CS229 Final Project
Making Research Easier: PubMed Co-Citations

Andrew Guan
guana@stanford.edu

Trevor Tsue
ttsue@stanford.edu

15 December 2017

Abstract

In this joint project with CS221, we sought to improve scientific publication search engines by determining the co-citations of scientific articles. We define co-citation between two papers as whether or not there is a third paper that cites both of those paper. This could vastly improve search engines because two papers may not be linked by citing each other, but another paper that cites both creates a creates a new link between the papers. Co-citation is a human level indication of similarity between two papers, because it shows that the two papers are both relevant to the understanding of another paper, and being able to predict this relationship for newer papers for which there might still be few cited by relationships would help researchers find similar papers that will eventually be co-cited. By vectorizing paper abstracts and gathering other metadata for feature vectors, we trained machine learning models on 17,000 papers about chronic obstructive pulmonary disease (COPD). In summary, our neural net was able to perform extremely well in predicting co-citation relationships, minimizing both false negatives and false positives.

1 Introduction

With the rise of online search engines, information lies at the ends of our fingertips. However, with such a massive quantity of information available, the new problem that arises is organizing and identifying useful information accurately and efficiently. With regard to research articles, there are massive search engines, such as Google Scholar and PubMed, that search through millions of articles to find those similar to the keywords in the search, but these search engines are always looking for improvement. One opportunity for improvement lies with the use of citations. While Google Scholar keeps information regarding other papers that cite a particular paper, PubMed does not even track these citations. Even further, neither have offered materials regarding co-citation. Essentially, paper A and paper B are called co-cited when there is another paper C that cites them both. Thus, there is a link between paper A and B, even though they may not be directly linked themselves. Therefore, using these co-citations, we can improve the results of search engines by allowing helping users to find previously undiscovered links. Therefore, this project aims to develop models that examine the abstracts of two papers in order to determine whether two papers will be co-cited. First, we vectorized the abstracts of each paper. Then, for each pair of papers, we take the cosine similarity, Jaccard similarity, and L_2 distance of the vectorized abstracts and combine them with other metadata, like number of shared authors, number of shared papers they cite, and difference in publication date, to build to a feature vector that we put into a logistic regression model and a neural network. These models then classified the whether this pair of papers is co-cited or not.

Since this is a joint project with CS221, we examined other machine learning models like the SVM to classify the papers as co-cited. In addition, we examined how the models compare against a oracle, which we defined to be human judgment of the papers, since human researchers will be the ones that have a final say about the relevance of co-cited papers.

2 Literature Review

Small (1975) notes that co-citation corresponds to significant connections between papers and is useful for studying the development of a particular field and its relationships to other fields [7]. With this idea in mind, we want to create an algorithm that discovers these connections. Currently, many search engines use the BM25 ranking algorithm, which ranks documents according to query by examining the frequency of words in a given document [5]. Similar to this work, we instead want to examine the frequency of words in the abstract of an article, emphasizing the less common key words in the abstracts and ignoring the more frequently used words that hold little meaning. Thus, we want to use this information to provide similarities between papers, which we will use to determine whether

the papers are co-cited. Also, since we are working with abstracts form PubMed, we wanted to examine the current related articles model implemented by PubMed. Lin and Wilbur (2007) designed a probabilistic topic-based model for similarity of content based on the Poisson distribution [3]. Instead of relevance, this model focuses on relatedness, the probability that a user would want to examine a document. Although this journal also works with data from PubMed, our approach searches for co-citations of papers instead of relatedness. Further, their model does not use any type of machine learning in finding relatedness between two papers. Through the use of supervised learning, we hope to be able to better predict the co-citation of two papers. Thus, while we are both searching for similarity among papers, this article examines the probability a user wants to examine the document while our model looks at whether the articles are co-cited.

Furthermore, Lin (2009) studied if searching the full text was more effective than just the abstract using the BM25 algorithm. The results conclude that searching the full text was not more effective than the just the abstract, but examining segments of the full text did outperform just the abstracts [6]. To maintain the consistency and recognize the difficult of accessing segments form the actual paper, we chose to use just the abstract, knowing that it is just as accurate as examining the entire text. Finally, during the examination of co-cited papers clusters, Boyack et al. (2013) argues that the use of the jaccard similarity is more useful than the use of cosine similarity [8]. With this in mind, we chose to combine both similarity methods in our model in order to find the most accurate model possible.

3 Data

The data we collected for this project originates from Google Scholar and from the National Center for Biotechnology Information (NCBI) website library known as PubMed, which houses the abstracts and reference for the biomedical literature for the National Institute of Health [10]. Specifically, we gathered the titles, authors, and abstracts of biomedical articles pertaining to chronic obstructive pulmonary (COPD) disease from PubMed. Then, we collected the other papers that cited these articles from Google Scholar. We chose to examine articles about COPD disease because we wanted a topic that is well-grounded in the research field and has enough papers to build an accurate model but not too many articles such that it requires high computational power and is not efficient. We retrieved 2,500 papers that were published before 2010, for a total of 2,023,566 pairs of papers in the training set and 867,244 pairs of papers in the test set. An example paper is shared below.

```
16773. Calif Med. 1967 May;106(5):354-8.

Chronic obstructive pulmonary emphysema. Is exercise beneficial?

Ambrus L, Thal SH, Weinstein SB, Warnecke J.

Following a six-week program of training in a series of exercises,
a significant number of patients with chronic obstructive pulmonary
emphysema showed decided improvement in functional activity. Subjective
improvement also was noted and kept the patient motivation high.
Preliminary observations indicated that the improvement could be
maintained long after the end of the training period.


PMCID: PMC1502724
PMID: 6046043  [Indexed for MEDLINE]
```

3.1 Representing the Data

In order to use our abstracts in our algorithms, we needed to first represent them in a form that our models can use. To achieve this, we utilized the vector space model to represent them, where each term in the abstract corresponds to a dimension in this vector space model. This model uses the frequency of terms to derive semantic meaning from the abstracts [1].

3.2 Preprocessing the Data

To preprocess our data, we are using the TF-IDF (term frequency–inverse document frequency) weighting system for each term to find out the importance of a term to the document. TF-IDF increases in value as it appears more often in a document, but this value is offset by the frequency of the word in the entire collection of documents, since some words appear more frequently in general [2]. Thus, this weighting enhances the importance of relevant words of a document while mitigating the importance of common, irrelevant words. Additionally, we also implemented word-stemming according to the Porter2 Algorithm, which strikes a balance between not being too aggressive and actually stemming a non-trivial amount of words. This algorithm is also fairly well used in the NLP community [4].

3.3 Building Feature Vector

1. Cosine Similarity:
   $cos(\theta) = \frac{a \cdot b}{\|a\|\|b\|}$
2. Jaccard Similarity:
   $J(a,b) = \frac{|a \cap b|}{|a \cup b|}$
3. L_2 Distance:
   $L_2(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$

We defined our feature vector to capture ℓ_2 similarity, Jaccard similarity, and cosine similarity between the vectorized forms of the abstracts (which were the three similarity metrics that we experimented with for our K-Means experiments). We also included relationships between the three metrics (i.e. $\ell_2 * Jaccard$, $Jaccard * Cosine$, etc). We also thought that the

number of shared authors was relevant - if two papers shared authors, the likelihood that the papers are related should increase, since it is more likely that an author will publish in the same field as a previous paper than in a separate field. Similarly, we thought the number of shared citations was relevant as well - if two papers both cited the same paper, that paper is relevant for both of our test vectors, so the likelihood that the two papers are related should be greater. Note that this is slightly different than our attempted prediction, which is whether or not there is a third paper that cites both papers of interest. Finally, we thought that the difference in publication date might be relevant - two papers published closer in publication date are more likely to be solving similar questions within a field.

4 Methods

With our preprocessed data from the two original vectors now placed into the input feature vector, we wanted to build a supervised learning model to determine whether these two papers were co-cited. For our baseline, we made a vector of the counts of each word in the abstract without any additional preprocessing and use a similarity metric between these two vectors to build the feature input vector. This algorithm becomes our lower bound that we want to improve. For our oracle, we used human judgment to determine the similarity between two papers. For our models, we tried two different approaches, a logistic regression model and a Neural Net model, to classify our feature vector as either 1 (co-cited) or 0 (not co-cited).

For our first attempt at classification, we used stochastic gradient descent, which finds a θ that attempts to minimize the cost function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\|\theta\|$$

where the λ term is an added regularization term and $\|x\|$ is defined as

$$\|x\| = \sqrt{\sum_i x_i^2}$$

This regularization term encourages a smaller θ in order to mitigate overfitting. The stochastic gradient descent algorithm minimizes the loss function by updating θ for each example by finding the gradient of the objective function above. Therefore, for each training example $x^{(i)}, y^{(i)}$, we update θ as follows:

$$\theta := \theta - \alpha((h_\theta(x^{(i)}) - y^{(i)})x^{(i)} + \frac{\lambda}{2}\theta)$$

The learning rate that we used was $\alpha = 0.01$, which we found to work well without taking too long to converge. The regularization term we used was $\lambda = 0.0001$, which we found to strike a good balance between discouraging variance while not encouraging bias.

For our second attempt at classification, we used a neural net with one hidden layer with 100 neurons using a RELU activation function g(z), which is defined as follows:

$$g(z) = \max\{0, z\}$$

The output node is calculated using softmax(x), which is defined as:

$$softmax(x)_j = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

Our neural net attempts to find two sets of weights, $W^{[1]}, b^{[1]}$ and $W^{[2]}, b^{[2]}$ such that we minimize the Loss function:

$$L(\hat{y}, y) = -[(1-y)\log(1-\hat{y}) + y\log\hat{y}]$$

where $\hat{y}$ is defined as:

$$\hat{y} = \arg\max[softmax(W^{[2]}g(W^{[1]}x^{(i)} + b^{[1]}) + b^{[2]})]$$

This is achieved by updating each parameter by taking the gradient of the loss function with respect to that parameter and updating the parameters once per batch. The mini batch size we used was 200, which we found to strike a good balance between moving through our relatively larger dataset quickly while not having each batch take too long.

5 Results

We ran our algorithms on 2,895,621 pairs of papers with a total of 20,012 co-citations (0.7% of the total comparisons). We split 70% of our data into a training set of 2,023,566 pairs with 14,091 co-citations (Table 1). The remaining comprised our test set of 867,244 pairs with 5,921 co-citations (Table 2). Because of the sparsity of the data, we decided to include some more metrics than training and test error, so we use the following metrics to analyze the effectiveness of our models.

$$\begin{aligned} Precision &= \frac{tp}{tp+fp} \\ Recall &= \frac{tp}{tp+fn} \\ F1 &= 2 * \frac{precision * recall}{precision + recall} \\ logloss &= -logP(yt|yp) \\ &= -(ytlog(yp) + (1-yt)log(1-yp)) \end{aligned}$$

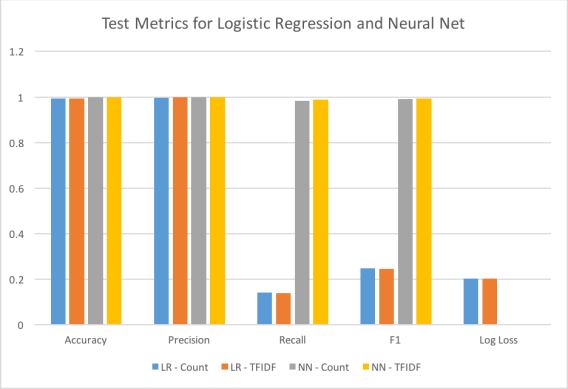
where tp = true positive, fp = false positive, fn = false negative, $yt \in \{0, 1\}$ is the actual label, and yp is the probability that yt = 1.

Table 1: Training Set - 2,023,566 pairs, 14,091 co-citations

	Logistic Reg.		Neural Net	
Metric	Count	TF-IDF	Count	TF-IDF
Accuracy	0.993	0.993	0.999	0.999
Precision	0.998	0.998	1.000	1.000
Recall	0.137	0.138	0.983	0.986
F1	0.241	0.242	0.991	0.992
log loss	0.207	0.207	0.003	0.003

Table 2: Test Set - 867,244 pairs, 5,921 co-citations

	Logistic Reg.		Neural Net	
Metric	Count	TF-IDF	Count	TF-IDF
Accuracy	0.994	0.994	0.999	0.999
Precision	0.997	0.998	1.000	1.000
Recall	0.141	0.140	0.984	0.987
F1	0.247	0.246	0.992	0.993
log loss	0.202	0.202	0.003	0.002



Neural Net Confusion Matrix

		p	**n**	**total**
actual value	**p′**	True Positive: 5921	False Negative: 102	P′
	n′	False Positive: 0	True Negative: 861,221	N′
	total	P	N	

6 Discussion

As mentioned before, most pairs of papers are not co-cited. Therefore, accuracy is not the best metric for our model because a model that classifies all the examples as 0 will still have a near perfect accuracy, since only 0.7% papers are actually co-cited. Thus, we want to examine other metrics like the rate of false positives and false negatives. All of our models have a very high precision, resulting in a very low rate of false positives. While limiting the number of false positives makes a stronger model, the researchers will still filter out the unrelated papers when going through the list of possible co-cited articles. Thus, the rate of false negatives is the most important metric, as if a pair of papers is a false negative, then the researcher will never view the related paper in the first place. Thus, we want to maximize the recall. On this note, F1 gives insights to how well both precision and recall performed, so we want to maximize this as well. Finally, the log loss measures the accuracy of the model by examining the probability the entry is classified as postive or negative, so we want to minimize this loss function as well.

As expected, the TF-IDF outperformed the count vector in this experiment in all categories. With regard to precision, TF-IDF had a better ratio of true positives to true positives + false positives. The recall is also higher for TF-IDF, so there are fewer false negatives. With regard to both the count and TF-IDF vector, these vectors had a very limited difference for precision and accuracy. However, since most of the not co-cited papers determine the accuracy and the precision, we can see that the abstracts of these papers would not likely by similar in the first place, resulting in large distances regardless of the count or TF-IDF vectors. For recall, on the other hand, there is a bigger difference of about 0.003 for the count neural net and TF-IDF neural net. Since recall focuses more on the actual co-cited articles, we can see that the TF-IDF places a greater emphasis on the key words and less on the more general words. Thus, the key words have more weight, and papers with similar key words are have similar vectors and are more likely to be classified as co-cited, reducing the number of false negatives.

In comparing the two models, the logistic regression performed comparably to the neural net in producing few false positives, as precision was high in every model. However, the recall of the Logistic Regression model was lacking. This is likely because our Logistic Regression model can only capture linear relationships between the input data features and θ; for many papers, it is simple to predict co-citation in a few select cases by simply examining the metadata between the two papers. For example, a shared author is a strong indication that two papers have a co-citation relationship, especially because it is common for a researcher to cite their own work in future works or to cite multiple works from a friend. However, as evidenced by the neural net model, there might also be non-linear rela-

tionships between the features that we extracted. The logistic regression model would not be able to capture these non-linear relationships, which explain why recall was significantly lower in both the count and TF-IDF implementations of the Logistic Regression model as compared to the neural net.

Throughout our experiments, we were especially careful to avoid overfitting to our training data. To mitigate this, in both the logistic regression model and the neural net model, we included a regularization constant to discourage overly complex models. As evident by comparing Table 1 and Table 2, the metrics did not change much between the training data and the test data. Therefore, we conclude that we did not overfit to the training data.

7 Summary

In summary, we predicted co-citation relationships on papers retrieved from PubMed on COPD that were published before 2010 in order to try and predict co-citation relationships of papers in the present. We tried both a Logistic Regression model and a Neural Net model. Both models succeeded in producing few false positives, but the neural net model vastly outperformed the logistic regression model in producing few false negatives as well.

7.1 Future Work

Two areas are prominent in exploring future work. The first arises from reexamining the process of research. The original goal was that given that a researcher is interested in paper A, to find a paper B that would be likely to contribute to research given paper A. However, research is rarely conducted in such a straightforward manner - researchers are generally actually interested in a set of papers A, B, and C, and might be interested in a fourth paper D. This problem is slightly different, because the feature extraction would no longer take in two papers, but rather a set of papers and another paper. Further, the output of the model could change as well; given a set of papers, it might be very likely for that paper to have a co-citation relationship with one of these papers, so the output of the model might change to a real number, which could be the number of times that paper is co-cited with a paper in our set of interest.

The second area is closely related the the first. In this paper, we only examined one particular field (COPD) - could these models work just as well across different fields? For example, I might be interested in the frequency of co-citation between a paper on COPD and a paper on Alzheimer's. Relating to the first area, given a set of papers that spans multiple fields, could this model work just as well in predicting co-citation relationships between a paper from a different field?

8 Contributions

Andrew is responsible for the development of the feature extractor for the development of the input vector, the development multi-threading process to download the citation information from Google Scholar, the preprocessing the data, and the organization of the material into the cohesive main function. Trevor is responsible for parsing the papers and abstracts to build the original word vectors, designing of the code to download citation information from Google Scholar, and implementation of the learning models from the preprocessed data. Both authors contributed to writing this project.

We also received advice from Bohdan Khomtchouk, who is a postdoc working in the Gozani Lab. Bohdan has been incredibly helpful with us in coming up with the project, finding next steps, and pointing us in the right direction for where to find the data and how to process it. However, all the code and all the writeup was written by Andrew and Trevor.

9 References

1. Turney, Peter D. and Patrick Pantel. "From Frequency to Meaning: Vector Space Models of Semantics." *Journal of Artificial Intelligence Research*, vol. 37, 2010, 141-188.

2. Ramos, Juan. "Using TF-IDF to Determine Word Relevance in Document Queries." 2003.

3. Lin, Jimmy and W John Wilbur. "PubMed related articles: a probabilistic topic-based model for content similarity." *BMC Bioinformatics*, 8:423, 2007. doi:10.1186/1471-2105-8-423

4. Willet, P. "The Porter stemming algorithm: then and now." *Electronic Library and Information Systems*, 40:219, 2006. doi:10.1186/1471-2105-8-423

5. Robertson, Stephen and Hugo Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond", Foundations and Trends in Information Retrieval: Vol. 3: No. 4, pp 333-389, 2009. doi:10.1561/1500000019

6. Lin, Jimmy. "Is searching full text more effective than searching abstracts?" *BMC Bioinformatics*, 10:46, 2009. doi: 10.1186/1471-2105-10-46

7. Small, Henry. "Co-citation in the scientific literature: A new measure of the relationship between two documents." *Essays of an Information Scientist*, Vol:2, p.28, 1974.

8. Boyack, K., Small, H., and R. Klavans. "Improving the accuracy of co-citation clustering using full text." *Journal of the Association for Information Science and Technology*, 64:9, 2013. doi: 10.1002/asi.22896

9. Pedregosa et al. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, 12, pp. 2825-2830, 2011.

10. National Center for Biotechnology Information (NCBI)[Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – [cited 2017 December 14]. Available from: https://www.ncbi.nlm.nih.gov/

10 Code Libraries Used

1. Scikit-learn for implementing logistic regression and neural net models.

2. Google scholar API for retrieving cited by information

3. PubMed API for retrieving retrospective citation information