

# CS466 Final Project Report

By: Arjun Nair(arjunpn2)

Project Link: <https://github.com/arjunpnair7/FinalProject-CS466>

## Background

Throughout this paper, I will be referring to the K – nearest neighbors algorithm (KNN). KNN is a machine learning algorithm that can predict the label of a class. KNN generates this prediction by first finding the k nearest neighbors using some designated similarity metric. Then, the k nearest neighbors will do a group vote to predict what the label of the test point should be. The underlying assumption is that points that are close to each other are similar. Consequently, the choice of distance function significantly impacts the performance of the classifier.




I also refer to word embeddings in this paper. Word embeddings are a numerical, vectorized representation of words. The similarity between two embeddings can be found by computing the cosine similarity metric. The cosine similarity metric is essentially a way to describe how aligned two vectors are.

## Introduction

The purpose of my project was to extend the underlying ideas of the global alignment algorithm by applying them beyond the biological context. For my project, I implemented a KNN classifier that can predict the genre of a movie. My classifier uses a movie dataset from Kaggle. To evaluate the similarity between two movies, I applied the global alignment algorithm to the overview of the movies. The calculated scores are then used for the KNN classification.

## Methods

I used the [following](#) dataset from Kaggle that is annotated with various features such as release date, run-time, overview, etc. The dataset contains 1000 entries. I used the first 900 entries for the training set and reserved the last 100 entries for testing/validation purposes. An excerpt of this dataset is provided here:

imdb_top_1000.csv (438.1 KB)									
Poster_Link	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating	Overview			
	The Usual Suspects	1995	142 min	Drama	8.5	The imprisoned man told over a number of years, finding police and eventual redemption through acts			
	The Godfather	1972	175 min	Crime, Drama	9.2	An organized crime dynasty's aging patriarch transfers control of his			
	The Dark Knight	2008	152 min	Action, Crime, Drama	9	When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, Batman must			

I decided to use Python to complete this project as it has many powerful visualization and data analysis libraries. I used Pandas to parse the data and convert it into a compatible format for me to work with.

## Technical Challenges

When comparing the summary of two movies, it is difficult to evaluate the similarity by using the alignment algorithm to match strings. This is because a similar description can be described in many ways, and it is unlikely that the exact words and positions would match. To address this, I converted each word of the summary into its corresponding word embedding. Then, I used the cosine similarity to determine the reward. Specifically, if the cosine similarity was above a certain threshold, then it would be considered a match, otherwise it would be a mismatch. If the word was not present in the embedding set, I would treat it as a gap.

The usage of a large set of word vectors introduced performance issues as checking for the existence of a word in the embedding dictionary was a time-consuming operation. With the repeated computations over the 900 neighbors of the training set, it was not feasible to evaluate the accuracy in a timely manner.

Initially, I implemented caching in my code to speed this up and avoid redundant computations. However, even with this change, the method was still taking too long. I believe caching was not effective here due to the variable, inconsistent nature of written language which does not allow for much reuse.

To address this issue, I opted to significantly reduce my test size. Rather than using 100 as I had initially planned, I decided to use 25 test elements with the 900 training examples. I also used a mini version of the original embedding dataset so that I could speed up the computation. These changes significantly reduced the time for the computations to complete.

The dataset used for this project contained movies annotated with multiple genres, which required defining a criterion for evaluating the success of the predictions. Given the inherent variability in natural language, I opted for a relaxed evaluation approach: if the model correctly predicted at least one genre (from the multiple possible genres for a movie), the model would consider the prediction successful. By using such a function, we can capture the partial correctness of the predictions.

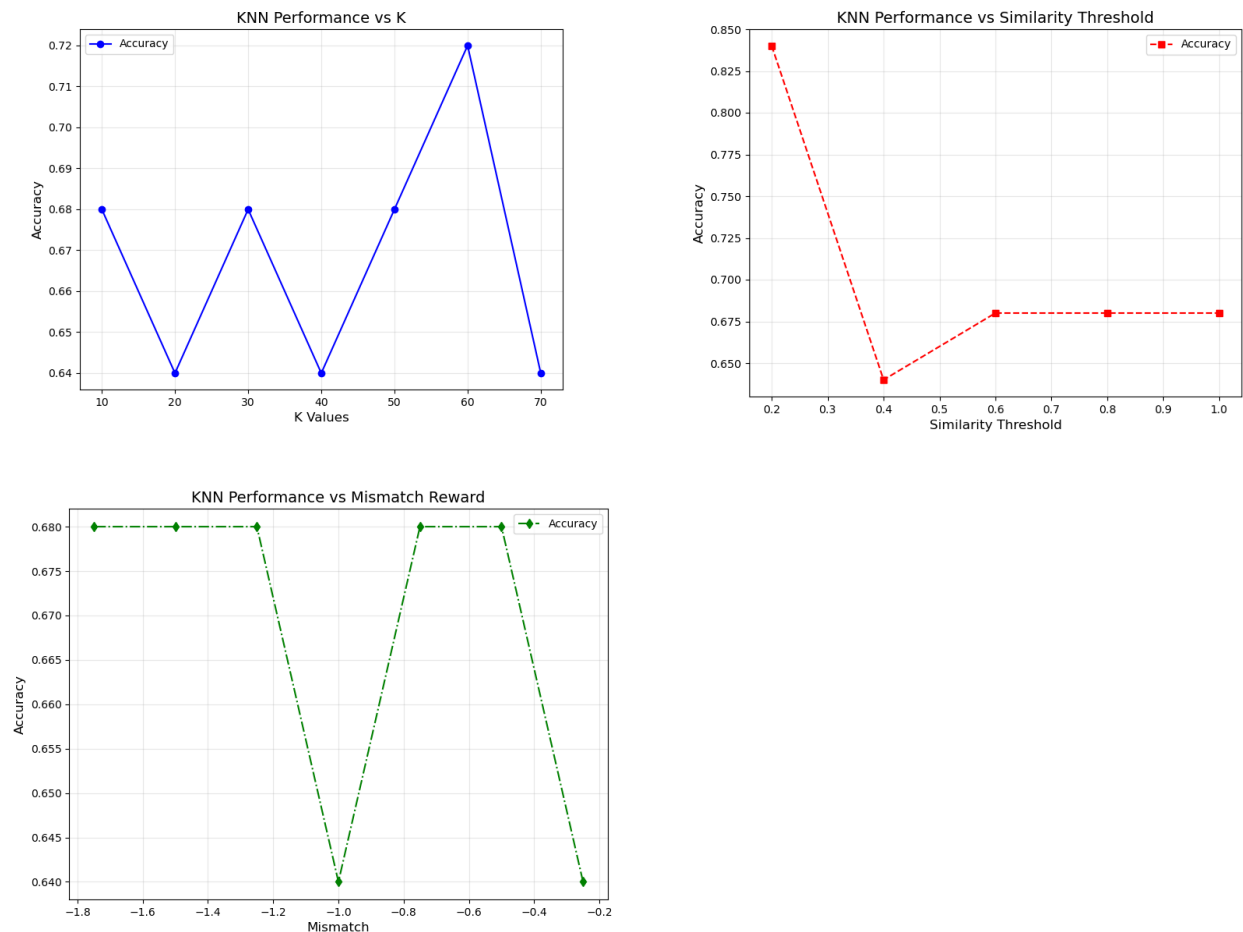
## Validation Results

To implement the classifier, I first created a generic implementation of the KNN algorithm and the global alignment algorithm. Both implementations were completely decoupled from any movie-specific information so that I could test it easily.

To test the accuracy, I created a series of unit tests for both methods. Rather than manually working through vector math calculations by hand to generate the tests, I decided to test the method on simpler, well-known domains. For global alignment, I tested it on various DNA sequences by subbing in a standard reward function. This way, I could verify the accuracy of the method by comparing it to the results of well-known [websites](#) that can

compute the alignment scores. To test KNN, I decided to test it with a series of points in space using the Manhattan distance as the distance function.

I experimented with various hyper-parameters to see how it would affect the accuracy of the classifier. Specifically, I experimented with K (the number of neighbors examined when doing the group vote), similarity threshold (the threshold for cosine similarity for the reward function to consider it a match), and the mismatch reward (the reward given for two vectors whose cosine similarity is below the threshold).



Above, we can see a graph of the accuracy value when compared to different parameter settings. For reference, this graph was created when the classifier was run with 900 training samples and 25 test samples. The base classifier used `gap_penalty = -1`, `similarity_threshold=0.7`, `similarity_reward=1`, `string_mismatch_penalty=-0.5`. In the experiments above, the k value, threshold value, and mismatch values were varied respectively.

We can observe that lower similarity threshold values resulted in a greater accuracy. I believe this may be due to the nature of the embedding file. Since I used a smaller embedding set, the accuracy values of the embeddings may not have been as high as I had

anticipated. We can see this with words like 'king' and 'queen' which would usually have very similar meanings. However, the cosine similarity for these two words only resulted in a similarity of around .65.

For the k values, we can see there is fluctuation in the accuracy, but the greatest accuracy is achieved at  $k = 60$ . I believe this makes intuitive sense as the more neighbors we examine, the more representative the group vote of the KNN will be. We can see that after  $k = 60$  the performance goes down. This may be because the model is beginning to overfit too closely to characteristics of the dataset I used. Consequently, the model is losing its ability to generalize which causes the decline in accuracy.

There is significant variability in these results, and the observed patterns may not generalize well. This is likely due to the small test size of 25, which was constrained by hardware limitations. As a result, such a limited sample size may be insufficient to reliably identify clear or consistent patterns.

## Conclusion

Throughout this project, I explored the underlying ideas of the global alignment algorithm by applying them beyond the context of biology. By using the alignment algorithm for the task of language matching, the model was able to analyze the similarities between movie overviews and use that to predict the genre of a new movie based on its summary. By encoding the text description using embeddings, the model was able to learn meaningful patterns to generate genre predictions.

## Future Work

I believe the performance of this project can be further improved. Specifically, I think the greatest potential for improvement lies in tuning the parameters of similarity threshold, match reward, and mismatch reward used in the reward function. I think there is a sweet spot that can produce better results. Also, it is important to remember that my training and testing set were relatively small in this experiment. I think it would be interesting to extend the model on more powerful hardware by using a larger training and testing set.

Furthermore, in this implementation I used a binary threshold in the reward function for matching the embeddings. I think it would also be worthwhile to experiment with a more nuanced function that would assign different rewards based on the level of similarity. For example, cosine similarities between .5 and .7 would receive a reward of 5 while cosine similarities greater than or equal to .7 would receive a reward of 10.