Topic Modeling for Sentiment Analysis

Michael Ackerman, Marlene Barajas, Jessalyn Johnson, Anvita Warty

ABSTRACT

Modern imperative languages like Python can be used to create programs that address natural language processing objectives like topic and sentiment analysis. Through the use of data mining techniques, intelligent systems can be built to extrapolate the sentiment of a given document based on its topic. In this paper, our goal is to predict the rating of a movie based on its movie description.

I. BACKGROUND

Data mining is a process and concept described in the larger study of data science. Data mining aims to discover new information within inputs of large datasets by recognizing and uncovering patterns. While data mining has multiple uses, its ability to discern patterns from large sets of input data make it the perfect candidate for the development of intelligent systems that can make predictions on unseen data. Data mining is essentially a two-step process. After understanding how data mining will be used for a given context or what goal data mining will fulfill, the input data must first be preprocessed or prepared, and a mathematical model, which will discover the patterns based on powerful statistical reasoning, must be chosen. The mathematical model can come from either supervised or unsupervised machine learning algorithms. Supervised algorithms use manually labelled training data to learn patterns. Unsupervised algorithms, however, learn patterns by inspecting and dissecting raw data. In most cases, the use of both algorithms creates robust data mining systems.

Natural language refers to the evolution of language that occurs naturally from its use everyday. The way that a language evolves differs depending from person to person, but it usually relaxes grammatical and syntactic rules. It is hard for artificially intelligent programs to emulate natural language for this reason. However, data mining techniques can provide the appropriate foundation needed for such artificially intelligent systems to be built. This paper will explore the ways that programmers can use libraries and resources available in Python to build a natural language processing system that can classify sentiment and model topics.

II. INTRODUCTION

When analyzing the sentiment of a text, the traditional approach would be to extract the most descriptive adjectives within the text. For example, analyzing “pizza is delicious and great” for sentiment analysis would use “delicious” and “great” to conclude that the text is positive. Of course, this assumes that the corpus used in this analysis labels those words as positive terms the same way we do in our natural language. This is why it is important to choose the correct corpus for sentiment analysis.

Our aim is to predict how the public feels towards different genres of movies. Since the text our expert system will analyze will always be explicitly about movies, the corpus the system uses comes from 50,000 IMDB movie reviews that have been labeled positive or negative depending on the reviewer's opinion towards whatever the movie was about. Following this line of reason, our system extracts topics from these reviews to make assumptions about what topics reviewers felt were either positive or negative; this is how the expert system will use topic modeling to help predict a sentiment towards new data.

III. CONCEPTS

A. Topic Modeling

Topic modeling is the unsupervised categorization of documents, texts, or data in general. It can be considered a form of unsupervised learning. This can be useful for saving the time a person would have spent on reading through lengthy texts. The value of being able to make conclusions about the subject matter of large amounts of data varies on the application. In some cases, topic modeling can point out the subject of an article, clarify a theme that a large amount of text shares, or even uncover the similarities between different sets of data.

B. Sentiment Analysis

Sentiment analysis can be used to discern the tone of a text. For example, one can conclude that the sentence “this movie is fantastic” shares a positive sentiment about a movie while the sentence “this movie is okay” shares a neutral sentiment. This type of analysis can be used by a company that wants to understand how the general population feels about their product. As long as they can access reviews about their product, they can use sentiment analysis to extrapolate the percentage of positive, negative, and neutral opinions that the reviews conveyed. While this might not be a thorough assessment of why a population holds a sentiment, this method of analyzing allows the analysis of larger sets of data.

While sentiment analysis be done in such a way that is applicable to any new data, for our application we will not be using any tools that target traditional connotative nouns and adjectives. Instead, we will be using topic modeling concepts to determine a sentiment. We will specifically be trying to avoid labelling words like “great” and “wonderful” positive or negative because movie reviews often include aspects of both. An overwhelmingly negative review might still include “the actor was great despite the movie,” and we don’t want it to cause confusion in our program.

C. Natural Language

It’s important to understand that natural language is constructed naturally among people and can differ from region to region. Therefore, in order to emulate natural language one must consider *which* natural language they are referring to. This should be factored into creating an expert system from the corpus that is used to train it, as this is what will teach the expert system what its natural language is. If you train a topic model using a dictionary of words that it will not encounter, it will have low coherence and will not make accurate predictions as a result.

IV. METHODS/ALGORITHMS

Python has been the preferred language to use in data science and machine learning because of the extended amount of libraries, modules, and toolkits that exist to aid these fields. These tools are the reason our project is implemented with Python.

B. Data Processing

Data processing is a series of operations performed on given data to reconstruct or classify information into a format that is more easily analyzed and processed by computers. The Natural Language Toolkit (NLTK) is open-source software that provides Python with modules that give our expert system the functionality to process the data in our movie reviews.

In our experiment, five data processing operations are performed before generating a model. *Tokenization* is used to reduce the text down to lowercase, unpunctuated words. Words that are *less than three* characters in length will be removed. *Stopwords*, words that are most commonly used in the language, will also be removed from the data. In English, these are words like "the," "is," or "and." Words are then *lemmatized*, which means they are converted to their first-person variants and verbs are made present tense. In this operation, words like "am," "are," and "is" will convert to "be." Words will also be *stemmed*. In the stemming process, words are reduced down to their root form. The word "harnesses" will be reduced to "harness."

After performing the following operations, the given sentence from our data "One of the other reviewers has mentioned that …" transforms into the series of words "review, mention, watch, episod, hook, right, …" While this could be used as our corpus already, we want to do a better job of processing words that might not be distinctive enough to result in valid topics. This entails removing words that appear in less than 15 documents, more than half of the documents, or, after the former two steps, are not within the 100000 most frequently used. Now these words are ready to be used as a Bag of Words corpus to generate an LDA model.

C. Latent Dirichlet Allocation

The Latent Dirichlet Allocation (LDA) is an unsupervised machine learning method that is used for topic modeling. It is based on the words contained within a document or input text file. For a given number of unknown topics specified by the programmer or user, the LDA's goal is to accurately map those topics to the provided documents. Realistically speaking, in everyday language, documents only map to one topic; however, the LDA considers each input document as a mixture of all possible topics. For example, imagine a set of sentences are written inside a document and is fed into an LDA algorithm. A possible output may look like the following: “Topic A: is 30% broccoli, 15% banana, 10% breakfast, 10% munching”. Despite the variety of topics displayed, the general consensus from the viewer reviewing this output will be that the topic of the given document is related to food.

As mentioned before, the programmer or user must specify the number of topics that is applicable to the set of documents. This number can neither be too big nor too small. For either extreme, the LDA will not be able to accurately nor effectively categorize documents into any of the topics. In fact, the end result might be a symmetric distribution of the topics to all the documents. A symmetric distribution means that for a given document, each topic has the same likelihood. While developing an expert system, this is not a favourable outcome. After specifying the number of topics, the algorithm will create a distribution of these unknown topics across all the documents, and it will also create a random distribution within a singular document. The randomness of the distribution can be controlled by a hyperparameter. In the event that the programmer has prior knowledge about the similarity of the documents, they can advantageously change this hyperparameter in order to maximize the accuracy of the algorithm.

An LDA works by creating two matrices: a document-topic matrix and term-topic matrix. After applying a random distribution of topics within a single topic, the algorithm analyzes the words within the document. The LDA will have to parse through a text document and sort its words with a probability score that represents the likelihood that it pertains to a specific topic. This likelihood is assigned based on two variables: how often the word occurs with respect to the topic across all documents and what topics are present within the given document. The reason why LDA is referred to as a form of unsupervised learning is because it uses clustering techniques to group together similar words. In the case of topic modeling, the similarity is the shared topic. This constitutes the term-topic matrix. Once this matrix is discovered, the documents can be reassessed based on this matrix. Depending on how frequently the document uses terms specified for a particular topic, the likelihood of the document being in that particular topic will be assigned. This will constitute the document-topic matrix, and it will eventually determine the output of the algorithm.

D. Sentiment Analysis

Our expert system only takes the top three topics returned as potential classifications for both positive and negative topics, meaning that the algorithm takes six topics in total. Then, the function that returns a sentiment finds the average of the top three topics in both categories. If the average of the top three positive topics is greater than the negative topics’, then the system predicts general positivity towards the movie description. If the inverse is true, then the system predicts general negativity.

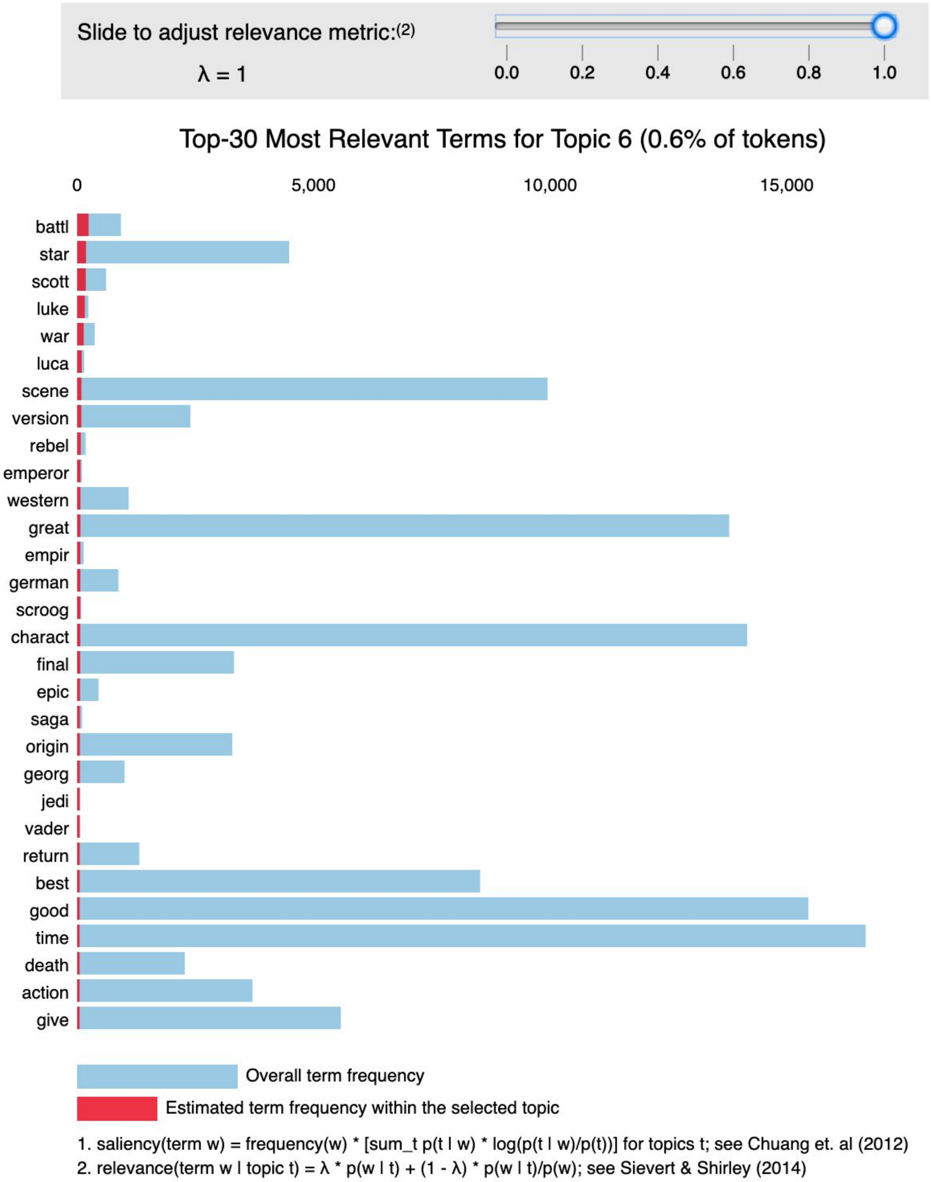
The function that returns a sentiment analysis in our expert system also returns a rating as a percentage. The closer that the two negativity and positivity scores are for a movie description, the lower it's prediction will be. This is because we assume that if a topic has scores that are near equal, then the opinions on the movie are very polar; if only half an audience likes something, then the score should be around 50%. Our expert system still only considers six different topics to calculate a percentage, so this calculation is done at the same time that the general sentiment is being determined.

V. EXPERIMENTS

Most of the experimentation throughout our project was regarding the creation of the LDA model that the expert system uses to classify movie descriptions. Using gensim's parallelized Latent Dirichlet Allocation class, we tinkered with its parameters in attempts to generate the most cohesive topics. With each experiment, our model always made 10 passes through the corpus during training and used 7 cores for parallel processing. The only variable to experiment with is the amount of topics we ask our LDA model to generate, but to do this we use visualization tools that help us understand our results.

pyLADvis is a Python library designed to help interpret topic modeling results. It helped us alter the relevance metric of our results so that we could see what direction we were working towards. In the remainder of this section, we will only be showing the experiments in which we generated 75 and 300 topics for understanding, but our tests included 15, 35, and 150 topics as well. One can access these missing test models in our code.

A. Relevance Metric


Slide to adjust relevance metric:(2)

λ = 1

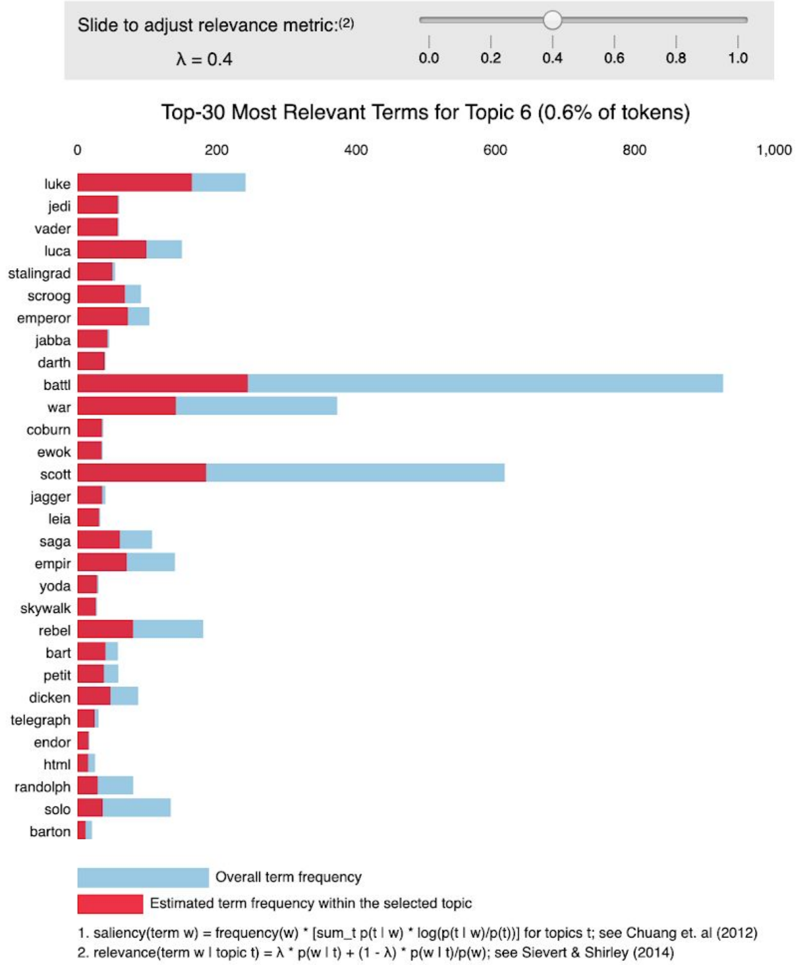
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

The picture above shows the most relevant terms for a selected topic. Blue bars represent the frequency of a term across the entire corpus, while red bars represent the frequency of a term for the selected topic. However, there are terms in the picture above which could be applicable to any topic, such as the terms “scene”, “character”, and “best”. These 3 words can probably be found in every other movie review and judging by the size of the blue bars, these words appear frequently throughout the corpus. We can filter these types of words by adjusting the *relevance metric* (lambda). By default, pyLADvis shows a relevance metric of 1 (lambda=1), which applies no filtering. Lowering the lambda values will filter out terms that have a high frequency across the entire corpus, while exposing terms that have a high frequency for the given topic, but not necessarily a high frequency for the corpus. From the formula:

$$Relevance\ Score\ =\ \lambda\ \times\ P(w|t)\ +\ \frac{(1-\lambda)\times P(w|t)}{P(w)}$$

where $w = term$, t = topic, and P = probability

When lambda = 1 (the max), the equation reduces to P(w|t), which does no filtering. Lowering the lambda value (lambda < 1) effectively raises the visibility of the *unique* terms for the topic. (Note, term frequency-inverse document frequency, or TF_IDF, algorithm works in a similar way.) Below are the results of the same topic, but using a lambda value of 0.4.



Slide to adjust relevance metric:(2)

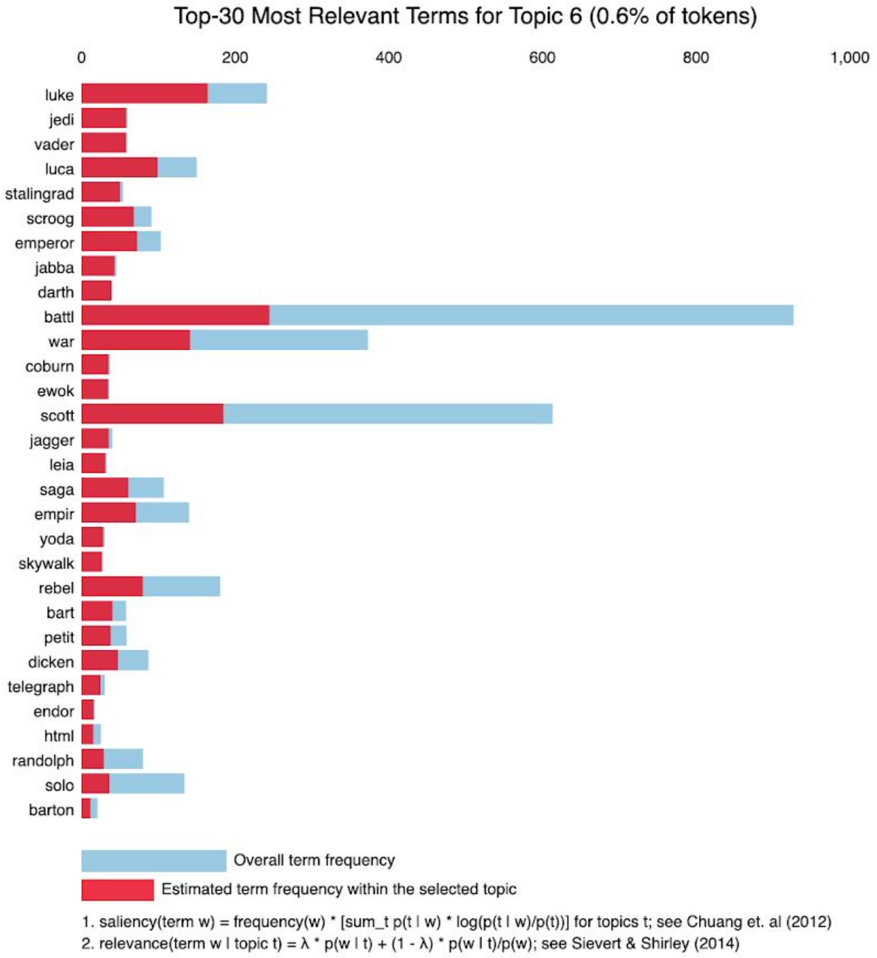
λ = 0.4

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

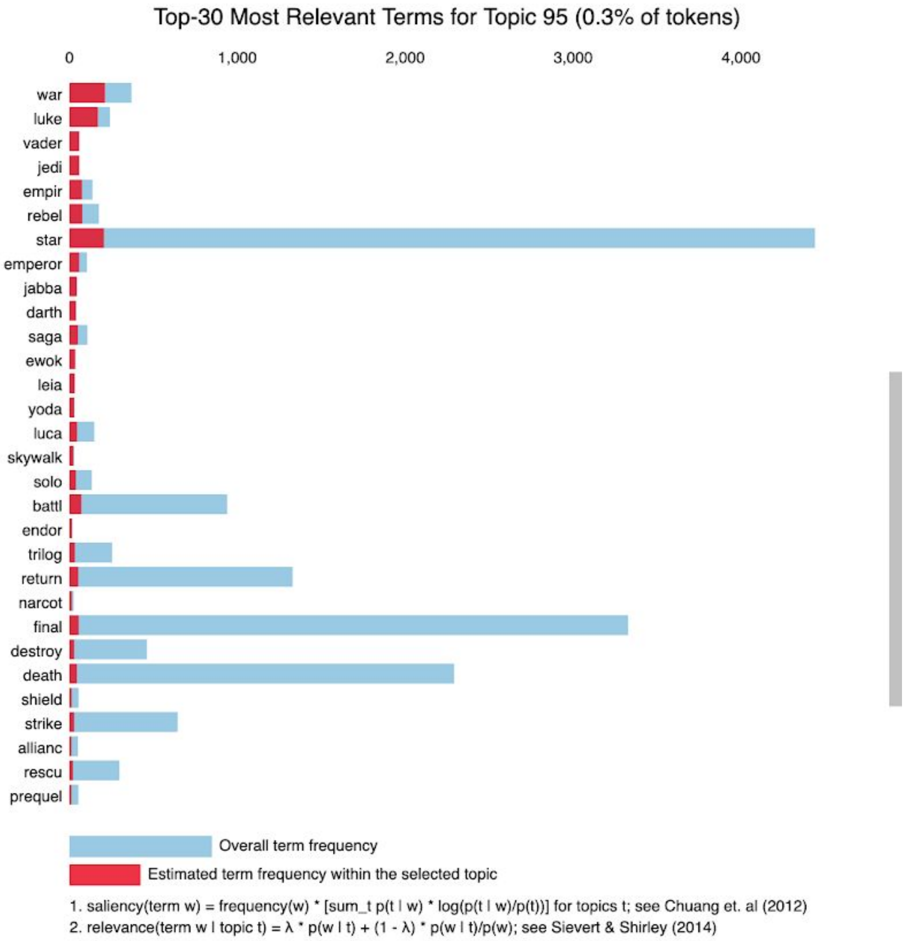
Here, the results are a little better, as the terms that can apply to most topics got filtered out (such as good, scene, character, etc). At the same time, some of the more unique terms are now visible, such as “jedi” and “vader”. Notice that the blue bars for those terms are not that large and the red bars encompass much of the blue bars. This indicates that these terms do not appear that often across the entire corpus, but are frequent for this topic. The LDA model has correctly identified these terms as highly relevant for this topic. However, there are still some terms that are not very relevant to *Star Wars*. This example was created by generating a total of 35 topics, and the next section will demonstrate how adjusting the number of topics can affect the quality of the results.

B. Amount of Topics

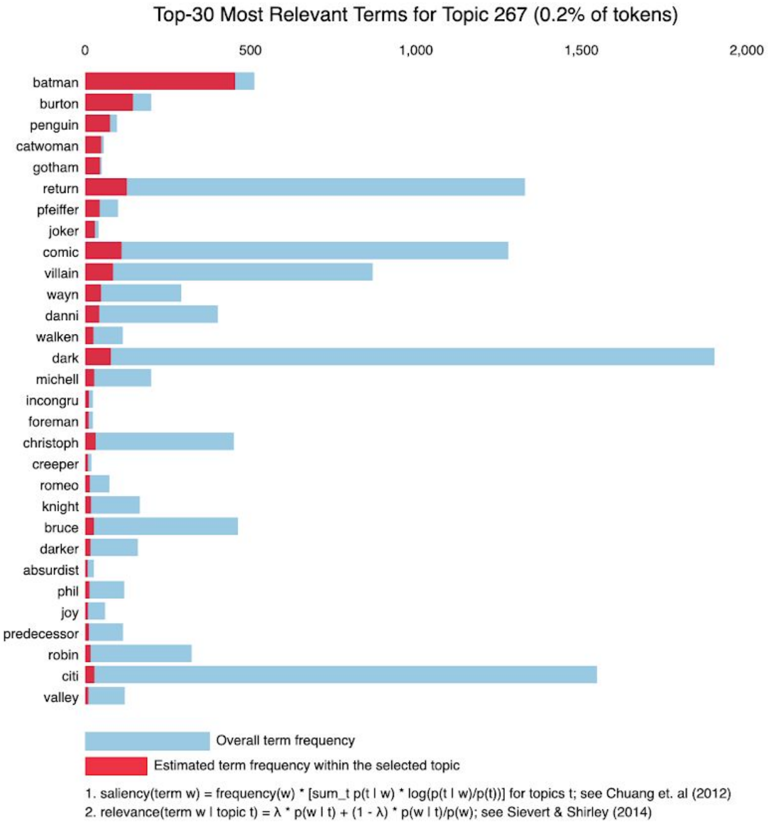
The number of topics we choose to extract affects the specificity of our topic outcomes. Choosing to generate a small amount of topics might skew the results because one can assume that 50,000 movie reviews should contain a much higher amount of topics. Choosing too many topics can also skew the results because then the model might begin to create topics that are too distinct, needlessly creating niche topics that will be harder to classify new data into.



The picture above shows the relevancy results for one topic, based on 25,000 positive movie reviews with 75 total number of topics generated. At a quick glance, one could deduce that this topic is related to *Star Wars* in some way, as the top terms include “luke”, “jedi”, “luca” (for George Lucas), and “vader”. However we also see terms that are not as related to *Star Wars*, such as “stalingrad”. “Stalingrad” may have been grouped into this topic as a famous WWII battle occurred there, and *Star Wars* obviously has a lot of battles. In this situation, more topics would be beneficial, as non-*Star Wars* terms will be grouped into other topics, and *less frequent, but more relevant* Star Wars terms will populate the topic.



The image above shows the results when generating 300 topics instead of the former 75. Notice that every single term here relates to *Star Wars* and the other non-*Star Wars* terms were grouped into other topics. Below is another example from the same output, but a different topic:



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

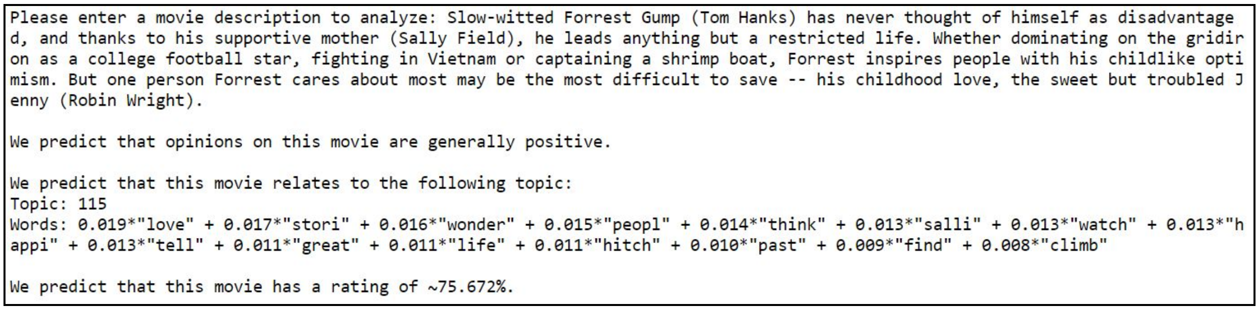
Every single word here relates to the *Batman* movies directed by Tim Burton from the early 1990s. This further supports the idea that generating 300 topics will provide accurate results. It is important that our topics are accurate because the expert system will need to classify new information into these topics before sentiment analysis.

VI. RESULTS

The following shows the output that our expert system returns when inputting different movie descriptions. We have included three examples each for positive and negative results.

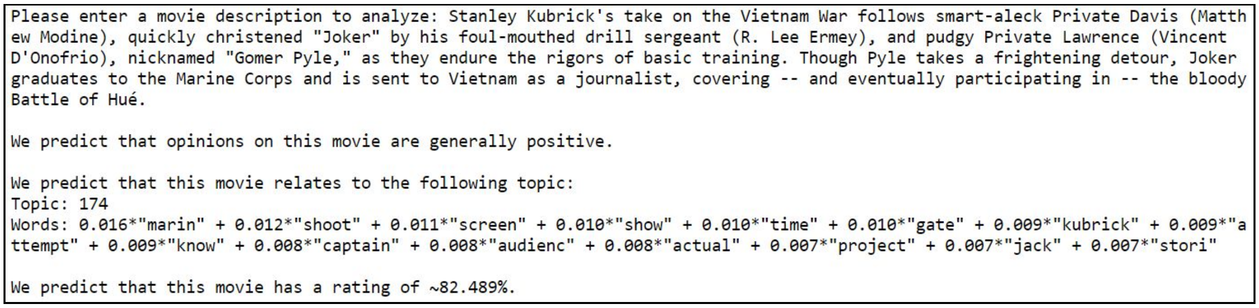
A. Examples Output - Positive Results

Using *Forrest Gump*'s movie description as input:



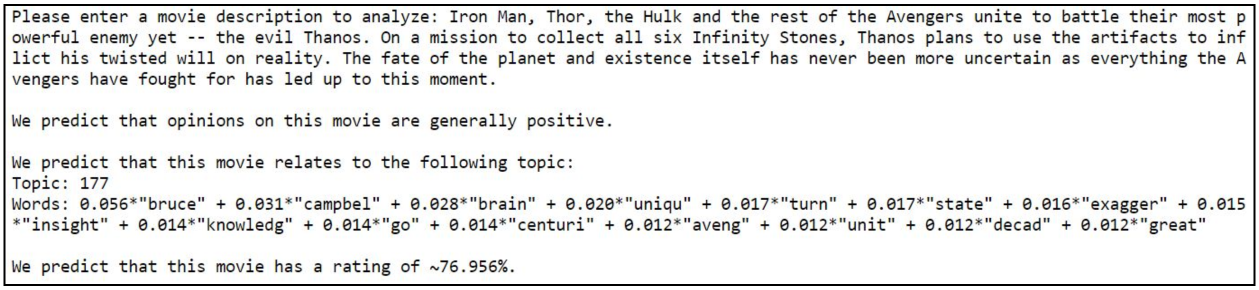
```
Please enter a movie description to analyze: Slow-witted Forrest Gump (Tom Hanks) has never thought of himself as disadvantage
d, and thanks to his supportive mother (Sally Field), he leads anything but a restricted life. Whether dominating on the gridir
on as a college football star, fighting in Vietnam or captaining a shrimp boat, Forrest inspires people with his childlike opti
mism. But one person Forrest cares about most may be the most difficult to save -- his childhood love, the sweet but troubled J
enny (Robin Wright).

We predict that opinions on this movie are generally positive.

We predict that this movie relates to the following topic:
Topic: 115
Words: 0.019*"love" + 0.017*"stori" + 0.016*"wonder" + 0.015*"peopl" + 0.014*"think" + 0.013*"salli" + 0.013*"watch" + 0.013*"h
appi" + 0.013*"tell" + 0.011*"great" + 0.011*"life" + 0.011*"hitch" + 0.010*"past" + 0.009*"find" + 0.008*"climb"

We predict that this movie has a rating of ~75.672%.
```

Using *Full Metal Jacket*'s movie description as input:



```
Please enter a movie description to analyze: Stanley Kubrick's take on the Vietnam War follows smart-aleck Private Davis (Matth
ew Modine), quickly christened "Joker" by his foul-mouthed drill sergeant (R. Lee Ermey), and pudgy Private Lawrence (Vincent
D'Onofrio), nicknamed "Gomer Pyle," as they endure the rigors of basic training. Though Pyle takes a frightening detour, Joker
graduates to the Marine Corps and is sent to Vietnam as a journalist, covering -- and eventually participating in -- the bloody
Battle of Hué.

We predict that opinions on this movie are generally positive.

We predict that this movie relates to the following topic:
Topic: 174
Words: 0.016*"marin" + 0.012*"shoot" + 0.011*"screen" + 0.010*"show" + 0.010*"time" + 0.010*"gate" + 0.009*"kubrick" + 0.009*"a
ttempt" + 0.009*"know" + 0.008*"captain" + 0.008*"audienc" + 0.008*"actual" + 0.007*"project" + 0.007*"jack" + 0.007*"stori"

We predict that this movie has a rating of ~82.489%.
```

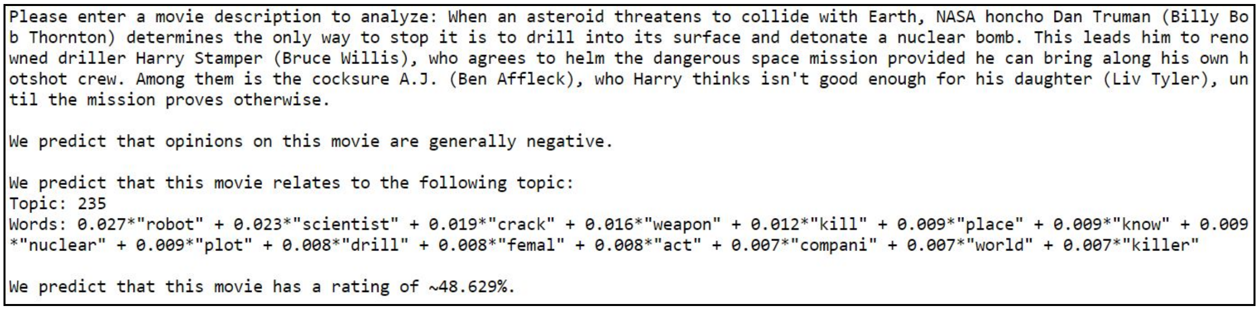
Using *Avengers: Infinity War*’s movie description as input:



```
Please enter a movie description to analyze: Iron Man, Thor, the Hulk and the rest of the Avengers unite to battle their most p
owerful enemy yet -- the evil Thanos. On a mission to collect all six Infinity Stones, Thanos plans to use the artifacts to inf
lict his twisted will on reality. The fate of the planet and existence itself has never been more uncertain as everything the A
vengers have fought for has led up to this moment.

We predict that opinions on this movie are generally positive.

We predict that this movie relates to the following topic:
Topic: 177
Words: 0.056*"bruce" + 0.031*"campbel" + 0.028*"brain" + 0.020*"uniqu" + 0.017*"turn" + 0.017*"state" + 0.016*"exagger" + 0.015
*"insight" + 0.014*"knowledg" + 0.014*"go" + 0.014*"centuri" + 0.012*"aveng" + 0.012*"unit" + 0.012*"decad" + 0.012*"great"

We predict that this movie has a rating of ~76.956%.
```

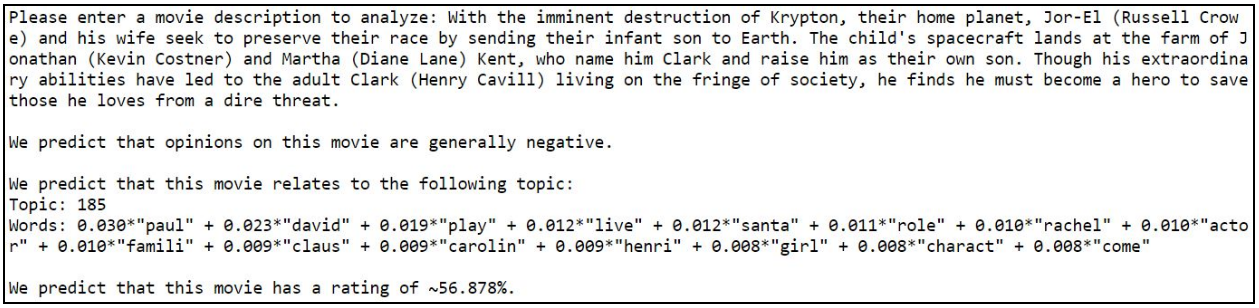
B. Example Output - Negative Results

Using *Armageddon*’s movie description as input:



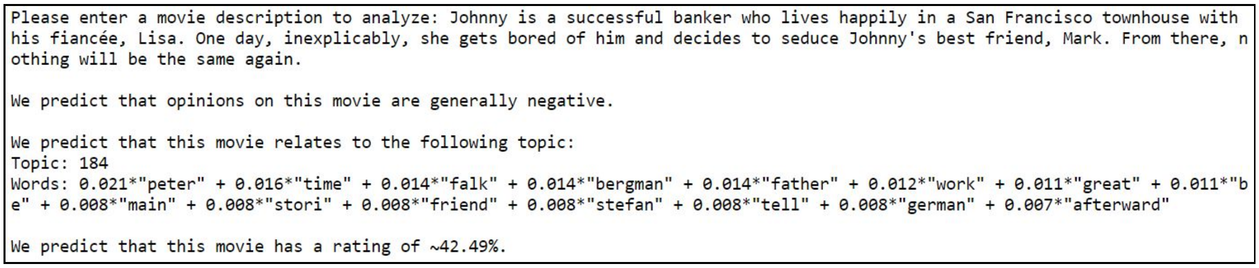
```
Please enter a movie description to analyze: When an asteroid threatens to collide with Earth, NASA honcho Dan Truman (Billy Bo
b Thornton) determines the only way to stop it is to drill into its surface and detonate a nuclear bomb. This leads him to reno
wned driller Harry Stamper (Bruce Willis), who agrees to helm the dangerous space mission provided he can bring along his own h
otshot crew. Among them is the cocksure A.J. (Ben Affleck), who Harry thinks isn't good enough for his daughter (Liv Tyler), un
til the mission proves otherwise.

We predict that opinions on this movie are generally negative.

We predict that this movie relates to the following topic:
Topic: 235
Words: 0.027*"robot" + 0.023*"scientist" + 0.019*"crack" + 0.016*"weapon" + 0.012*"kill" + 0.009*"place" + 0.009*"know" + 0.009
*"nuclear" + 0.009*"plot" + 0.008*"drill" + 0.008*"femal" + 0.008*"act" + 0.007*"compani" + 0.007*"world" + 0.007*"killer"

We predict that this movie has a rating of ~48.629%.
```

Using *Man of Steel*’s movie description as input:



```
Please enter a movie description to analyze: With the imminent destruction of Krypton, their home planet, Jor-El (Russell Crow
e) and his wife seek to preserve their race by sending their infant son to Earth. The child's spacecraft lands at the farm of J
onathan (Kevin Costner) and Martha (Diane Lane) Kent, who name him Clark and raise him as their own son. Though his extraordina
ry abilities have led to the adult Clark (Henry Cavill) living on the fringe of society, he finds he must become a hero to save
those he loves from a dire threat.

We predict that opinions on this movie are generally negative.

We predict that this movie relates to the following topic:
Topic: 185
Words: 0.030*"paul" + 0.023*"david" + 0.019*"play" + 0.012*"live" + 0.012*"santa" + 0.011*"role" + 0.010*"rachel" + 0.010*"acto
r" + 0.010*"famili" + 0.009*"claus" + 0.009*"carolin" + 0.009*"henri" + 0.008*"girl" + 0.008*"charact" + 0.008*"come"

We predict that this movie has a rating of ~56.878%.
```

Using *The Room*’s movie description as input:



```
Please enter a movie description to analyze: Johnny is a successful banker who lives happily in a San Francisco townhouse with
his fiancée, Lisa. One day, inexplicably, she gets bored of him and decides to seduce Johnny's best friend, Mark. From there, n
othing will be the same again.

We predict that opinions on this movie are generally negative.

We predict that this movie relates to the following topic:
Topic: 184
Words: 0.021*"peter" + 0.016*"time" + 0.014*"falk" + 0.014*"bergman" + 0.014*"father" + 0.012*"work" + 0.011*"great" + 0.011*"b
e" + 0.008*"main" + 0.008*"stori" + 0.008*"friend" + 0.008*"stefan" + 0.008*"tell" + 0.008*"german" + 0.007*"afterward"

We predict that this movie has a rating of ~42.49%.
```

D. Discrepancies

Popular topics such as sports and superheroes can lead to our expert system assuming that the topic is very polarizing. This is because there might be many reviews pertaining to Batman in both the positive and negative categories, which makes sense because there are many Batman movies that vary greatly in rating. Our program tries to curb this discrepancy by looking at the top three classified topics instead of just one. This way, even if there is some polarity within the top results, the latter two can help realign the calculations.

Another natural discrepancy is with the fact that our corpus comes exclusively from IMDB reviews, which has a primarily Western audience. That gives our expert system a bias meaning our results are only applicable to a Western demographic.

VII. FUTURE WORK

One way to improve our results is to alter the function that calculates the rating as a percentage. Since this calculation is formulated from the potential topics a movie description fits into, there are a lot of results that the function could factor in. Anybody who wishes to change this function could easily access all 300 topics generated from the positive and negative categories and see the likelihood of how any given movie description fits into each. However, it is unlikely that using that many more topics would wield accurate results.

Another consideration is with our LDA model generation. We tested many different amounts of topics and concluded that extracting 300 topics generates topics that make sense to our natural language. With further extensive testing, there might be a more precise number of topics that leads to more accurate results.

VIII. CONCLUSION

In conclusion, it is possible to create an expert system that predicts movie ratings with reasonable results. While it is difficult to get precise ratings, the general sentiment that our expert system returns is sound. Using topic modeling to broadly assess the sentiment of a population works as long as the assessment uses corpus specific to what is being assessed. Therefore, this method works best when the assessment has a clear, distinct demographic it is making the predictions for.

REFERENCES

[1] Baccianella, S., Esuli, A., & Sebastiani, F. (2010, May). Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In Lrec (Vol. 10, No. 2010, pp. 2200-2204).

[2] Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.

[3] Rehurek, Radim and Petr Sojka (2010), Software Framework for Topic Modelling with Large Corpora. ELRA. (Page 45-50).

[4] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[5] M. S. Incorporated, "Data Mining Explained," MicroStrategy. [Online]. Available: https://www.microstrategy.com/us/resources/introductory-guides/data-mining-explained.

[6] T. Doll, "LDA Topic Modeling," Medium, 11-Mar-2019. [Online]. Available: https://towardsdatascience.com/lda-topic-modeling-an-explanation-e184c90aadcd.

[7] V. Sasikumar, "Natural Language Processing - Topic modelling (including latent Dirichlet allocation-LDA &...," Medium, 01-Mar-2019. [Online]. Available: https://medium.com/@vivekvscool/natural-language-processing-topic-modelling-including-latent-dirichlet-allocation-lda-860e5a3d377f.

[8] "Topic Analysis," MonkeyLearn, 27-Jan-2020. [Online]. Available: https://monkeylearn.com/topic-analysis/.