Creating the Foundation of a Movie Search Engine

Alexandra Pfleegor

***Abstract*—In this paper, I will discuss the process of creating the building blocks for a search engine for movies. First, we will look at related work before moving on to the methodology for my project. Specifically, we will discuss the data collection process and the process of building the models needed for the search engine. Following that, we will look at the results from the model and discuss how it could be used to eventually create a movie search engine. Finally, we conclude the paper with a conclusion discussing possible future work and how parts of the model can be used for other work.**

***Index Terms*—search engine, named-entity recognition, topic modeling**

I. Introduction

For this project, we aim to create the building blocks for a movie search engine. Since building an entire search engine is too large of a scope, we create models to help with parts of the search process, such as finding the keywords and topics from movie plot summaries. The process of creating parts of the search engine can also bring insight into how major search engines operate. The models that we create can also be used in other ways, such as classifying the genre of movie or predicting the IMDb rating based on the extracted keywords.

II. Related work

Much of the existing research within this subset of Natural Language Processing has to do with recommendation systems, generating summaries, classification of genres based on summaries or reviews and predicting a movie's success based on the plot.

A. Recommendation Systems

Recommendation systems are a popular subject of research because they have become an integral part of companies', such as Netflix, business models. One main reason that users like Netflix is that it can recommend movies for them to watch based on their previously indicated preferences. Any way to increase the performance of these algorithms could help sell more subscriptions.

The main types of recommendation systems are collaborative filtering systems, content-based systems and hybrid systems. For the purposes of Natural Language Processing, most researchers focus on content-based systems.

In Shi et al. 's 2013 paper, they created a recommendation system that added contextual information, specifically movie plot keywords and mood tags, in their algorithm to decide what to recommend [1]. Often, movie recommendations are based on what similar users have also liked. This paper, instead, tries to recommend movies on similar topics to movies that a user has previously liked. Shi et al. show that their algorithm that takes into account this contextual information improves the recommendation system more than other algorithms that do not include contextual information.

Other papers, such as Reddy et al. 's 2019 paper, use different features to help improve the recommendation algorithm such as the genre of movies that a user watches [2] For this particular paper, if a user rates a certain movie high, then they are recommended movies within that same genre.

The research area of topic modeling has also been used to improve on recommendation systems. Bergamaschi et al. looks to create a movie recommendation system when there is little to no information about user preferences [3]. They compare Latent Semantic Allocation (LSA) and Latent Dirichlet Allocation (LDA), which are both popular topic modeling algorithms. Both algorithms create a document by topic matrix. They find that LSA performed better than LDA in trying to recommend movies with similar plots.

Bhattacharya and Lundia's paper also uses plots to create a recommendation system [4]. Their work specifically is very similar to our current work because they use count vectorization and cosine similarity to calculate the similarity between movie plots. However, in their case, they then use this information to recommend other movies instead of using it to match search queries.

Another subset of recommendation system research is on how to improve the efficiency of models. Wang et al. 's 2014 paper helps to improve performance on the basic model by helping to reduce the computational complexity of the model by using K-means clustering [5]. They find that their model improves performance in terms of giving better recommendations to the users.

B. Summary Generation

Generating summaries is not as big of a research problem in movies as it is in things like news stories. However, there has been some research in this topic that can also lend itself to other areas of interest. Li et al. 's 2017 paper deals with attempting to summarize a long movie plot by matching sentences of the plot on Wikipedia with shots in the movie [6].

Another related topic has been movie review summarization, as seen in Khan et al. 's 2020 paper [7]. This helps a person quickly see the good and bad things about a movie so they can decide whether to see it. The summary, on the other hand, is just an overview of the story. In their paper, Khan et al. use a weighted graph-based ranking algorithm to rank the sentences in a movie review. They then use these ranks to create an extractive summary.

C. Genre Classification

The classification of movie genres can be used in other algorithms, such as helping a movie recommendation system. Ertugrul and Karagoz's paper uses bidirectional LSTM to classify movies based on their plot summaries [8]. They find that Bi-LSTM performs better than a basic logistic regression model and basic recurrent neural networks.

D. Predicting Success

Finally, Lee et al. 's 2020 paper creates a model to predict the popularity of a movie purely based on the plot summary [9]. They do this using data on movies from 2000 to 2018. This direction of NLP research could potentially help movie studios in weeding out the movie scripts that have little chance of succeeding to save the time of the Script Readers.

III. Methodology

A. Data Collection

We collected the data using the Open Movie Database (OMDb), since it is relatively easy to use and user friendly. It also provides many details for each movie, including the title, year of release, the Motion Picture Association film rating (G, PG, PG-13, R), the release date, runtime, genre, director(s), writer(s), actors, a brief description of the plot, language, a list of awards (specifically the number of Oscars won and the number of total wins and nominations), a link to the movie poster, a list of ratings from Internet Movie Database (IMDb), Rotten Tomatoes and Metacritic, a metascore, IMDb rating, number of IMDb votes, the IMDb ID for the movie, the type of product (movie or tv show), DVD release date, box office gross revenue, production studio, and website for the movie.

To limit the runtime and processing power needed to run the code, we only used a limited subset of movies to test our model on. Specifically, we chose to include all Disney animated movies (except for 7 that did not return a result) and Marvel movies. However, the model should work on a wider set of movie data.

B. Development of Methods

The main goals for this implementation is to extract keywords / topics that are likely to come up in a search query. Specifically, we aim to create a search engine that works for people who do not remember the title or all of the details of the movie.

IV. Results and Analysis

Our final result is a `pandas` DataFrame with 10 different columns. In figure [1], we can see the first 5 rows of the resulting dataframe from implementing our code on the Disney dataset.

A. Title

The first just includes the title of the film. Since the end goal of this project is to create a movie search engine, we want to provide the link to the IMDb webpage for the movie if the person enters the full title.

	title	year	decade	genre	actors	named entities	director	plot topics	tfidf vector	production
0	Academy Award Review of Walt Disney Cartoons	1937	1930	[anim, short, comedi]	[billy bletcher, dorothy compton, eddie holden]	[Disney, Snow White, Seven Dwarfs]	[n/a]	[compilation, short, release]	(0, 8)\t0.26253361164209704\n (0, 863)\t0.2...	[na]
1	Snow White and the Seven Dwarfs	1937	1930	[anim, famili, fantasi]	[adriana caselotti, harry stockwell, lucille l...	[Snow White, Magic Mirror, Doc, Sneezy, Grumpy...	[william cottrell, david hand, wilfred jackson]	[charm, creature, kingdom, stepmother, forest,...	(0, 226)\t0.13579487920663044\n (0, 449)\t0...	[na]
2	Pinocchio	1940	1940	[anim, adventur, comedi]	[dickie jones, christian rub, mel blanc]	[Inventor, Gepetto, Pinocchio, Jiminy Cricket,...	[norman ferguson, t. hee, wilfred jackson]	[marionette, wish, boy, fairy, conscience, fro...	(0, 2707)\t0.14607432597135594\n (0, 1067)\...	[walt, disney, productions]
3	Fantasia	1940	1940	[anim, famili, fantasi]	[leopold stokowski, deems taylor, the philadel...	[Disney, Western, Leopold Stokowski, Philadelp...	[james algar, samuel armstrong, ford beebe jr.]	[animator, picture, music, magician, limit, st...	(0, 718)\t0.07697953789902512\n (0, 1977)\t...	[na]
4	The Reluctant Dragon	1941	1940	[anim, comedi, famili]	[robert benchley, frances gifford, buddy pepper]	[Humorist, Robert Benchley, Walt Disney, Walt ...	[alfred l. werker, hamilton luske, jack cutting]	[story, dragon, poetry, way, tour, animation, ...	(0, 2257)\t0.17626133935877134\n (0, 139)\t...	[walt, disney, productions]

Fig. 1. Resulting DataFrame for Disney dataset

B. Year and Decade

The next two columns are correlated: the year of release and decade of release. Since most people might not remember the exact year that *Snow White and the Seven Dwarfs* was released, the release year is most likely not as important as the general time period of release. For example, if someone is looking for a movie that they remember came out in the 1990s, we want to be able to include movies from the 1990s higher in our eventual results ranking algorithm than movies from other eras. The specific year is also included in case the person remembers the exact year of release.

C. Genre

The next column includes the different genres of the movie. Most films are tagged with multiple genres, since they can fit into many different categories. For example, *Iron Man* is tagged as action, adventure and scifi. The output specifically uses stemming in their list of genres so that things such as "animated" and "animation" will be equal.

D. Actors and Director

The following column includes a list of the main actors in the movie. A later column also includes a list of directors. These two are important because someone could only remember a random actor that is in a movie (or the director) along with a few plot points. We want to be able to find the resulting movie in this case, even with limited information.

E. Production

Another column that is less relevant for the search engine is the production company for the movie. Although this would be beneficial, many movies have "N/A" listed under this column. However, if the data is complete, knowing that a movie is a Disney film can help to greatly reduce the number of possible search results.

F. Named Entities

The final three columns make up the bulk of the project. They are the named entities, plot topics and Tf-idf (term frequency - inverse document frequency) representation of the given plot.

The named entity column uses a sub topic of Natural Language processing known as named-entity recognition (NER). This uses part of speech (pos) tagging and a database of entities to extract entities from a given sentence, as well as

labeling the entity with an appropriate label (person, organization, product, geopolitical entity, etc.).

In our project, we tested four different NER models before deciding on a final one to include in our output. Three of the models were different variations of spacy's NER model. Specifically, we tried the "`en_core_web_sm`", "`en_core_web_md`" and "`en_core_web_lg`" models. The main difference between these three models is the size of them and thus the accuracy, precision and recall of the models. The larger model tends to have better end metrics than smaller models, but it takes a lot longer to load the larger model. The final NER model that we tested is one included in python's NLTK library: `ne_chunk()`. Although this model does not capture as much of a variety of items (such as time, products, dates, works of art and cardinal numbers), it consistently identified the main characters of a movie.

The most telling example showing that NLTK's `ne_chunk()` is better for our purposes is with *Snow White and the Seven Dwarfs*. Below, we can see the plot summary from IMDb.

> The beautiful and kindhearted princess Snow White charms every creature in the kingdom except one - her jealous stepmother, the Queen. When the Magic Mirror proclaims Snow White the fairest one of all, she must flee into the forest, where she befriends the lovable seven dwarfs - Doc, Sneezy, Grumpy, Happy, Bashful, Sleepy, and Dopey. But when the Queen tricks Snow White with a poisoned apple and falling into a deep sleep, only the magic of true love's kiss can awaken her.

We can see the results of the NER models on this plot in table [I], where `GPE` stands for geopolitical entity.

TABLE I
NAMED ENTITIES FOR SNOW WHITE AND THE SEVEN DWARFS

model	person	GPE	other
en_core_web_sm	Snow White Dopey Queen	Sneezy Grumpy Happy Bashful Sleepy	one the Magic Mirror seven
en_core_web_md	Snow White Dopey	kingdom	one seven
en_core_web_lg	Snow White Queen	kingdom	one the Magic Mirror seven
ne_chunk()	Snow White Doc Happy Bashful Dopey	Sneezy Grumpy Sleepy	Magic Mirror

For spacy's models, `en_core_web_sm` model only identifies 6 out of the 7 dwarfs, `en_core_web_md` only identifies 1 out of the 7 dwarfs, and `en_core_web_lg` does not identify any of the 7 dwarfs. In comparison, `ne_chunk()` identifies all 7 dwarfs as named entities. Although we are not able to identify all of the dwarfs as people, we are able to have them all listed as entities with `ne_chunk()`.

G. Plot Topics

The next column in our results dataframe is one that includes potential plot topics. Currently, this is limited to only nouns, but we can expand it to other parts of speech. However, since the plots are relatively short, it is difficult to do traditional topic modeling. Thus, our model extracts potential key words from the plot, instead of traditional topic modeling.

We tried using a Hierarchical Dirichlet Process (HDP) for topic modeling, but the resulting topics were not significant in terms of potential search terms. Since the plots are all relatively short, the number of potential related terms for a specific topic is very small. For our example, we will use the movie *Frozen*, since its plot is relatively long compared to other plots. Below, we can see the plot summary of *Frozen*.

> Fearless optimist Anna teams up with rugged mountain man Kristoff and his loyal reindeer Sven and sets off on an epic journey to find her sister Elsa, whose icy powers have trapped the kingdom of Arendelle in eternal winter. Encountering Everest-like conditions, mystical trolls and a hilarious snowman named Olaf, Anna and Kristoff battle the elements in a race to save the kingdom. From the outside Elsa looks poised, regal and reserved, but in reality she lives in fear as she wrestles with a mighty secret: she was born with the power to create ice and snow. It's a beautiful ability, but also extremely dangerous. Haunted by the moment her magic nearly killed her younger sister Anna, Elsa has isolated herself, spending every waking minute trying to suppress her growing powers. Her mounting emotions trigger the magic, accidentally setting off an eternal winter that she can't stop. She fears she's becoming a monster and that no one, not even her sister, can help her.

We can see the the probability distribution for the first 3 topics generated using the top 7 words from HDP in table [II]. Looking at the top 7 words for each topic, we can see that they are not necessarily interconnected in any discernible way.

Thus, instead we just used the list of words created from the pre-processing for HDP. This includes separating words into tokens, lemmatizing the tokens and creating a list of the nouns from these tokens. Since the plot given in the data is already a summary, the nouns in the plot are most likely key words that could be used to search for the movie. Therefore, for *Frozen*, our resulting list for plot topics is

H. Tf-idf Vector Representation

The final column in our dataframe is the tf-idf (term frequency-inverse document frequency) vector column. This column includes the tf-idf vector representation of each plot. For preprocessing, we removed all punctuation and stopwords from each plot, as well as tokenizing and lemmatizing the plots. We use the list of all plots as our corpus for our tf-idf model. By having the tf-idf vector forms of all of the

TABLE II
PROBABILITY DISTRIBUTION FOR 3 TOPICS FROM FROZEN

topic number	top 7 words	probability
0	magic	0.44298
	sister	0.44298
	snowman	0.00439
	monster	0.00439
	emotion	0.00439
	minute	0.00439
	moment	0.00439
1	winter	0.44298
	ability	0.44298
	snowman	0.00439
	monster	0.00439
	emotion	0.00439
	minute	0.00439
	moment	0.00439
2	kingdom	0.1913
	optimist	0.1913
	team	0.1913
	mountain	0.1913
	icy	0.1913
	moment	0.00189
	race	0.00189

optimist	team	mountain	man	reindeer
journey	sister	icy	power	kingdom
winter	condition	troll	snowman	battle
element	race	reality	fear	secret
ice	snow	ability	moment	magic
minute	emotion	monster		

plots already, we hope to reduce the runtime when it comes to comparing plots to the search query.

We included in our final project the methods needed to find the plot that best matches our search query using cosine similarity. For example, if our search query is "olaf princess sister", the model will find the indices with the highest cosine similarity. In future implementations, these cosine similarities can be used to rank search results. In table [III], we see the results of the search query "olaf princess sister".

TABLE III
RESULTS OF SEARCH QUERY "OLAF PRINCESS SISTER"

index	title	cosine similarity
1	Snow White and the Seven Dwarfs	0.047029
19	Sleeping Beauty	0.091166
39	Aladdin	0.045075
61	Lilo & Stitch	0.034981
78	Enchanted	0.042968
86	The Princess and the Frog	0.041429
89	Tangled	0.027029
95	Brave	0.062808
101	Frozen	0.189729
119	Frozen II	0.136890

As we can see, both *Frozen* movies have the highest cosine similarities, which is what we want. This method helps to bring together named entities and our topic modeling to match search queries to the corresponding movie plots.

For example, in figure [2], we can see the different tokens from *Frozen*'s plot where the size of each token represents the number of times that it is in the plot. This shows that the words "Anna", "Elsa" and "sister" show up the most within

Fig. 2. Word Cloud for the plot of Frozen after removing stopwords

the plot summary. Thus, a search query with these three terms will likely result in a higher cosine similarity for the movie *Frozen*.

V. CONCLUSION AND FUTURE WORK

Since our model only has the first steps to creating a movie search engine, there are many ways that we can expand on this project in the future, with the main way being creating the final movie search engine. However, there are many smaller steps in between our model and the final implementation.

One potential thing to add to our current model is vector semantics so things with similar meanings can be equivalent when searching. For example, when searching for the movie *Tangled*, we might want to use the word "thief" to describe Flynn Rider. However, in the plot, Flynn Rider is described as a bandit. Thus, an addition to make our model better is to add a way to make it so that search terms with similar meanings can lead to similar results.

Another addition to the model that would be beneficial is to add minimum edit distance to account for misspellings in the search queries. For example, we want the query "Idina Mensel" to match the actress "Idina Menzel". So, adding minimum edit distance to correct these spelling mistakes will help us improve our overall search engine.

In terms of creating the final search engine, future work should focus on creating or implementing a page rank algorithm. The model should parse through the query, first potentially checking if the title is located in the query.

Then, we should parse through the query again to look for years or named entities. The years would help for the eventual page rank algorithm. We should consider years around the year provided, in case the user is incorrect about the release year. Also, the year should not be an eliminating factor for certain movies, since the user could be talking about a number or the movie is set in a certain year. This applies to all of the steps for our search engine, since we do not want to accidentally eliminate the movie that a user is searching for. Named entities should be checked against actors, directors, production studios and named entities.

Next, we should parse through the query looking for genres. We will need to use stemming on these genres to make sure that things like "animated" and "animation" match.

Finally, we should go through the task of checking the cosine similarity between the plot and the query, after removing stopwords.

REFERENCES

[1] Y. Shi, M. Larson, and A. Hanjalic, "Mining contextual movie similarity with matrix factorization for context-aware recommendation," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 1, pp. 1–19, 2013.

[2] S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok, and B. Venkatesh, "Content-based movie recommendation system using genre correlation," in *Smart Intelligent Computing and Applications*. Springer, 2019, pp. 391–397.

[3] S. Bergamaschi, L. Po, and S. Sorrentino, "Comparing topic models for a movie recommendation system." in *WEBIST (2)*, 2014, pp. 172–183.

[4] S. Bhattacharya and A. Lundia, "Movie recommendation system using bag of words and scikit-learn," *Int J Eng Appl Sci Technol*, vol. 4, pp. 526–528, 2019.

[5] Z. Wang, X. Yu, N. Feng, and Z. Wang, "An improved collaborative movie recommendation system using computational intelligence," *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 667–675, 2014.

[6] X. Li, T. Utsuro, and H. Uehara, "Movie summarization based on alignment of plot and shots," in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2017, pp. 189–196.

[7] A. Khan, M. A. Gul, M. Zareei, R. Biswal, A. Zeb, M. Naeem, Y. Saeed, and N. Salim, "Movie review summarization using supervised learning and graph-based ranking algorithm," *Computational intelligence and neuroscience*, vol. 2020, 2020.

[8] A. M. Ertugrul and P. Karagoz, "Movie genre classification from plot summaries using bidirectional lstm," in *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*. IEEE, 2018, pp. 248–251.

[9] J.-H. Lee, Y.-J. Kim, and Y.-G. Cheong, "Predicting quality and popularity of a movie from plot summary and character description using contextualized word embeddings," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 214–220.