Commoditizing Creativity:
Using Screenplays to Predict Film Production Budget

Amir Malek (am6370)

Brian Pennisi (bp2221)

Johnny Ma (jlm10003)

Ted Xie (tx607)

All code and data can be found at: https://github.com/JohnnyMa314/CommoditizingCreativity

I. Business Understanding

Film and television production companies have always followed a fixed sequence for content creation and business development. Broadly explained, this content creation lifecycle consists of a creative pitch and business negotiations, followed by pre-production, production, and post-production. Once these stages are complete, a film or show is ready for launch. Due to industry-wide interest in accelerating this content creation lifecycle, we decided to take a closer look at the business negotiations (more specifically, film financing), where a screenplay submitted for review is approved and financed for production. Studios receive thousands of solicited and unsolicited screenplays each day, and must hire and train professional script readers to perform "script coverage," where scripts receive feedback as to their quality and expected cost of production. As the volume of received scripts is simply too high for the script readers to cover, most are filtered from consideration without even being read. If we can develop a data science based pipeline to instantly estimate a script's expected production cost, we can prevent script readers from wasting time reading scripts the studios cannot or will not financially support.

We can draw a comparison to the issue of resume scanning for hiring managers and recruiters. Faced with a large number of incoming resumes for only a few available slots, data scientists leverage available data to create applicant tracking systems (ATS), which helps organize job applications and even filter them, forwarding only the most qualified candidates to hiring managers and recruiters. Both of these examples implement machine learning techniques to encode aspects of the text corpus as features, and determine the weight of those features based on the human inputs of the target variable. For example, the resume dataset for the ATS must have had a large training set of resumes where the target variable (i.e. the resume grade or rating) was determined by resume readers. The model could then use those feature weights to evaluate a test set to determine how well it can generalize on new data.

We can apply a similar approach to the issue of film financing by creating a model which takes movie scripts as an input and provides an estimated budget category, either low or high budget as an output. From a business perspective, this would alleviate the creative bottleneck created by limited script readers, and allow for faster filtering of scripts from a large volume to meet budget requirements. For a small production lab such as NYU's, where we estimate each reader spends around 6 hours per week reading scripts, the ability to automatically screen high budget scripts could allow them to identify lower budget scripts that the production company can actually produce.

II. Data Understanding

Based on the business problem, our data requirements are movie scripts as well as their associated budgets. Using a web crawler on IMSDB.com, an internet movie script database commonly used by industry and academics alike, we acquired raw .txt files and associated metadata for 1084 screenplays. The scripts contain scene descriptions, stage direction, and character dialogue, with an average of 5,000 lines of text and 24,000 tokens after preprocessing per screenplay. See Appendix A-1 for an example script. The observed budget for each film was collected from boxofficemojo.com and the-numbers.com, the industry standard box office data websites for tracking movie financials. These budget values were obtained using the associated IMDB-ID (a unique identifier for each movie), as well as a name + year search using fuzzy string matching, with low probability matches manually fixed for accuracy. When movie financial sources disagree about a movie's budget, we record the lower amount, as Hollywood accounting firms have an incentive to inflate the budget ex-post to reduce taxation on profits.

One issue that we foresaw with the dataset was the inconsistencies in the budget due to inflation, as our dataset spans about 90 years of movie production. As part of our data preparation we standardized the budget values using inflation calculations with a consumer price index (CPI) from the U.S. Bureau of Labor Statistics. A few of the movies in our dataset were produced by international companies with budgets reported in foreign currency. We changed these budgets into USD using Oanda.com's currency converter. We end up with 1,084 budgets standardized to current day USD.

Given we are working with text data, understanding the problem we are trying to solve is important in order to decide which language models to apply. Each record is a movie script that is 20-200 pages long. In order to predict budget effectively, we need our models to do a good job of working with a large and complex corpus. As such, performance guides many of our decisions while semantic context is captured by leveraging language models like those from spaCy and Word2Vec.

III. Data Preparation and Exploratory Analysis

Target Variable

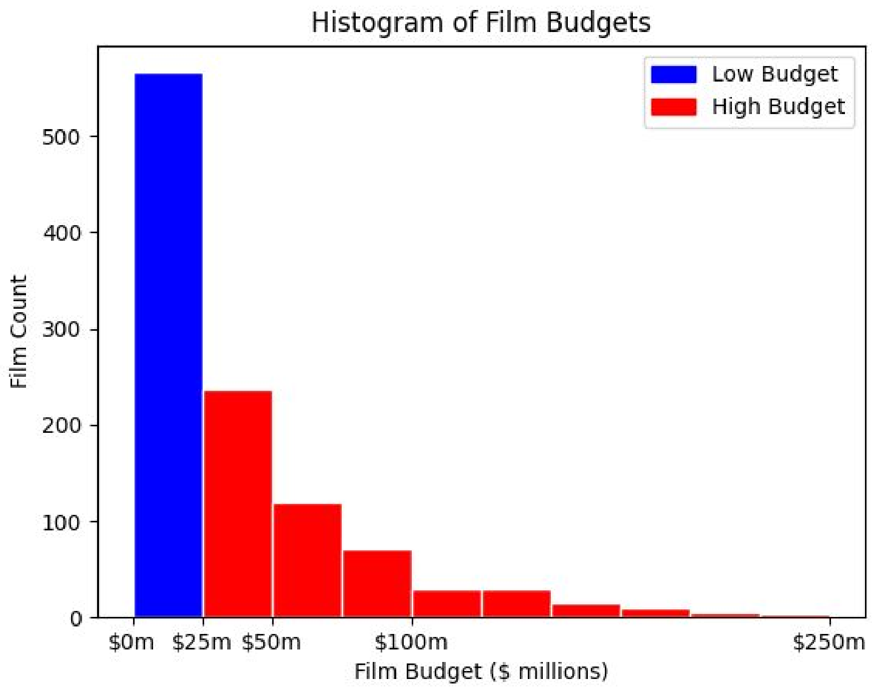


Figure 1: Histogram of Film Budgets

Figure 1 shows the distribution of the (inflation adjusted) film budgets in our dataset. As expected, there are exponentially fewer movies as the budget linearly increases. Though the budget prediction problem would be best formulated as a regression problem, as our data is limited to 1,084 scripts we opt for the more simplified and interpretable approach of budget classification. We created two equal sized categories for film budget, each containing 542 films. A **low budget** film was defined as costing equal to or less than $25,000,000, and a **high budget** as above $25,000,000. While a $250,000,000 movie is certainly different from a $25,000,000 movie, and $20,000,000 different from $100,000, these categories match up roughly to the categorization standards of production companies.

Feature Engineering

Working with text presents a unique step compared to structured data since one needs to convert the words in a document to structured data before priming. Featurizing large corpuses (ie. 20-200 pages) of text that make up a movie script is an additional challenge compared to phrases or shorter passages such as Tweets or movie reviews due to performance concerns from long range dependencies (ie. the first scene relates to a later scene, which is 10,000 words later) and high variation. As such, we need our vectorizers, which turn the scripts into structured data to be trained by our classifiers, to be efficient in time and memory as well as have an effective way to aggregate all of the words in the document such that it accurately represents the entire passage.

We start by looking at different models of script representation. Ultimately, we chose normalized word counts via TF-IDF, Named Entity Recognition (NER) summaries built on Spacy's off-the-shelf English language model, and an averaged word embeddings model using Word2Vec.

Term Frequency-Inverse Document Frequency, or TF-IDF is a normalized word count model that represents a corpus as a collection of word counts, normalized by inter-document frequency. TF-IDF is often the best place to start because it can produce low cost and relatively intuitive statistics on the word counts. These are then combined into a k dimensional vector based on the number of words that fit certain criteria, which we discuss further in Section IV.

Spacy is a powerful yet lightweight option for language models. Spacy's off-the-shelf English language models use pre-trained word embeddings which provide relatively reliable accuracy which border SOA (as of 2017.) Paying attention to performance, we used the lightweight English model 'en_core_web_sm'. It uses a deep neural network architecture which takes into account a window of preceding words, word shape, and other features in order to understand the context in which the word is used. Context is important because it allows us to know that Nike is a company when appearing the sentence "I work at Nike" and a greek goddess in "The goddess of victory is Nike." Armed with this language model, we leveraged the Named Entity Recognition attribute which recognizes and counts instances of 18 different types of entities across our corpus including names, companies, locations, and currency. The preview of the language model output and summary of the entity types can be found in Appendix A-2.

Similar to spaCy, Word2Vec uses word windows in order to form numerical representations of words. The words get assigned values based on the context in which they occur. As such, words that appear in a similar context to one another are close in a vector space where cosine similarity is the default to measure closeness. Further, they have a useful property where we can add these vectors to produce similar words. For instance, "king" - "male" + "female" = "queen". This is important because it allows us to represent our corpus using a normalized sum of each of the word embeddings in the document. We then apply our Word2Vec language model to each word and average it in order to represent the script.

IV. Modeling and Evaluation

Language Models

As mentioned in Section III, the hyperparameter tuning of our language models is an important consideration since they determine how we create our structure training data. In our case, we trained models for both TF-IDF and Word2Vec. It should be noted that we chose not to look for the best model empirically due to memory limitations.

Important hyperparameters to consider in TF-IDF include the minimum number of times a word should appear in total or across all documents. Given the size of our movie scripts, it is important to find the right trade off between performance and number of words. One can also reduce the bias by adding more detailed features using ngram_range, which allows one to specify the maximum and minimum number of words that make up a key in your language model. “ngram_range” by default only takes into consideration a single word. However at the risk of adding significantly more features that occur less often than single words, you can count phrases to pick up features like “not good.” The optimal parameters were “ngram_range = (1,2)”, which allows for unigram and bigrams as well as “min_df = 0.2” which removed all fields that occured in less than 1 in every 5 documents. Overall, our language model serves as a nice trade off on the bias-variance spectrum where the ngram_range including bigrams optimizes for bias while the min_df addresses the variance.

Similar to TF-IDF, one can establish a bound on the number of times a word appears in a document or across the corpus. This is especially important when one is trying to optimize for both time and space complexity. Our final model was 233 MB. Since our storage cost is linear, it is important to find the right number of words. Across 1,084 movie scripts, using “min_count = 2” gets 60.6% of all of the total words in the text.

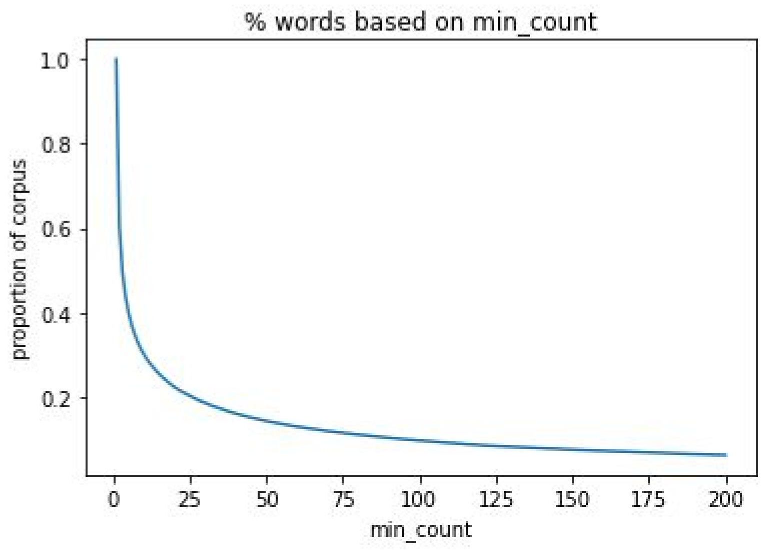


Figure 2: EDA of unique words across our corpus of scripts

The number of dimensions "size" represent the number of features that the model can use to uniquely draw your word in a vector hyperplane, the higher this is, the lower the bias but higher the variance. Also larger dimensions linearly take up more space in memory.

One of the most important hyperparameters in Word2vec is which algorithm to use to train the word embeddings: CBOW and Skip-Gram. CBOW uses the traditional approach of trying to predict a word given the trailing and leading words. Skip-Gram does quite the opposite in that it tries to predict the leading and trailing words (or context) based on a single word or phrase. The latter works well with a small amount of the training data and represents well even rare words or phrases. This is useful for movie scripts which may include rare words to effect. Unfortunately, given that we are constrained by both time and space complexity, we went with CBOW. The optimal hyperparameters for our Word2Vec language model were "min_count = 2", "sg = 0" (for CBOW), "size = 300", and "window = 5."

Classifiers

The baseline model for binary classification would be an uniform 50/50 guess on low vs. high budget, as the two categories have an equal number of films. To improve on this baseline, we will leverage the aforementioned textual features of the scripts and feed them into a suite of classification algorithms. We use Logistic Regression with L2 regularization, Gradient Boosting Machine, and Support Vector Machine for budget classification. We chose these algorithms as they can theoretically approximate the conditional probability distribution used in binary classification, at varying degrees of complexity given finite data. For our TF-IDF model, we also use two Naive Bayes algorithms: Multinomial and Bernoulli Bayes.

We split our data into 80-20 training and test sets, as is the standard in machine learning. We run 5-fold cross validation and grid search to select hyperparameters, evaluated on the held-out test set using F1-Score as our metric of choice. We chose F1 over both classes as there is no significant difference between the recall and precision of the classes for the best performing algorithms. If production companies are looking to focus on identifying low or high budget movies, we can easily optimize for precision on either budget category. If the producers are worried about losing out on good scripts in a specified budget category, we can optimize for recall instead.

The results of each of the models are shown below in Table 1.

Classification Algorithms	TF-IDF	Named Entity Recognition	Word2Vec
Logistic Regression (C = 0.01)	**0.712**	0.546	**0.627**
Gradient Boosting Machine (L = 0.05)	0.647	**0.571**	0.611
Support Vector Machine (C = 10)	0.703	0.559	0.610
Multinomial Bayes (alpha = 1)	0.637	N/A	N/A
Bernoulli Bayes (alpha = 5)	0.643	N/A	N/A

Table 1: F-1 Score by Classifier

The TF-IDF features perform best with an F1-Score of 0.712 using a Logistic Regression classifier. The strongest n-grams predictive of a high budget include “night,” “fade exterior,” “wipe,” “car drives,” and "snow," which are all indicative of expensive filming conditions or a large number of scenes. The Logistic Regression classifier performs best in two out of the three feature sets, with the CBOW Word2Vec embeddings performing better than NER using every model. Overall, TF-IDF is the best performing feature set across the board. This is likely due to the ability of the TF-IDF to capture distinctive and important features of long form text documents, compared to NER which captures a coarser set of features and the flattened effect of averaging word embeddings over long passages. While it is surprising to see the lower performance of Gradient Boost Machine, it's important to note that this algorithm is heavily dependent on "n_estimators". Unfortunately due to processing constraints, we would need access to more computing power in order to produce a better fitting model.

As we are interested in improving the performance of our pipeline, we use the best prediction algorithm, Logistic Regression, on different feature combinations. We report the results of each combination below in Table 2. As precision and recall for both budget classes are within ±0.02 of the averaged precision and recall, class specific measurements are omitted from the results.

Features	**Precision**	**Recall**	**Accuracy**	**F1-Score**
TF-IDF	0.71	0.71	0.711	0.712
TF-IDF + NER	**0.72**	**0.72**	**0.723**	**0.724**
TF-IDF + W2V	0.68	0.69	0.684	0.684
NER + W2V	0.64	0.64	0.636	0.636
TF-IDF + NER + W2V	0.69	0.69	0.690	0.690

Table 2: Performance Breakdown of Logistic Regression (C = 0.01) by Feature Set

The best performing combination of features TF-IDF and NER, with a slightly higher F1-Score of 0.724. The ROC curve of this best model is shown in Figure 3. The NER features likely add some context to the

TF-IDF model, though the ability of the TF-IDF to capture salient features leads to only a slight improvement. Adding the Word2Vec model decreases performance, likely due to collinearity with the word counts from the TF-IDF model.

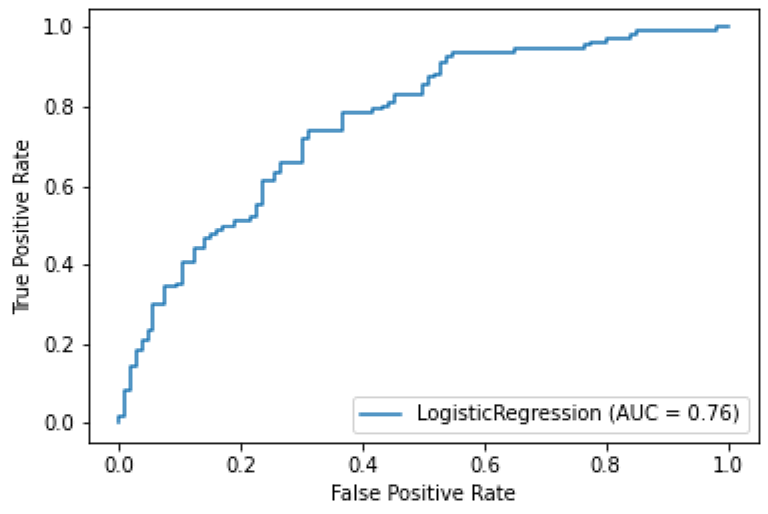


Figure 3: ROC Curve for Logistic Regression Model

V. Deployment

The combined feature model improves on the baseline significantly, with a 22% improvement versus random guessing. As the Logistic Regression model can return probabilities of class membership, a production company who is interested in finding high budget films can quickly eliminate scripts that have a small probability of being high budget. This model pipeline can also be useful for indie production companies who want to identify small budget scripts to finance. After running a model on a stack of digital scripts, a studio executive can then pass relevant scripts to the script readers, who would subsequently not waste their labor time reading scripts the studio would not have the funds to produce.

In practice, the model should be deployed for budget estimation. There are two main uses of predicted budgets. The budget classification pipeline can be utilized as a filter to find movie scripts that fit a budget constraint and desired genre to move forward into the production process. However, filtering can lead to opportunity costs when a script with high potential of success is discarded, so we will have to focus more on recall for filters in order to not incur a cost for missing a script An accurate budget prediction can also provide a guideline to keep production costs on track. Films typically have a pre-budget in the planning phase which investors can expect to match the true production budget. In the case where a script is already selected for production and a reliable budget estimate is needed, the model should be tuned to focus on precision. Ultimately, it would be useful to offer the probability estimates, ROC curve, and confusion matrix so the user can decide the tradeoff based on the expected values they assign to each outcome.

An API can be developed to help production or distribution companies access the current top performing model. This may be updated as new models are trained offline to incorporate new features, or where new learning methods are built. For more advanced users, it may be useful to allow the user to configure the featurization pipeline, such as by providing their own language model. The best model should be re-evaluated periodically to avoid concept drift. New scripts can be collected through web scrapers and there should be microservices that facilitate the updating of these models based on the new data. The influencers of movie budgets will shift as new technology, such as CGI, or popular genres change. The budgets throughout time will also be affected by inflation and should also be adjusted for. The model can be augmented to include more metadata features such as the screenplay author, author social network, or self reported genre. Being able to encode and condition on more information available at time of script coverage can lead to higher class prediction accuracy. The prediction model sets up a framework that allows analysis of other features and target variables.

The simplest way to gauge the market performance of the model is to compare predicted budgets with true budgets of scripts produced through the algorithm. The challenge is that the final budget is not known until the movie has presumably been released, so there could be a material lag of a year or more before we get this data.

VI. Considerations

The budget predictions are based on previously designated budgets, meaning the model is learning previous human made assignments. The model will lack data on unobserved scripts and their features. For instance if a training set contained only horror and comedy films, the model will not be able to accurately assign the budget when seeing a science fiction script feature such as "outer space." In addition, the dataset is affected by selection bias; only screenplays that are produced have their budgets recorded. As only 0.3% of received screenplays are greenlit, the vast majority of the scripts that come across a production studio's desk are not observed. While there is a 50-50 uniform split between our budget classes for produced scripts, there may be many more high budget scripts submitted than low budget, which would affect our prior on the distribution of budget classes for a given pile of scripts. As even scripts that are not produced are assigned budget categories during script coverage, partnerships with studios or script reading services could give us access to more representative screenplays and budgets.

Privacy and copyright concerns prevent certain studios and writers from releasing their scripts to the public, and script readers may be reluctant to send in scripts they know will be fed into the system. Encrypting script text upon submission, removing scripts from memory after prediction, and public model transparency releases could help relieve these concerns. Returning a list of top features indicative of budget could incentivize writers who want to aim for a specific budget category to use the service. However, if the budget classification model or top features are known to the public, writers could target a specific budget classification. The budget classification process can then be influenced by "white words" where key high value vocabulary, perhaps exploiting spurious correlations, are included in the corpus.

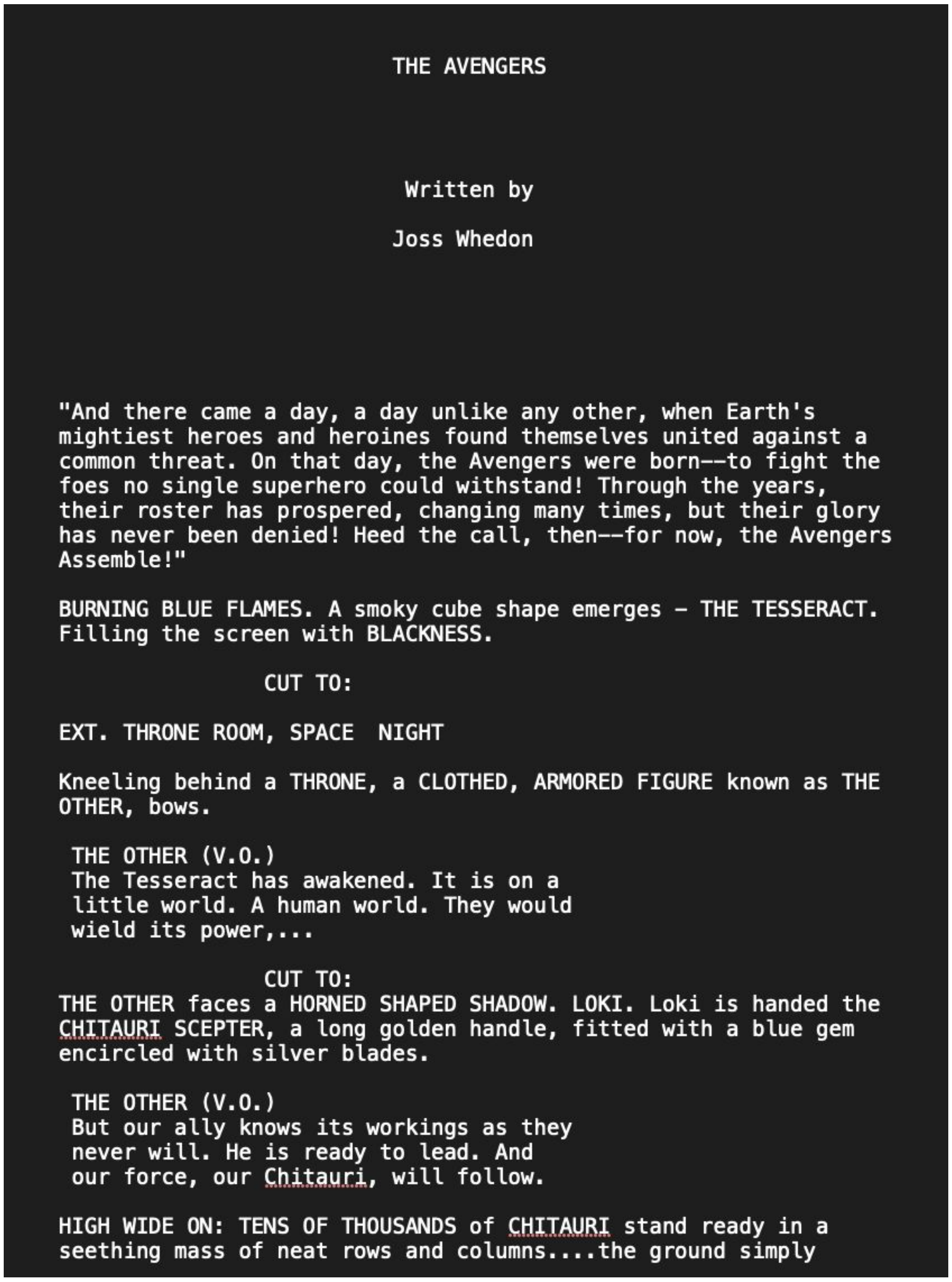
VII. Conclusion

Budget is one of the most important screening criteria for film scripts, and is often directly estimated during script coverage. The ability to automatically and near instantly parse stacks of screenplays and classify their budget categories can save an enormous amount of time for the script readers who currently spend hours sifting through potentially unfinanceable scripts.

Our service allows the conversion of text to features and classification in a matter of moments. The presented models tackle budget predictions through different classification algorithms such as Logistic Regression, Gradient Boosting Classifier, Support Vector Classification, and Naive Bayes. All our models performed better than random guessing. With F1 score as the optimized metric, the TF-IDF model with a Logistic Regression classification algorithm performed best. We were able to slightly improve model performance with a combination approach of features from TF-IDF and NER, giving us a final F1 score of 0.724.

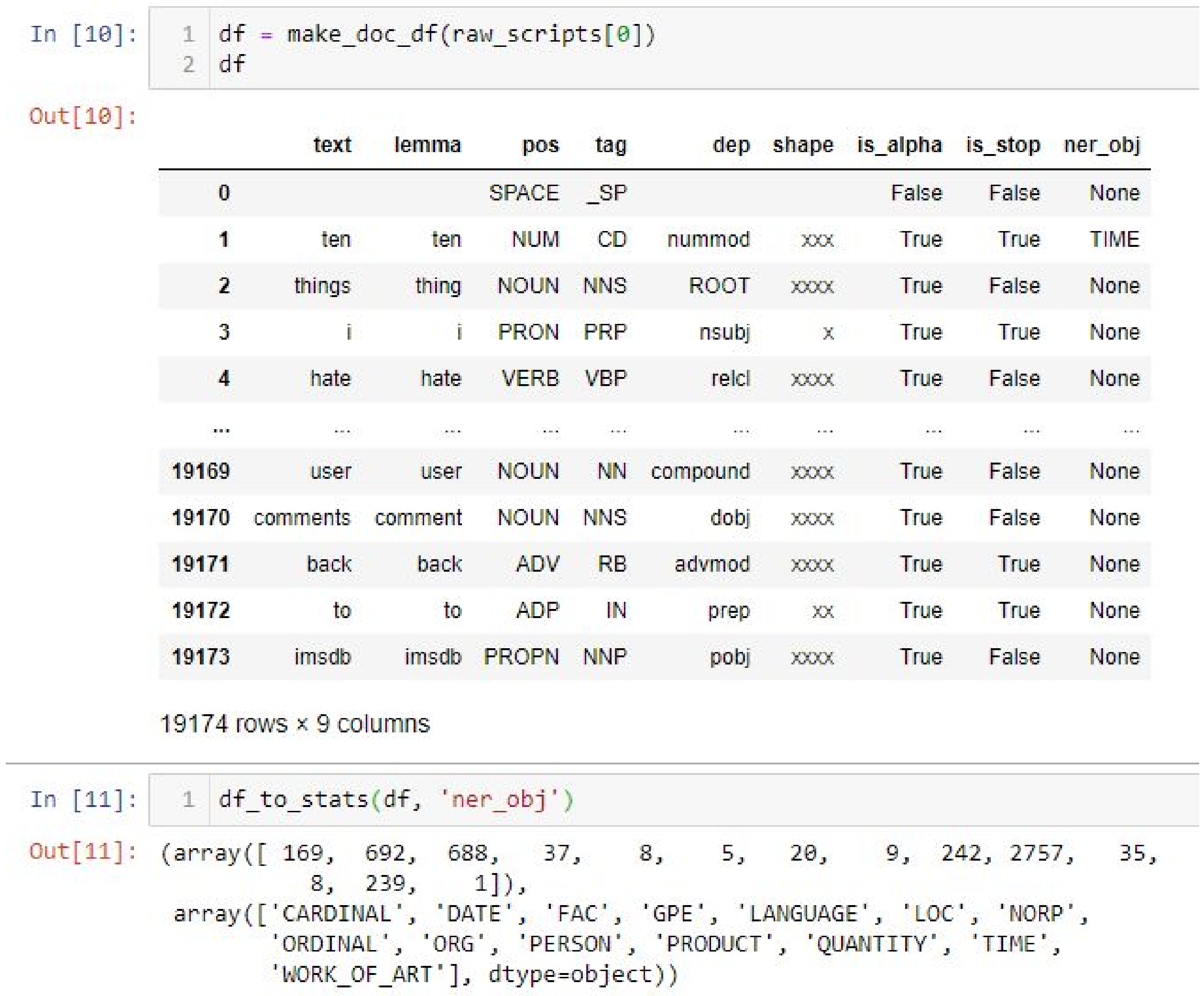
The purpose of the algorithm is to increase the efficiency of the script coverage process by ensuring more relevant scripts are read and unfeasible scripts are filtered out. The pipeline is not meant to pass absolute judgement on the potential of any given script. We developed this pipeline to provide insight into the process of estimating a film’s budget, and leave automatic script coverage or assessment of script quality for future work.

A. Appendix



```
                              THE AVENGERS



                               Written by

                              Joss Whedon





"And there came a day, a day unlike any other, when Earth's
mightiest heroes and heroines found themselves united against a
common threat. On that day, the Avengers were born--to fight the
foes no single superhero could withstand! Through the years,
their roster has prospered, changing many times, but their glory
has never been denied! Heed the call, then--for now, the Avengers
Assemble!"

BURNING BLUE FLAMES. A smoky cube shape emerges - THE TESSERACT.
Filling the screen with BLACKNESS.

                CUT TO:

EXT. THRONE ROOM, SPACE  NIGHT

Kneeling behind a THRONE, a CLOTHED, ARMORED FIGURE known as THE
OTHER, bows.

 THE OTHER (V.O.)
 The Tesseract has awakened. It is on a
 little world. A human world. They would
 wield its power,...

                CUT TO:
THE OTHER faces a HORNED SHAPED SHADOW. LOKI. Loki is handed the
CHITAURI SCEPTER, a long golden handle, fitted with a blue gem
encircled with silver blades.

 THE OTHER (V.O.)
 But our ally knows its workings as they
 never will. He is ready to lead. And
 our force, our Chitauri, will follow.

HIGH WIDE ON: TENS OF THOUSANDS of CHITAURI stand ready in a
seething mass of neat rows and columns....the ground simply
```

A-1: The first page of the raw .txt files for The Avengers (2012) screenplay.



```
In [10]: df = make_doc_df(raw_scripts[0])
         df
```

Out[10]:

	text	lemma	pos	tag	dep	shape	is_alpha	is_stop	ner_obj
0			SPACE	_SP			False	False	None
1	ten	ten	NUM	CD	nummod	xxx	True	True	TIME
2	things	thing	NOUN	NNS	ROOT	xxxx	True	False	None
3	i	i	PRON	PRP	nsubj	x	True	True	None
4	hate	hate	VERB	VBP	relcl	xxxx	True	False	None
...	...	...	...	...	...	...	...	...	...
19169	user	user	NOUN	NN	compound	xxxx	True	False	None
19170	comments	comment	NOUN	NNS	dobj	xxxx	True	False	None
19171	back	back	ADV	RB	advmod	xxxx	True	True	None
19172	to	to	ADP	IN	prep	xx	True	True	None
19173	imsdb	imsdb	PROPN	NNP	pobj	xxxx	True	False	None

19174 rows × 9 columns

```
In [11]: df_to_stats(df, 'ner_obj')

Out[11]: (array([ 169,  692,  688,   37,    8,    5,   20,    9,  242, 2757,   35,
                    8,  239,    1]),
          array(['CARDINAL', 'DATE', 'FAC', 'GPE', 'LANGUAGE', 'LOC', 'NORP',
                 'ORDINAL', 'ORG', 'PERSON', 'PRODUCT', 'QUANTITY', 'TIME',
                 'WORK_OF_ART'], dtype=object))
```

A-2: Demonstration of spaCy's language model on our first movie script (Out[10]), NER summary features (Out[11]: top array)), and 18 entity types extracted from the corpus (Out[11]: bottom array)

B. Bibliography

Agarwal, Apoorv, et al. “Parsing Screenplays for Extracting Social Networks from Movies.” *The Association for Computational Linguistics*, Dept. of Computer ScienceColumbia University, 2014, www.aclweb.org/anthology/W14-0907.pdf.

Chiu, Ming-Chang, et al. “Screenplay Quality Assessment: Can We Predict Who Gets Nominated?” *ArXiv*, Dept. of Computer ScienceUniversity of Southern California, 2020, arxiv.org/pdf/2005.06123.pdf.

Explosion.ai. *SPACY'S ENTITY RECOGNITION MODEL: Incremental Parsing with Bloom Embeddings & Residual CNNs*. *Youtube*, 12 Nov. 2017, www.youtube.com/watch?v=sqDHBH9IjRU.

Follows, Stephen, et al. “Judging Screenplays by Their Coverage.” *Stephen Follows*, Stephen Follows, Erroneous Wit, Somerset House, 2019, stephenfollows.com/wp-content/uploads/2019/01/JudgingScreenplaysByTheirCoverage_StephenFollows_c.pdf.

Kulshrestha, Ria. “NLP 101: Word2Vec - Skip-Gram and CBOW.” *Medium*, Towards Data Science, 26 Oct. 2020, towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314.

Kumar, Ritwik, et al. “Data Science and the Art of Producing Entertainment at Netflix.” *The Netflix Tech Blog*, Netflix TechBlog, 27 Mar. 2018, netflixtechblog.com/studio-production-data-science-646ee2cc21a1.

Lash, M. T., & Zhao, K. (2016, January 29). Early Predictions of Movie Success: The Who, What, and When of Profitability. Retrieved 2020, from https://arxiv.org/pdf/1506.05382v2.pdf

C. Contributions

Amir Malek: Literature Review, Data Preparation, Write-up

Brian Pennisi: Feature Engineering, Language Models, Write-up

Johnny Ma: Data Preparation, Language Models, Classification Models, Write-up

Ted Xie: Literature Review, Data Preparation, Write-up