance’ and for TF-IDF vectorizer, we used ‘max\_df = 0.7’, ngram\_range = (1,2), min\_df=0.004, and max\_features=1500.
* **Performance Metrics:** Our final model achieved an accuracy of exactly 80% on the test dataset as shown in the miner.
* **Miner Submission:** Miner Rank for the latest submissionis rank 20. This includes multiple submissions by same person. If we remove multiple submissions, Rank is 4.