CS229 Machine Learning Final Report
Paper Citation Classification
Category: Natural Language
Chung Bui (csb41), Camilo Saavedra (csaavedr), Sophia Liu (sophliu)
December, 2019

Introduction

Writing an academic paper can be a daunting task. Deciding on features including the topic, title, abstract, and journal, can be lengthy processes and all influence the impact of the research in the future. We would like to be able to predict the number of citations a research paper will receive, given various paper features. Given the large number of new papers published each year, such a tool would be useful for scientists to determine possible fields of exploration and choose meaningful paper features to maximize the impact of their research.

In this work, we classified academic paper titles into different bins based on citation number. The input to the algorithm is a set of features extracted from the paper title, journal title, and year of publication. The models explored include Naive Bayes, Random Forest, and Neural Network. These models are used to output a predicted classification for a given paper, which specifies a range of citation counts that the paper could have.

Related Works

There were several similar works in this field. Ibanez et al. classifies papers with few, some, or many citations. Their application of predicting on a time horizon was particularly novel, and their implementations of naive Bayes and logistic regression performed the best.

Yan et al. also used citation count as a measure of research popularity. Many input features were included, such as author experience and productivity, paper venue, and topic importance. It is likely that these factors, particularly the relative importance of the author, may be more correlated with citation count than other paper features. They were able to use simpler models, such as linear regression, kNN, and SVR. Other works also highlight the importance of choosing the appropriate paper features, such as paper age and the author importance, suggesting that many other relevant input features may be needed for accurate prediction. On the other hand, Baba et al. claim that the impact of research papers may be predicted using only paper content, such as the abstract.

Several studies looked into related metrics of paper performance. For instance, a recent study published earlier this year used more advanced deep learning techniques to predict the long-term performance of a recently published paper based on the citation in initial years. Despite the difference in input features, we found this implementation to be clever, and we found that a neural network implementation also yielded the best results in our system.

Dataset and Features

Initially, our first approach was to use third-party APIs to scrape existing databases for academic paper information like citation count, title, publication year, etc. However, these APIs often limited the number of queries per user per day, preventing the collection of the data at a scale which would be sufficient to train models. To remedy this issue, previous related works were searched for existing structured data. The dataset used was from a previous study on the effect of title length on citation counts [1]. In total this dataset contains ~140,000 records of general academic papers from 2007-2013. Each record includes the paper's title, year of publication, journal title, number of citations.

The dataset was split using a 85/15 train/test. The primary feature was the title, but secondary features, like the year of publication and the title lengths were also included as inputs to all models. In order to ensure all features were of similar magnitude, the numerical features were all normalized by subtracting the mean and dividing by the standard deviation of the training set.

The most important preprocessing step was how to parse the journal title and paper title text. Since the journals are discrete, a categorical numeric variable was created which encoded which journal the paper was from. The paper title text could be processed in several ways. The first pass was to create a bag of words model, where each title was transformed to a sparse vector containing words that it learned from the vocabulary. Another option was to use a pretrained embedding, like Google's universal sentence encoder. These would transform the titles into feature vectors which would then act as the inputs to the model.

Methods

3 different embeddings were explored, but they all lead to similar model performances. The embeddings either converted the string to a 50, 128, 256, or 500 feature vector. There was no discernible difference in final model performance from the input vector, so the 256 vector was chosen as a middle ground. Now having the features, three different models were chosen to carry out the classification problem: Naive bayes, random forests, and deep learning.

Naive bayes is a popular model for text classification, making it a good baseline model to determine what is good. Initially, the input features for Naive Bayes were the raw one-hot word vectors, where the traditional interpretation of the naive bayes model holds. However, improved performance arises when the input is set to be the encoder output and the numerical features are also added. This requires scaling the features to all be positive, so a slightly different data preprocessing pipeline is used. The increase in performance arises at a cost of losing model interpretability, as the input features to the model is no longer one-hot vectors of words.

Random forests are the second models explored. Random forest models are ensembles of decision trees which are able to quickly learn nonlinear mappings. These models have relatively few hyperparameters, and is able to fit nonlinear targets effectively. Using one-hot vector inputs would result in difficulty training due to the expansive vocabulary size, and would result in some very high variance, as the high dimensional space allows the random forest to easily shatter the training data at the cost of achieving abysmal performance on the test set. Thus, embedding is a crucial step to lower the input feature and prevent overfitting with random forests. However, there is still a large discrepancy between training and testing performance. Limiting the depth of the decision tree and increasing the number of trees increased the testing performance, but did not remove the variance error completely.

The third type of models were neural networks. These arise naturally as a continuation of the pretrained embedding layer. Thus, several fully connected layers were added after the embedding layer. The option to freeze, or make the embedding parameters trainable was explored, but making it fully trainable resulted in a very slow training process due to the large size of the embedding model, and the model performance was not improved significantly.

Thus, most exploration was done with the embedding parameters frozen.

Experiments, Results, and Discussion

For evaluating the performance of the three models discussed above, the F1 score was used as well as the accuracy. The F1 score is the harmonic mean of precision and recall. Below, is a table of the training and test results for the three models. It can be seen that with respect to the test data, the naive bayes performed the worst and the neural network performed the best. Random Forest model had high variance issue which was expected.

Model	Train F1 Score	Test F1 Score	Num. Train Samples	Num. Test Samples
Naive Bayes	0.4796	0.4783	118915	20986
Random Forest	0.9999	0.6336	118915	20986
Neural Network	0.7027	0.7026	118915	20986

Some of the values that had to be carefully chosen were the citation ranges for the different bins. Oversampling methods were explored to try more binning, but resulted in poor performance. Thus, the distribution of the train dataset was used to split the citation space in a way that would allow each bin to have significant amount of papers in them. Otherwise a certain bin may never be predicted and be redundant. Below is the confusion matrices for each of the models explored.

Naive Bayes

Citation Bins	<60	60-120	>120
<60	0.81	0.25	0.00
60-120	0.15	0.44	0.30
>120	0.04	0.31	0.70

Random Forest

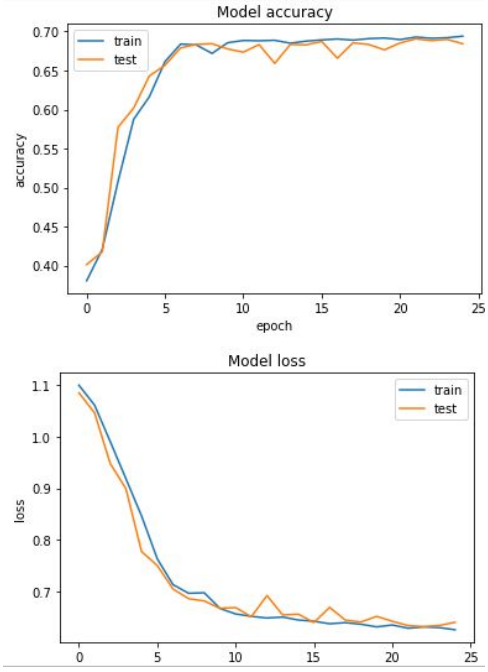
Citation Bins	<60	60-120	>120
<60	0.85	0.09	0.02
60-120	0.12	0.54	0.38
>120	0.03	0.36	0.60

Neural Network

Citation Bins	<60	60-120	>120
<60	0.78	0.08	0.00
60-120	0.17	0.61	0.33
>120	0.04	0.31	0.67

Note that most models have difficulty differentiating between the 2nd and 3rd bin: the number of misclassifications between the extreme categories is quite low, suggesting that the model is able to differentiate papers that obtain almost no citations. The hardest bin to classify is the central bin

The main source of error was bias related. The performance on the train/test set converge to very similar values. However, increasing the power of the neural network by adding more neurons or adding more layers still did not improve performance.The only model that was able to surpass this bias ceiling was the random forest, but it is not indicative of model goodness, as it came at the cost of abyssal performance on the unseen test set.



Conclusion and Future Work

Of the models explored, the neural network performed best. The Random Forest had a high variance problem and therefore generalized very poorly to the test dataset. In general, the main source of error was high bias.

Since all embeddings performed at a very similar level between all models, it suggests that the performance might be approaching a ceiling given the input features. To really improve the models, more features would have to be added. A natural continuation of the given features would be a condensed version of the abstract, where keypoint ideas and citations might give a more expansive view of what the paper really is about. In the future, it might also be intriguing to extend the system to generate relevant data features, such as the paper title. Overall, this was an interesting application that resulted in fair performance and familiarized us with various aspects of model exploration and development.

Contribution

Camilo Saavedra - Implementation of models in scikit/tensorflow and exploration of embedding space
Chung Bui - Dataset collection, data processing, implementation of models
Sophia Liu - Design and literature review, implementation of models

References

Abrishami, Ali, and Sadegh Aliakbary. "Predicting citation counts based on deep neural network learning techniques." *Journal of Informetrics* 13.2 (2019): 485-499.

Ammar, Waleed, et al. “Construction of the Literature Graph in Semantic Scholar.” *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, 2018, doi:10.18653/v1/n18-3011.

Baba, Takahiro, Kensuke Baba, and Daisuke Ikeda. "Citation Count Prediction using Abstracts." *Journal of Web Engineering* 18.1 (2019): 207-228.

Ibáñez, Alfonso, Pedro Larrañaga, and Concha Bielza. "Predicting citation count of Bioinformatics papers within four years of publication." Bioinformatics 25.24 (2009): 3303-3309.

Martín-Martín, Alberto et al. “Evidence of Open Access of Scientific Publications in Google Scholar: A Large-Scale Analysis.” OSF, 13 Oct. 2018. Web.

Yan, Rui, et al. "Citation count prediction: learning to estimate future citations for literature." Proceedings of the 20th ACM international conference on Information and knowledge management. ACM, 2011.

Weihs, Luca, and Oren Etzioni. "Learning to predict citation-based impact measures." Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries. IEEE Press, 2017.

Code Link

The following links to a zip file containing the code for the project.

https://stanford.box.com/s/399kb2l6akc64f7d60t9juz026nrn2ur