UNIVERSITY OF CALIFORNIA
Los Angeles

An Application of Document Embedding Methods with Movie Plots

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics and Data Science

by

Margaret Burroughs Johnson

2023

ABSTRACT OF THE THESIS

An Application of Document Embedding Methods with Movie Plots

by

Margaret Burroughs Johnson
Master of Applied Statistics and Data Science
University of California, Los Angeles, 2023
Professor Yingnian Wu, Chair

This thesis will explore and compare Natural Language Processing methods to determine the similarity between movies based on plot descriptions. Three document embedding methods, TF-IDF, Doc2Vec, and S-BERT, are implemented. The results are evaluated using spectral clustering and normalized mutual information. Specific case studies are also presented. The objective is to identify movies that are most similar in topic or theme based solely on plot content.

The thesis of Margaret Burroughs Johnson is approved.

Nicolas Christou

Tao Gao

Yingnian Wu, Committee Chair

University of California, Los Angeles

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

This paper applies natural language processing techniques, including TF-IDF [1], Doc2Vec [2], and S-BERT [3], to measure the similarity of movie plots. A numerical vector representation of each movie is generated, which allows for the use of cosine similarity to identify similar films. The data comes from movie plot descriptions from Wikipedia [4]. It includes 33,416 films released between 1901 and 2017.

Streaming services, including Netflix and Amazon, were early adopters of recommender systems, investing in machine learning algorithms to recommend new content to viewers to keep them engaged. When it comes to movies, the most familiar examples of recommender systems involve collaborative filtering, which recommends content to users based on what similar users also enjoy [5]. Famously, in 2006 Netflix launched an open competition with a cash prize for teams to compete in improving their collaborative filtering algorithm based on user reviews [6]. Since then, the amount of data available to streaming services has grown exponentially, and recommendations have become increasingly accurate.

The goal of this paper is not to create a model comparable to the sophisticated recommender systems that streaming providers currently use. These recommendation systems perform extremely well. However, they tend to recommend well-known movies first, which limits exposure to other movies the viewer might find interesting. Rather, this paper aims to explore methods of identifying movies that are most similar in topics or themes based solely on plot content. In contrast to collaborative filtering, the models described in this paper do not rely on movie popularity or viewer characteristics. Therefore, the films identified

will likely be more obscure. All the plot descriptions, including for foreign films, are in English, allowing for the opportunity to discover films with similar plot points across different languages.

The three embedding methods are examined in Chapter 4. First, TF-IDF is performed alongside UMAP for dimension reduction [7]. Second, a Doc2Vec algorithm is implemented using the PV-DM variation. Third, a pre-trained S-BERT model is applied. Finally, spectral clustering is performed on all three outputs. The results are presented in Chapters 5 and 6. To evaluate the models, a normalized mutual information score is calculated to compare the cluster assignments to the genre labels [8]. The assumption is that, while film genre is only one piece of information, movies of the same genre tend to be more similar than movies of different genres. The results of each model are visualized using UMAP to reduce the data to two dimensions. Finally, cosine similarity is used to return the top 10 closest results for 4 target films.

Based on the normalized mutual information scores, it appears that the models perform relatively well. S-BERT and Doc2Vec score very similarly, clearly outperforming TF-IDF. However, an examination of the case studies in Chapter 6 implies that S-BERT may be superior in generating cohesive movie neighborhoods at a local level.

CHAPTER 2

Methodology Overview

2.1 Document Embedding

To identify similar documents (movie plots) and perform clustering, numerical representations of the documents must be generated. This paper will cover three methods: (1) TF-IDF, which is a simple statistical measure of word frequency; (2) Doc2Vec, which is an unsupervised shallow neural network; and (3) S-BERT, which is a pre-trained transformer-based neural network. These three approaches are detailed in Chapter 4.

2.2 Evaluation

Evaluation presents a challenge because the models presented in this paper are unsupervised. The task is not predictive with a clear right or wrong answer. However, the data does include the genre of each film. Assuming that movies within the same genre are more similar than movies of different genres, we can use the genre labels as a proxy for movie similarity. This assumption allows us to perform clustering and evaluate how closely those clusters align with the genre labels.

2.2.1 Spectral Clustering

This paper uses spectral clustering to cluster the embedded plot data into groups. Based on graph theory, this method has significant benefits over other popular clustering methods, such

as k-means. For example, it works well with high-dimensional data because it first computes an adjacency graph rather than relying on standard distance measures [9]. Also, it performs much better in cases where clusters wrap around each other and cannot be separated by drawing a straight line. Spectral clustering was implemented using the scikit-learn Python library [8].

2.2.1.1 Spectral Clustering Main Equations and Structures

Below is the structure of the spectral clustering algorithm, given the embedded plots, $\underset{n\times c}{X}$, where n is the number of plots and c is the number of components.

1. Construct the Weight Matrix $\underset{n\times n}{W}$, using pairwise Euclidean distances. $\underset{n\times n}{W}$ will be a symmetric matrix with zeros on the diagonal. For a pair of rows in X, x_i and x_j,

$$W_{ij} = \sqrt{(x_i \cdot x_i) - 2(x_i \cdot x_j) + (x_j \cdot x_j)}$$

2. Construct Affinity Matrix $\underset{n\times n}{A}$, which will be a symmetric matrix with zeros on the diagonal. In this case, we construct a fully-connected graph using the Gaussian radial basis kernel, where σ is a free parameter that controls the width of the neighborhoods [10]. For this implementation, σ is set to 0.7.

$$A_{ij} = exp\left(\frac{-|W_{ij}|^2}{2\sigma^2}\right)$$

3. Construct the Degree Matrix $\underset{n\times n}{D}$, which is a diagonal matrix whose i-th diagonal element is the sum of all elements in the i-th row of A

$$D_i = \sum\nolimits_j A_{ij}$$

4. Construct the Normalized Laplacian Matrix $\underset{n\times n}{L}$ [11], such that

$$L = D^{-1/2}(D - A)D^{-1/2}$$

5. Find the first k eigenvectors of L $(u_1, ..., u_k)$ corresponding to the smallest k eigenvalues of L. Form the matrix $\underset{n\times k}{U}$ with these eigenvectors. Create matrix $\underset{n\times k}{T}$ by normalizing the rows of U to have unit length of 1 [10].

$$T_{ij} = \frac{U_{ij}}{\left(\sum_k U_{ik}^2\right)^{1/2}}$$

6. Use this matrix T as the input to a k-means algorithm to cluster the data into k clusters, where each point is represented by a row in T. If the original point x_i will be assigned to cluster q if and only if row i of matrix T is assigned to cluster q [9]. Because the movie plot data includes 19 distinct genre labels, we set $k = 19$.

2.2.2 Mutual Information

This paper will use normalized mutual information (NMI) to compare the clustering results with the actual genre labels for each of the three embedding methods. This enables evaluation and comparison of the results and performance of each model. NMI will be implemented in Python using scikit-learn [8].

NMI is built off mutual information (MI), which has no upper bound. NMI normalizes the metric so that scores range from 0 (no mutual information) to 1 (perfect mutual information). The equation for MI is as follows, where U and V are two different label assignments of the same N data points – in this case, the spectral clustering results and the ground-truth genre labels. Here, $P(i) = \frac{|U_i|}{N}$ and $P'(j) = \frac{|U_j|}{N}$.

$$\mathrm{MI}(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

NMI is calculated by dividing MI by the mean of the entropy of each U and V. The entropy of U is calculated as $H(U) = -\sum_{i=1}^{|U|} P(i)\log(P(i))$, and likewise for $H(V)$.

$$\mathrm{NMI}(U,V) = \frac{\mathrm{MI}(U,V)}{mean(H(U),H(V))}$$

2.3 Similarity Measurement

Finally, we will use cosine similarity to measure the difference between a target film and each other film in the data set. Cosine similarity is often recommended as the best distance measure for semantic tasks [12]. The cosine similarity between vectors a and b is calculated as the dot product divided by the product of the norm of each vector. Given a film of interest, we will return the top 10 most similar films.

$$Cosine\ Similarity = cos(\theta) = \frac{a \cdot b}{\|a\|\|b\|}$$

CHAPTER 3

Data

3.1 Data Set Information & Preparation

The data used in this paper comes from movie plot descriptions scraped from Wikipedia [4]. The raw data includes information on movie title, release year, genre, country, and director. This section provides some exploratory analysis and explains the methods used to clean and prepare the data for use in the models discussed in Chapter 4. The data set originally included 33,868 unique films, which was reduced to 33,416 after cleaning. All films were released between 1901 and 2017, and the distribution of release years is shown in Figure 3.1.

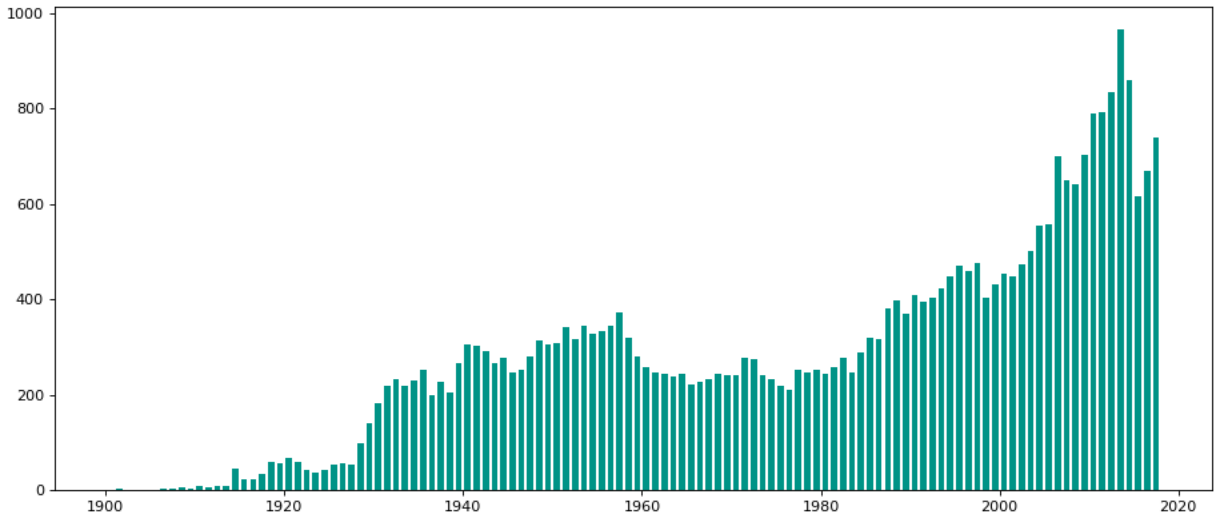


Figure 3.1: Distribution of Films by Release Year

Just over half of the films in the final data are from the United States, with the remainder

coming from 23 other countries or regions. As noted earlier, all of the plot descriptions, even for foreign-language films, are in English, so it will be interesting to see if this allows for finding similarities between films of different languages.

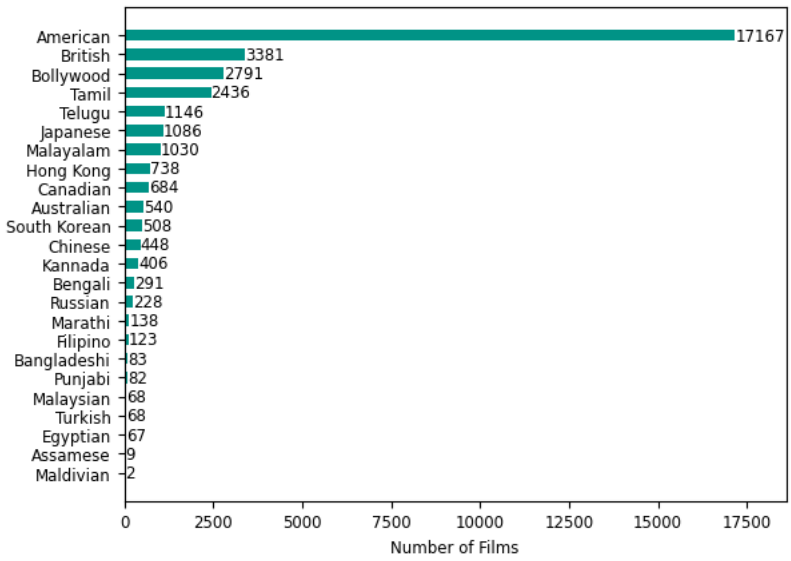


Figure 3.2: Distribution of Films by Region of Origin

3.1.1 Cleaning

The original raw data included some duplicate films. For example, if a movie was released in multiple countries, sometimes the scraped data had multiple rows for the same film (with the same plot). Duplicate plots were dropped, leaving 33,868 unique films.

One limitation of this data is that the plot descriptions vary from film to film, ranging from 1 word to over 6,000 words (Table 3.1). To avoid issues caused by very short plots, 348 plots with fewer than 15 words were removed.

Table 3.1: Plot Length Statistics

Total Movie Plots	33,868.00
mean word count	338.01
std word count	287.99
min word count	1.00
25%	110.00
50%	257.00
75%	525.00
max word count	6,177.00

In the raw data, the "Genre" column took over 2,000 unique values. Investigation revealed this was mainly due to variations in formatting or phrasing, and most of these could be grouped into one of the more general categories in Figure 3.3. Here, the "Unknown" category includes 5,511 films where the genre was either missing or did not easily fit into one of the other categories.

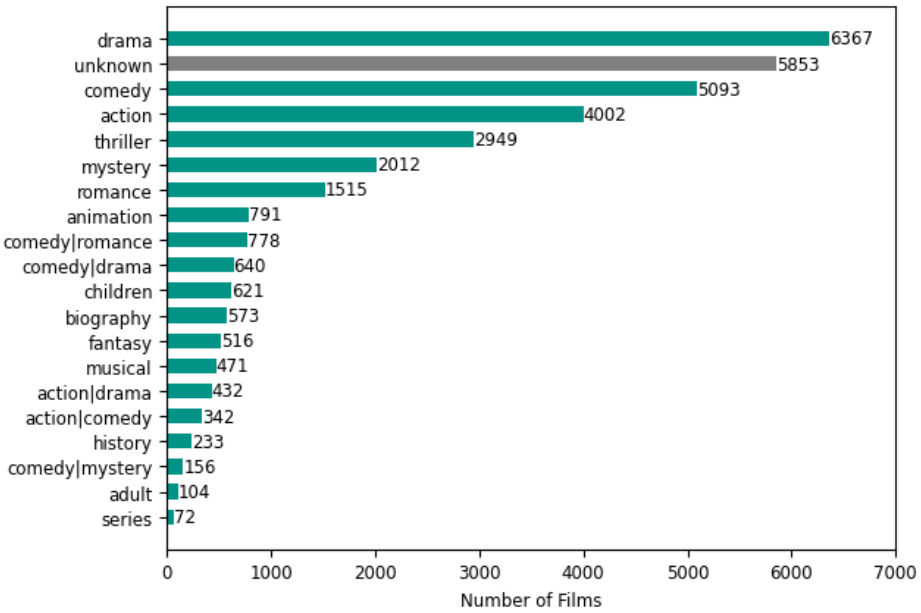


Figure 3.3: Number of Films by Genre

3.1.2 Named Entity Recognition

A main challenge that became apparent when observing preliminary results was the presence of character and actor names within the text. Including these names as words led to nonsensical results when looking at a movie's nearest neighbors. For example, there is nothing particularly similar about the 2014 thriller *Gone Girl* and the 2005 romantic comedy *The Wedding Date*, aside from both films featuring main characters named Amy and Nick.

To resolve this, all person names were identified and removed using *en_core_web_trf* from the spaCy library [13]. This is a transformer pipeline built on the roBERTa model from HuggingFace. It is publicly available for production use and has multiple applications, including Named Entity Recognition (NER). The example in Figure 3.4 shows the result of NER on the first paragraph of the plot description of *Casablanca*. Here, spaCy accurately identifies linguistic entities, including people, places, and dates, enabling the removal of all entities tagged as "PERSON."

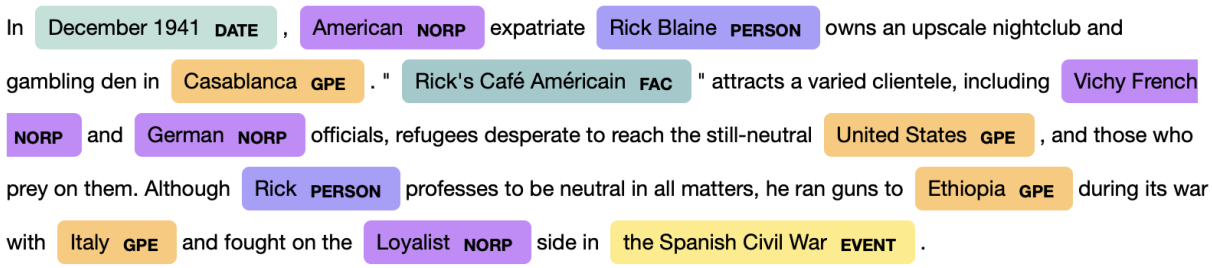


Figure 3.4: NER with spaCy on Casablanca

Something to note is that spaCy does not distinguish between the names of fictional and non-fictional characters. Therefore, name removal could have unfortunate results for historical or other non-fiction films but should still benefit the overall results.

3.1.3 Data Pre-Processing

Additional steps were taken to process the text. First, all numbers and brackets related to in-text citations were removed, along with extra empty white space. Then, the final pre-processing steps were tailored to each of the three modeling approaches.

For TF-IDF, punctuation and stop words were removed. Although TF-IDF would have automatically discounted the weights of very common words, removing them greatly improves computation time. Likewise, words appearing in fewer than 20 documents were removed. These adjustments decreased the size of the embedded results, allowing for faster UMAP processing in the dimension reduction step.

For Doc2Vec, stop words were retained. However, words appearing less than 5 times in the corpus were removed to speed up computation, since exceptionally rare words are not helpful when it comes to training. In addition, punctuation was removed, and the words were tokenized prior to being fed into the Doc2Vec model.

For S-BERT, punctuation and stop words were retained in order to keep as much context as possible. S-BERT was trained on text that included punctuation, so removing it could reduce accuracy [3].

CHAPTER 4

Models

4.1 TF-IDF With UMAP

One of the earliest and simplest numerical text representation techniques is TF-IDF (Term Frequency-Inverse Document Frequency), first introduced in 1972 [1]. At the time, TF-IDF was an important advancement in the world of NLP. It improved on earlier techniques by giving more weight to less frequent, more specific words and less weight to very common words. This paper uses the Scikit-Learn Python implementation of TF-IDF [8].

TF-IDF represents each document as a vector with n dimensions, where n is the number of words in the corpus. The value of each word in a document is a combination of TF (term frequency), which measures the importance of the word relative to other words within the document, and IDF (inverse document frequency), which measures how rare the word is within the corpus.

$$TF = \frac{number\ of\ times\ the\ word\ appears\ in\ the\ document}{total\ number\ of\ words\ in\ the\ document}$$

$$IDF = log\left(\frac{number\ of\ documents\ in\ the\ corpus}{number\ of\ documents\ in\ the\ corpus\ that\ contain\ the\ word}\right)$$

$$TF - IDF = TF * IDF$$

The result is a sparse matrix that can be either used in its raw form or fed into another

machine learning algorithm. Working with this raw TF-IDF matrix can be computationally expensive. Identifying similar vectors can be challenging due to the curse of dimensionality [14], whereby distances between pairs of points tend to lose meaning in high dimensions. When applying common distance measures such as Cosine Similarity or Euclidean Distance or attempting to perform clustering, all points in a high dimensional space can appear nearly equidistant.

Dimensionality reduction techniques can help solve this problem and improve computation time. This paper focuses on UMAP (Uniform Manifold Approximation and Projection), a non-parametric dimension reduction algorithm that works effectively on sparse and non-linear data. It is particularly adept at preserving local structures with minimal distortion, but it has been shown to preserve more of the global structure than the similar t-SNE method while being more computationally efficient [7].

UMAP works by first computing a fuzzy topological representation of a data set and then optimizing a low-dimensional embedding of the graph through stochastic gradient descent [15]. The intuition is that UMAP attempts to learn a manifold on which the data is uniformly distributed. The authors of the original UMAP paper created an implementation in Python, which this paper will use [16].

The algorithm below outlines UMAP, where X is the data to have dimensions reduced, $n\_neighbors$ is the selected neighborhood size, $n\_components$ is the selected dimension of the target reduced space, min_dist is the minimum distance between embedded points, and n_epochs is the number of epochs to perform at optimization. The output data is Y.

Algorithm 1: UMAP

Result: Optimized Embedding Y

1. Construct the relevant weight graph;

for *all* $x \in X$ **do**

  Generate local fuzzy simplicial sets using $n\_neighbors$

  Union together all fuzzy simplicial sets

end

2. Generate low-dimensional embedding Y;

Spectral Embedding: Initialize Y with $n\_components$

Optimize Embedding: **if** $epoch < n\_epochs$ **then**

  minimize fuzzy set cross-entropy;

  use stochastic gradient descent with negative sampling to optimize Y;

else

  end embedding optimization;

end

Words appearing in less than 20 documents were removed to speed up computation. The result of the TF-IDF step is a 33,520×15,297 matrix, which was then reduced to 33,520×200 using UMAP. One weakness of TF-IDF is that it does not preserve any information about word order or context. It is a simple statistical measure drawing on the frequency of word usage. The other models discussed in this paper have developed more sophisticated approaches.

4.2 Doc2Vec

The Doc2Vec algorithm was first introduced by Le and Mikolov in 2014 [2]. Originally known as "Paragraph Vector," the approach uses neural networks to generate numerical vector representations of documents. Documents similar to one another will be closer together in

the vector space, allowing the use of clustering methods to identify groups of similar texts.

Doc2Vec is an extension of Word2Vec, which produces vector representations of words. There are two main versions of Doc2Vec: Paragraph Vector Distributed Memory (PV-DM) and Paragraph Vector Distributed Bag of Words (PV-DBOW). PV-DM is based on the Continuous-Bag-of-Words (CBOW) version of Word2Vec, which uses surrounding words to predict a missing word. PV-DBOW is based on the Skip-Gram version of Word2Vec, in which a word is used to predict the surrounding context [17].

In practice, researchers have found that PV-DBOW performs well when applied to short texts, such as tweets [18]. Although PV-DM can be more computationally expensive than PV-DBOW because it stores the word embeddings in addition to the document embeddings, an advantage is that it preserves word order. This paper will focus exclusively on the Distributed Memory version of Paragraph Vector (PV-DM), which uses the CBOW model to produce word embeddings.

In the CBOW variation of Word2Vec, context words are represented as one-hot vectors and passed through a weight matrix (W) to the hidden layer (h). The network uses the hierarchical softmax function to predict the missing word in the output layer. Hierarchical softmax uses binary trees to approximate the softmax activation function at a much lower computational cost [19]. The prediction is then compared to the target word, and the error is used to update the weight matrix through backpropagation. PV-DM operates in the same way, with the addition of a document ID vector that is passed in conjunction with the context word vectors. Both the word weight matrix (W) and the document weight matrix (D) are trained simultaneously. Figure 4.1 [20] shows a visual representation of the algorithm flow, where the vocabulary size is M, the number of documents in the corpus is N, and each word and document are mapped onto a vector of p dimensions.

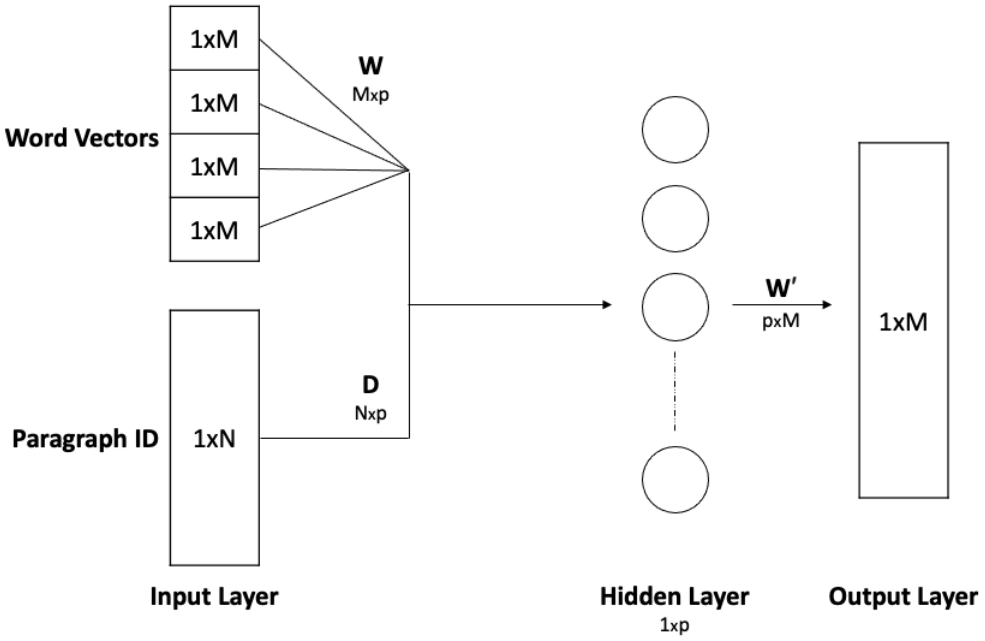


Figure 4.1: Doc2Vec PV-DM Algorithm

This paper implements Doc2Vec through the Gensim library in Python [21]. The window size determines how many of the context words surrounding the missing word will be passed to the model at each step. The plots tend to be relatively long, and we are focused on the broad content of each plot, so a window size of 15 was selected [22]. The training ran for 30 epochs. The initial learning rate is 0.025 and decreases at each epoch until reaching 0.0001.

The resulting weight matrices, D and W, are the resulting document and word embeddings. In this case, there are 33,520 document vectors in D (one for each plot) and 33,283 word vectors (one for each word included). At training, the vector length was set to p=200, which applies to both sets of vectors. An example of the word embeddings is shown in Figure 4.2, where cosine similarity was used to return the top 10 words with vectors closest to that of "Princess."

```
Top 10 most similar to 'princess'

[('prince', 0.6264454126358032),
 ('king', 0.560699462890625),
 ('queen', 0.5016921162605286),
 ('serpent', 0.42189037799835205),
 ('goddess', 0.3878280222415924),
 ('palace', 0.38101547956466675),
 ('crown', 0.37158021330833435),
 ('duke', 0.3704667389392853),
 ('courtier', 0.36971068382263184),
 ('kingdom', 0.3660765290260315)]
```

Figure 4.2: Doc2Vec: 10 Words Most Similar to "Princess"

Finally, the document vectors in D are normalized to be of unit length. This is achieved by dividing each element of the vector by its length.

4.3 S-BERT

A limitation of the previous two methods is that they depend entirely on the data fed to them for training. The final model discussed in this paper is S-BERT (Sentence Bidirectional Encoder Representations from Transformers), which is implemented using the *all-distilroberta-v1* model [3] from Hugging Face. One key advantage of this model is that it is pre-trained on over 1 billion sentence pairs and triplets – far more text than is in the movies data set. The S-BERT model takes input text and maps each document to a 768-dimensional dense vector space.

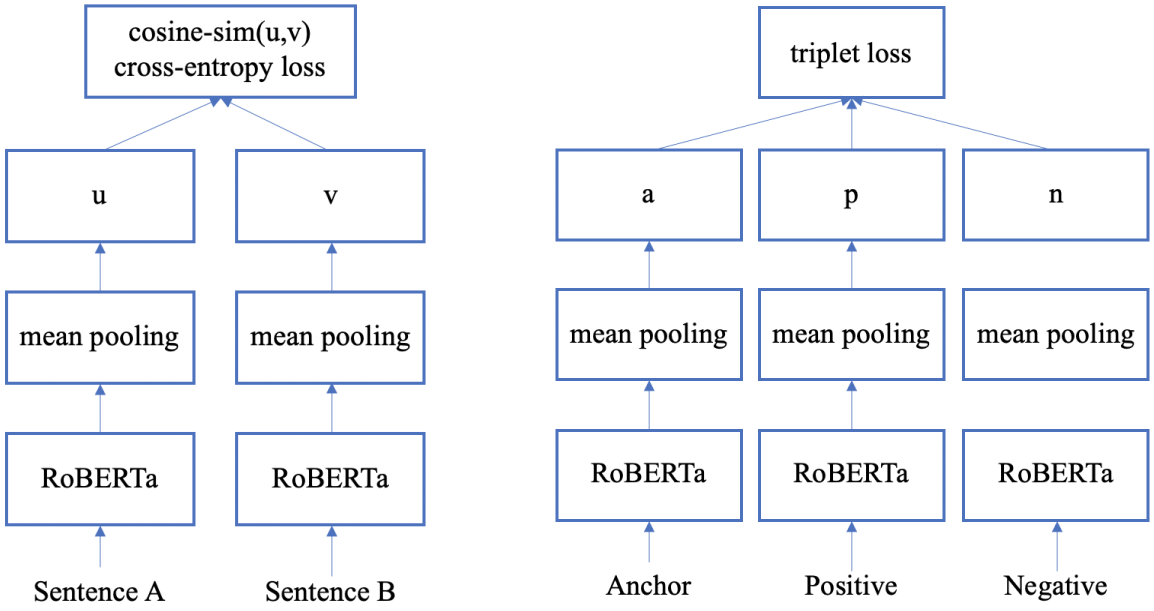
The S-BERT model used in this paper was trained by fine-tuning the pre-trained RoBERTa model, an improvement on the original BERT model [23]. Like BERT, RoBERTa uses the encoder part of a transformer architecture to process all the words in a text at once rather than sequentially. The model features a self-attention mechanism, allowing it to understand the meaning of a word based on its context and relationships with other words. While RoBERTa works very well for a variety of tasks, including word embeddings, next-word prediction, and text generation, it is not well designed for document embedding or semantic

similarity calculations. Researchers have found that feeding pairs of text to the model to find the most similar combination is very computationally expensive, and other methods, such as averaging the output layer or using the CLS token, do not yield reliably good results [3].

S-BERT solves these problems by providing an efficient way to produce embeddings, which may be compared using similarity calculations or used as input to clustering analyses. The researchers who developed S-BERT used contrastive learning through siamese and triplet networks to fine-tune the weights in the underlying RoBERTa model. In both cases, a mean pooling function is added as a head to the RoBERTa model to produce a sentence embedding.

For the siamese network (Figure 4.3a), a batch of sentence pairs is passed through the model, and cosine similarity is used to calculate the similarity of each resulting set of vectors. The goal is for actual sentence pairs to be closer in the vector space than to non-pairs. Cross-entropy loss is then applied to update the weights in RoBERTa. The triplet network (Figure 4.3b) is designed to process sentence triplets, where one sentence is an anchor, one sentence is in agreement (positive), and one sentence contradicts (negative). The goal is for the embedded vectors of the anchor and the positive sentences, a and p, to be closer together than the embeddings of the anchor and negative sentences, a and n. In the original paper, the authors aim to minimize the triplet loss function below, with $\epsilon = 1$ [3].

$$Triplet\ Loss = max(\|a - p\| - \|a - n\| + \epsilon, 0)$$



(a) Siamese Network Training

(b) Triplet Network Training

Figure 4.3: S-BERT Training Structures

After the RoBERTa piece of the model is sufficiently tuned, it can be used for inference. New sentences are passed through the RoBERTa model, mean pooling is applied, and the resulting vectors are the embeddings. One drawback of this pre-trained model is that it is intended for sentences and short paragraphs, so each input is automatically truncated to 512 tokens, which includes words and punctuation. Although 34% of the plot descriptions will be at least somewhat truncated, this paper examines whether the pre-trained S-BERT model will still yield good results.

CHAPTER 5

Results

This chapter explores the normalized mutual information scores between the 19 true genres and the 19 cluster labels generated via spectral clustering for each method. Spectral clustering is only performed on the 27,667 plots where genre is known. We find that NMI is highest for Doc2Vec, followed very closely by S-BERT. As expected, TF-IDF with UMAP scores lowest. As outlined in Chapter 2, NMI ranges from 0 to 1, where a score of 0 indicates no mutual information and a score of 1 indicates perfectly matching information.

We could expect these NMI scores to be much higher if the goal of this paper was to build a genre classification model. However, scores greater than zero still demonstrate value since the task examined is unsupervised and self-supervised document embedding. When the clusters are assigned at random, NMI is essentially zero.

Table 5.1: Normalized Mutual Information

Model	**Normalized Mutual Information**
TF-IDF with UMAP	0.0869
Doc2Vec	0.1233
S-BERT	0.1226

NMI is also helpful in evaluating the similarity of results between the three methods, as shown in Figure 5.1. Among the three methods, S-BERT and Doc2Vec produced the most similar results. It is worth noting that the results of each method are more similar to the results of the two other methods than to the original genre labels. This is not necessarily

a surprise, as the models were each fed identical data (plots) without any information on genre.

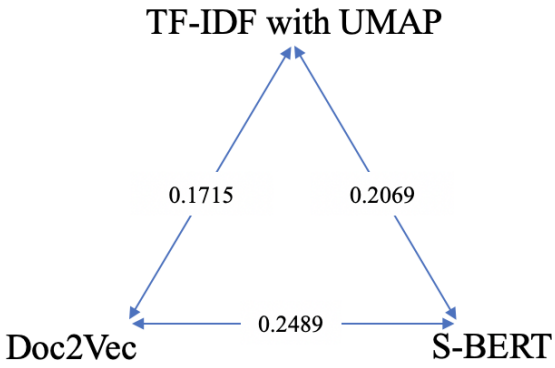


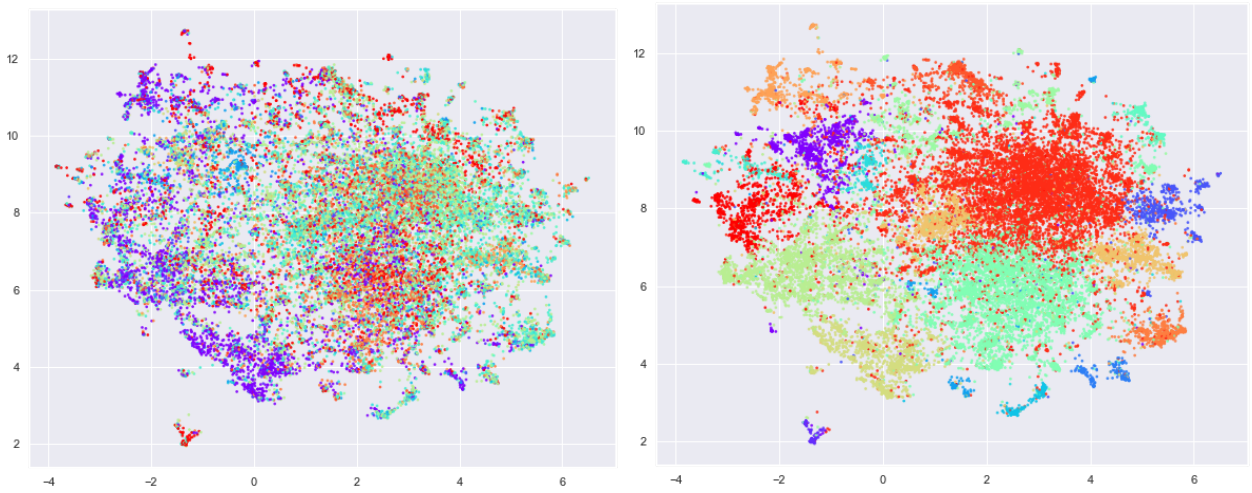
Figure 5.1: Normalized Mutual Information Between Methods

The rest of this chapter will explore the results of each method. The plot embeddings are visualized using UMAP (as described in Chapter 4.1), using 2 components.

5.1 TF-IDF Results

The visualization in 5.2 is generated by using UMAP with 2 components on the original TF-IDF embeddings. The colors in Figure 5.2a represent the 19 true genres, while the colors shown in Fgure 5.2b are assigned by performing spectral clustering on the 200-component UMAP generation.

While a few areas of homogeneous genre appear in 5.2a, there is a lot of overlap throughout the space. Also, 5.2b shows that the sizes of the clusters are far from uniform, with one very large cluster (shown in bright orange) making up 31% of the points. There are also many groups of outliers around the perimeter – the 10 smallest clusters make up less than 15% of the points. These findings suggest that the overall TF-IDF and UMAP embedding is not ideal for this task and that spectral clustering is not performing as well as expected.



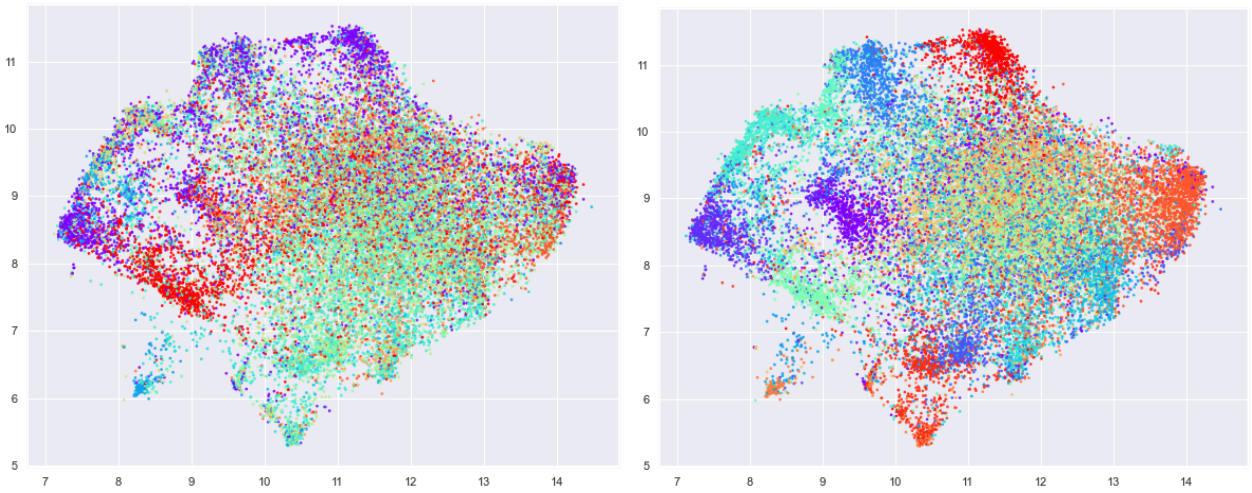
(a) Colored by Genre

(b) Colored by Cluster

Figure 5.2: TF-IDF Results

5.2 Doc2Vec Results

Compared to TF-IDF, the Doc2Vec results have more areas where plots of the same genre are grouped close together in the vector space (Figure 5.3a). These areas also correspond better to the assigned clusters shown in Figure 5.3b. This illustrates that the Doc2Vec results are superior to TF-IDF, confirming what the NMI score indicated.



(a) Colored by Genre

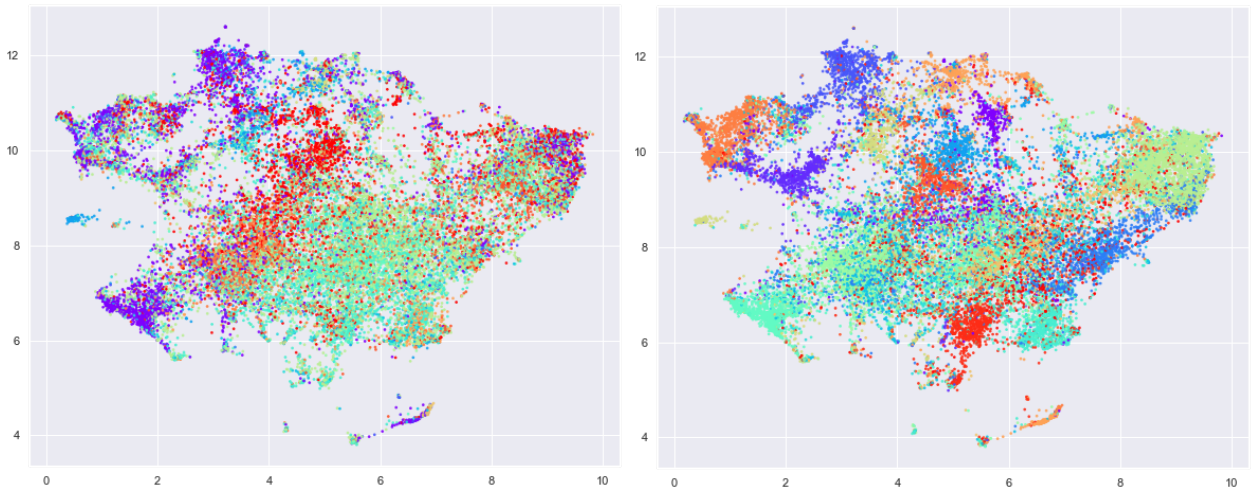
(b) Colored by Cluster

Figure 5.3: Doc2Vec Results

5.3 S-BERT Results

Like the Doc2Vec results, the S-BERT results shown in Figure 5.4 indicate performance superior to TF-IDF. Compared to TF-IDF, there are more areas where plots of the same genre are close together, which enables better spectral clustering results.

One interesting point is that the S-BERT results show more empty space and a less smooth perimeter compared to Doc2Vec, implying greater separation of neighborhoods. The NMI scores of Doc2Vec and S-BERT are quite similar, but it is possible that the S-BERT results are more useful for finding films most similar to a given film of interest. This is a subjective task without a clear way to evaluate performance using the data at hand. Nevertheless, the next chapter presents some case studies to help explore this question.



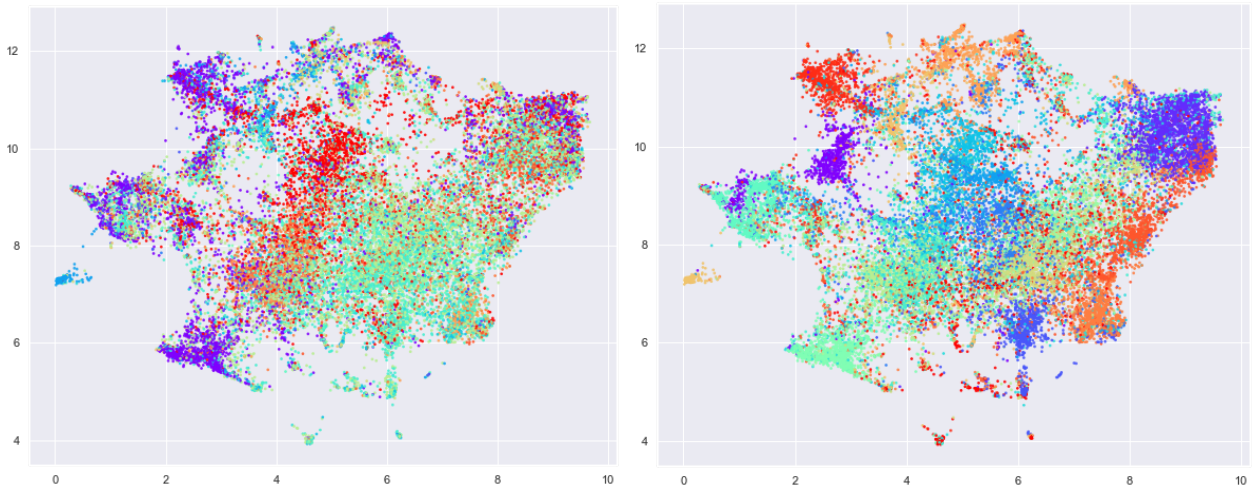
(a) Colored by Genre

(b) Colored by Cluster

Figure 5.4: S-BERT Results

5.4 Experiment: Combining Doc2Vec and S-BERT

Because Doc2Vec and S-BERT have similar NMI scores (Table 5.1) but still contain different information (based on their mutual NMI of 0.2489), combining the results of these two methods could be interesting. The embeddings were concatenated to create a 968-dimensional vector for each film. Spectral clustering was then applied in the same way as described above. Despite the additional information, the NMI between the clusters and the true genres was 0.1351, only a slight improvement. More improvement may have been achieved by using another method of combining the embeddings and selecting features, such as PCA. However, predicting genre is not the main goal of this paper and is only a proxy for model performance.



(a) Colored by Genre

(b) Colored by Cluster

Figure 5.5: Doc2Vec & S-BERT Combined Results

CHAPTER 6

Case Studies

This section will finally put the models into action and discuss specific results. For each model, we return the 10 embedded vectors closest to the vector (movie plot) of interest. The distance between vectors is calculated using cosine similarity, as described in Chapter 2.3. Evaluating performance or accuracy on this task is quite subjective, but it is still an interesting exercise.

The first movie selected as a target is *Casablanca*, the renowned 1942 romantic drama set during World War II. The models return a number of films that are also from the mid-1900s, as well as films set during wartime. This is more true of Doc2Vec and S-BERT, confirming that these embeddings likely better represent the films. The second recommendation from S-BERT, *Before Sunset*, is also interesting as it involves a couple who meets again after spending years apart, similar to Humphrey Bogart and Ingrid Bergman's characters in *Casablanca*.

Table 6.1: Top 10 Most Similar to Casablanca

Target: Casablanca (1942)			
	TF-IDF w/ UMAP	**Doc2Vec**	**S-BERT**
1	Love Letters (1945)	A Song to Remember (1945)	Cornered (1945)
2	Suite Française (2015)	The Rich Are Always with Us (1932)	Before Sunset (2004)
3	Sherlock Holmes (1916)	The Iron Duke (1934)	An American in Paris (1951)
4	Stronger Than Desire (1939)	Orders to Kill (1958)	Play It Again, Sam (1972)
5	Poison Pen (1939)	Days of Glory (1944)	Head in the Clouds (2004)
6	The W Plan (1930)	This Was Paris (1942)	Hotel Reserve (1944)
7	The Day Will Dawn (1942)	The Right Person (1955)	Secret Mission (1942)
8	Neutral Port (1940)	Up from the Beach (1965)	Midnight in Paris (2011)
9	The Captive Heart (1946)	South Sea Woman (1953)	The Blood of Others (1984)
10	The Lost Moment (1947)	British Agent (1934)	The Shopworn Angel (1938)

The second target movie selected is *Ferris Bueller's Day Off*, the classic 1980s John Hughes film in which a high school senior concocts a plan to cut class with two friends. They borrow a Ferrari and spend the day enjoying the city of Chicago. In the results below, S-BERT outperforms the other two models. Several high school films from the 1980s make the list, including *Adventures in Babysitting*, *License to Drive*, and *Risky Business*. The recommendations from TF-IDF and Doc2Vec do not make quite as much sense. For example, *Another Me* is about a teenager in high school, but it is a psychological thriller where Doppelgänger stalks the main character. *American Pastoral* is a crime drama that also has dark themes.

Table 6.2: Top 10 Most Similar to Ferris Bueller's Day Off

Target: Ferris Bueller's Day Off (1986)			
	TF-IDF w/ UMAP	**Doc2Vec**	**S-BERT**
1	Another Me (2013)	American Pastoral (2016)	Adventures in Babysitting (1987)
2	Over the Edge (1979)	Dirty Girl (2010)	License to Drive (1988)
3	Bridge to Terabithia (2007)	The Happening (2008)	Risky Business (1983)
4	Welcome to the Dollhouse (1995)	Rushmore (1998)	Aloha, Bobby and Rose (1975)
5	Stuck (2007)	Amreeka (2009)	Scent of a Woman (1992)
6	Rita, Sue and Bob Too (1986)	Focus (2015)	The Final Destination (2009)
7	Planes, Trains and Automobiles (1987)	Superstar: The Karen Carpenter Story (1987)	Uncle Buck (1989)
8	Hope Floats (1998)	Mr. Peabody & Sherman (2014)	Hot Rods to Hell (1967)
9	The Whole Nine Yards (2000)	Love Finds Andy Hardy (1938)	Dead-End Drive In (1986)
10	Youth in Revolt (2010)	The Sisterhood of the Traveling Pants 2 (2008)	Corvette Summer (1978)

The third target selected is *Sleepless in Seattle*, a romantic comedy that tells the story of a woman who falls in love with a recently widowed man after hearing him call in to a radio station. The film follows the two characters and their separate lives until they finally meet in New York City at the end of the film. Here, the first recommendation from S-BERT and TF-IDF is *An Affair to Remember*, which is the film that inspired *Sleepless in Seattle*. TF-IDF also includes *Love Affair*, in which two people meet and fall for each other during an airplane flight gone wrong. They agree to meet in New York City in a few months to see if their attraction lasts.

Doc2Vec's first recommendation, *Love on a Diet*, is a Chinese comedy that is only similar in the sense that it involves a romance and the anticipation of a couple meeting at the end of the film. The second recommendation is the romantic comedy *Definitely, Maybe*. S-BERT's recommendations also include romantic comedies *She's Out of My League* and *While You Were Sleeping*.

Table 6.3: Top 10 Most Similar to Sleepless in Seattle

Target: Sleepless in Seattle (1993)			
	TF-IDF	**Doc2Vec**	**S-BERT**
1	An Affair to Remember (1957)	Love on a Diet (2001)	An Affair to Remember (1957)
2	Elevator (2012)	Definitely, Maybe (2008)	She's Out of My League (2010)
3	Love Affair (1994)	Before Sunset (2004)	The Second Face (1950)
4	Finding Mr. Right (2013)	Everything, Everything (2017)	While You Were Sleeping (1995)
5	London, Paris, New York (2012)	Where the Spies Are (1965)	The Break-Up (2006)
6	The Death of Poe (2006)	Night Moves (1975)	Dear Heart (1964)
7	Elevated (1997)	Can a Song Save Your Life? (2013)	Unfaithful (2002)
8	Chico and Rita (2010)	Dostana (2008)	White Palace (1990)
9	The Weather Man (2005)	Yasmin (2004)	Cross Country Cruise (1934)
10	Left Behind: The Movie (2000)	The Bourne Ultimatum (2007)	My Reputation (1946)

The final target film is *Get Out*, the acclaimed horror film that follows an African-American man who visits the family of his white girlfriend. The film touches on themes of racism and interracial relationships entwined in a gripping horror story. Interestingly, TF-IDF does not return any films in the horror/thriller genre. The Doc2Vec results include a few thrillers, including *House at the End of the Street*, *Contracted*, and *Paranormal Activity 2*. However, aside from genre, these films do not have many similarities with *Get Out*.

Looking at the S-BERT results, there are a few recommendations that do have similar themes. *Lakeview Terrace* is a thriller that involves an interracial couple that moves to a suburban neighborhood, where their next-door neighbor is a menacing cop. On the surface, it does seem to have some similarities to elements of *Get Out*. Other films on the list also explore racial themes, including *Freedomland*, *Devil in a Blue Dress*, *Watermelon Man*, and *Far from Heaven*. The Telugu horror film *Kshanam* is an interesting recommendation and one of only two foreign-language films listed, but it does not appear to have much in common with the target.

Table 6.4: Top 10 Most Similar to Get Out

Target: Get Out (2017)			
	TF-IDF	**Doc2Vec**	**S-BERT**
1	Sapphire (1959)	House at the End of the Street (2012)	Freedomland (2006)
2	Rosewood (1997)	Get on Up (2014)	Lakeview Terrace (2008)
3	Pressure (1976)	The Watermelon Woman (1996)	Kshanam (2016)
4	Nothing But a Man (1964)	Contracted (2013)	Devil in a Blue Dress (1995)
5	Crash (2004)	Agatha (1979)	No Way Out (1950)
6	Black Like Me (1964)	I Saw What You Did (1965)	Watermelon Man (1970)
7	Murder in Harlem (1935)	Back from the Dead (1957)	Hysteria (1965)
8	Higher Learning (1995)	Bahar (1951)	Stir of Echoes (1999)
9	The Watermelon Woman (1996)	Paranormal Activity 2 (2010)	Far from Heaven (2002)
10	The Black Stallion (1979)	One Mile from Heaven (1937)	Amos & Andrew (1993)

CHAPTER 7

Conclusion

It is clear that both the Doc2Vec and S-BERT models outperform TF-IDF based on the case studies and the NMI scores when comparing clusters with the true genres. Also, upon examining the case studies, it appears that the S-BERT embeddings make the most sense when it comes to plot similarity within a small neighborhood. Even though Doc2Vec scored similarly to S-BERT on NMI, the specific S-BERT recommendations discussed appear more related to the target film. It is possible that, although the overall structure of the two embeddings is comparable, S-BERT generates more accurate information at a local level, resulting in better recommendations.

In summary, the models discussed in this paper, particularly S-BERT, do a relatively good job of generating numerical representations of the movie plot descriptions. Obviously, the results cannot compare to the recommendations offered by companies like Netflix or Amazon, which harness an incredible amount of information, from viewing and rating data across users to movie popularity and individual preferences. However, the goal of this paper was to explore methods of identifying films that are similar based solely on their plot content. Although the resulting recommendations are less accurate, this opens the door for discovering lesser-known films that may be of interest to the viewer.

7.1 Future Analysis

Evaluating the quality of the numeric plot embeddings is a challenge, and the case studies discussed here are clearly subjective. This paper aims to use the genre labels, spectral clustering, and NMI to create a measure of quality. However, other methods of model comparison may be more effective, and future research could explore more sophisticated techniques. For example, the embeddings could be compared in their performance on a downstream task, such as a neural network classifier designed to predict genre.

In evaluating word-level embeddings, researchers have found that crowdsourcing word-relatedness tasks to gather data on model performance is superior to other methods, such as comparing effectiveness on downstream tasks [24]. In the case of movie plots, designing an analogous task for crowdsourcing poses a challenge, as nobody has seen every movie and may not be able to make a quick judgment of plot relatedness. However, data containing film ratings from many users could be utilized in a similar way. This data could help shed light on which model generates local neighborhoods most consistent with user preferences.

It is also worth noting that the quality of the embedded vectors depends on the quality of the plot descriptions used as input. The Wikipedia plot descriptions vary widely in terms of length and detail – more popular movies tend to have better plot summaries. This leads to poorer vector representation of lesser-known films. In addition, the films included only represent a small slice of the films in existence released through 2017. Obtaining higher-quality data would improve the results.

Additional research could also focus on creating a pre-trained model based on the transformer architecture that is intended for longer documents. The S-BERT model used in this paper was designed for sentences and short paragraphs, and it still performs relatively well compared to the other two models. But creating a similar model specifically for longer documents could greatly improve the results and would have many additional use cases beyond the scope of this project.

Bibliography

[1] Karen Sparck Jones. "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". In: *Journal of Documentation* 28.1 (Jan. 1972), pp. 11–21.

[2] Quoc Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053 (2014). arXiv: 1405.4053. URL: http://arxiv.org/abs/1405.4053.

[3] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: (2019). DOI: 10.48550/ARXIV.1908.10084. URL: https://arxiv.org/abs/1908.10084.

[4] Justin R. *Wikipedia Movie Plots*. 2019. URL: https://www.kaggle.com/datasets/jrobischon/wikipedia-movie-plots.

[5] J. Ben Schafer et al. "Collaborative Filtering Recommender Systems". In: *The Adaptive Web: Methods and Strategies of Web Personalization*. Ed. by Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 291–324. ISBN: 978-3-540-72079-9. DOI: 10.1007/978-3-540-72079-9_9. URL: https://doi.org/10.1007/978-3-540-72079-9_9.

[6] Steve Lohr. "A $1 Million Research Bargain for Netflix, and Maybe a Model for Others". In: *The New York Times* (Sept. 2009). URL: https://www.nytimes.com/2009/09/22/technology/internet/22netflix.html.

[7] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2018. DOI: 10.48550/ARXIV.1802.03426. URL: https://arxiv.org/abs/1802.03426.

[8] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Andrew Ng, Michael Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an algorithm". In: *Advances in Neural Information Processing Systems*. Ed. by T. Dietterich, S. Becker, and Z. Ghahramani. Vol. 14. MIT Press, 2001. URL: https://proceedings.neurips.cc/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf.

[10] Ulrike von Luxburg. "A Tutorial on Spectral Clustering". In: *CoRR* abs/0711.0189 (2007). arXiv: 0711.0189. URL: http://arxiv.org/abs/0711.0189.

[11] David Williamson. *ORIE 6334 Spectral Graph Theory Lecture 7*. Sept. 2016. URL: https://people.orie.cornell.edu/dpw/orie6334/Fall2016/lecture7.pdf.

[12] Sunilkumar P. and Athira P. Shaji. "A Survey on Semantic Similarity". In: *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)*. 2019, pp. 1–8. DOI: 10.1109/ICAC347590.2019.9036843.

[13] Matthew Honnibal and Ines Montani. "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing". To appear. 2017.

[14] Richard Bellman. *Dynamic Programming*. Princeton, New Jersey: Princeton University Press, 1957. URL: https://gwern.net/doc/statistics/decision/1957-bellman-dynamicprogramming.pdf.

[15] Tim Sainburg, Leland McInnes, and Timothy Q. Gentner. "Parametric UMAP Embeddings for Representation and Semisupervised Learning". In: *Neural Computation* 33.11 (2021), pp. 2881–2907.

[16] Leland McInnes et al. *UMAP*. 2018-2022. URL: https://github.com/lmcinnes/umap/blob/master/doc/index.rst.

[17] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781.

[18] Guanjin Wang and Stephen Wai Hang Kwok. "Using K-Means Clustering Method with Doc2Vec to Understand the Twitter Users' Opinions on COVID-19 Vaccination". In: *2021 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*. 2021, pp. 1–4. DOI: 10.1109/BHI50953.2021.9508578.

[19] Frederic Morin and Y. Bengio. "Hierarchical probabilistic neural network language model". In: *AISTATS 2005 - Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics* (Jan. 2005).

[20] Shuzhan Fan. *Understanding Word2Vec and Doc2Vec*. Aug. 2018. URL: https://shuzhanfan.github.io/2018/08/understanding-word2vec-and-doc2vec/.

[21] Radim Rehurek and Petr Sojka. "Gensim–python framework for vector space modelling". In: *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3.2 (2011).

[22] Omer Levy and Yoav Goldberg. "Dependency-Based Word Embeddings". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 302–308. DOI: 10.3115/v1/P14-2050. URL: https://aclanthology.org/P14-2050.

[23] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[24] Tobias Schnabel et al. "Evaluation methods for unsupervised word embeddings". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 298–307. DOI: 10.18653/v1/D15-1036. URL: https://aclanthology.org/D15-1036.