Edge prediction: Predicting Edge in Academic Citation Networks

Abstract

This paper attempts to assess the performance of various methods for predicting the citation of academic articles. Many researchers have sought to predict the future citation of new articles, and this interest has resulted in researchers using various machine learning methods for prediction. Our work asks a slightly different but related question. Given an article, how likely is it to cite another particular article? For our specific task, we found that sophisticated graph structure based model does not achieve very promising performance. To this end, we developed an intelligent and novel feature engineering pipeline that could generate highly accurate predictions with relatively simpler models. We achieve around 95% F1 score with random forest classifier with our engineered features, which largely outperformed the graph neural network based model.

1 Introduction

Analysis of academic research networks is an widely studied problem in the machine learning community. Some machine learning tasks related to academic network analysis include author name disambiguation, topic modeling, and expert identification. This study focuses on another task—citation prediction. Understanding and predicting citations is important for assessing the impact of an article [6**?**], understanding how knowledge flows through academic networks networks[4], and and potentially can be used to evaluate discrimination[9]. Predicting citation provides a challenging machine learning task. Academic citations take on a directed graph structure where an article is more likely to cite papers connected downstream of it on the network. In addition, citing a given article is relatively rare, making the learning task more difficult. Finally, other key features that one may want to use to predict citations, such as the paper's topic, are often not included in the metadata and must be learned as well. This study assesses the performance of various models and the impacts of various engineered features on accurately predicting citations. Furthermore, this paper proposes a novel and intelligent feature engineering pipeline that is able to generate outstanding citation prediction performance with very simple classifiers, largely outperforming sophisticated graph structure based deep models.

2 Related Works

Predicting academic citations has been of interest to machine learning researchers for years. A variety of researchers have attempted to predict citations using various methods and datasets. The goals of these papers often vary and include predicting the future number of citations, predicting the impact of an article on the ranking of researchers, and predicting the impact of articles on the impact factor of a

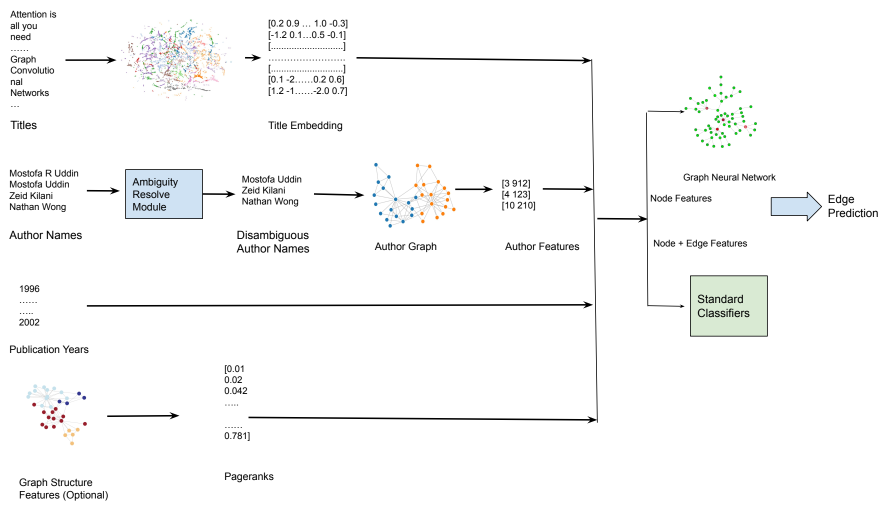


Figure 1: Schematic Diagram of Our Pipeline

journal. This research grappled with determining the time cutoffs for prediction (short-term citation or long-term citation prediction), identifying key features, and exploring new prediction methods.

Wong et al. identified preferential attachment (tendency to continue to cite highly cited papers), decay rate, and similarity as the critical mechanisms for predicting citations [13]. Meanwhile, other researchers found that various features such as journal prestige, author prestige, article length, number of citations can help predict future citations. [2]. Researchers have classified these features into four categories: author features, journal features, paper features, and others. Researchers have used various types of regression, decision trees, and support vector machines to predict citations using these features.

In addition to these features, researchers noted that the network structure was a vital component to predicting citations. Various researchers used network features such as network centrality to predict citation. Meanwhile, other researchers attempted to use more of graph structure by applying techniques such as Recurrent Neural Networks for prediction.[1].

Several researchers have explicitly approached citation prediction as a link prediction problem [11] [3] [5]. Our paper frames the problem in a similar way. Thinking about citation prediction as a link prediction problem opens us up to learning from a broad set of literature.

Other work has examined link prediction more broadly. Link prediction often relies on particular heuristics [16]. Some are classified as first-order heuristics, which depend on neighbors only. Extending from there, you can have second-order (relying on data two hops away) or h-order (h hops away). Other heuristics try to learn from the entire graph. Zhang and Chen point out that most higher-order heuristics rely on exponential decay. As a result, these proposed using graphical neural networks for link prediction. In particular, their SEAL method learns general graph structure features from local enclosing subgraphs.

3 Problem Definition

The edge prediction problem essentially takes two papers $u, v \in \mathcal{R}^1 \times d$ with associated information as input and predicts whether there is an edge between two. This can be reduced to a binary classification problem where given two inputs u and v, the task is to predict whether $f(u, v) = 1$ or 0.

In our study, we use the aminer.org citation network V1 dataset [12] to solve this problem. This dataset scraped information from hundreds of thousands of academic papers and contains basic information such as the paper title, authors, year of publication, the publication venue, abstract, and the identifiers for the other papers cited by the article.

We created a graph representing the relationship between different articles using this dataset. The number of nodes in the citation graph is 629,814. Using the references contained in the data, we found 631,492 positive edges (when a paper cites another paper) in total. These edges represent a small fraction of the over 396,664,414,968 possible edges (when including direction).

4 Methods

4.1 Preparing Train and Test Datasets

The citation network dataset is not split between test and training data. There are potentially multiple ways to construct the training and test sets. For instance, some papers focus on predicting future citations. A reasonable approach might aim to predict the citations by papers at terminal nodes (articles in which no other papers cite). Such an approach does not fully use the riches of the citation graph. As a result, in this paper, we opted to predict the existence of graph edges throughout the graph.

The entire graph contains 631,492 edges (citations) and all were used as positive examples in our dataset. Since our problem is essentially a binary classification problem, negative examples are also required for training and evaluation. However, the total possible number of negative examples (missing edges in entire graph) is huge (and it would not be feasible to train any classifier to with all of them. Therefore, we randomly sampled the negative examples from the total possible examples to create our datasets. To observe the effect of imbalance between the number of positive and negative examples in a dataset, we created three datasets- **Dataset-I**, **Dataset-II**, and **Dataset-III** with different ratio of positive and negative examples. For **Dataset-I**, we sampled negative examples equal the number of positive examples. In **Dataset-II** and **Dataset-III**, we sampled negative examples twice and thrice the number of positive examples respectively. Further, for training and evaluation, we randomly split the positive and negative edges in training and testing sets with 70:30 ratio for each dataset. While training with graph structure features, we recreated the graph by removing positive edges from the test set in adjacency matrix for a fair evaluation on test set.

4.2 Feature Engineering

Compared to other academic citation datasets, the available features in Citation Network V1 are relatively limited. In addition, many observations are missing data for many of the features. As a result, we generated three key features to include in our model.

4.2.1 Author Name Disambiguation & Feature Generation

The author of paper may be a key factor in predicting citations. For a paper to be cited, an author would likely need to know about the existence of the paper. Diffusion of knowledge among individuals is important aspect of this process. A recent working paper assessed the flow of new academic ideas through co-authorship networks [4]. As a result, we decided to a include features to the authors of articles and their network of coauthors into the model.

To accurately capture the relationship between authors, one must precisely determine the authors of a given paper. This task can be challenging because the same author can appear under different names, and two different authors may have similar names. Author disambiguation is its machine learning task that is separate from the primary goal of our analysis.

Existing survey articles have explored the variety of ways of conducting author name disambiguation [8]. Many machine learning methods rely on additional information such as citation graphs, topic similarities, and venue of publication, in addition to the author name to distinguish authors. Instead of taking these approaches, we used a heuristic-based approach that relied on the name feature alone. We took this approach because our ultimate goal is to predict citation, not authors. Some of the additional features used in disambiguation are already features in our model for citation prediction. As a result, we opted to take an approach that may generate unique information.

Overall we found over 1,300,000 paper-author combinations. When analyzing exact string matches, we saw over 450,000 unique authors. We then sought to combine these authors with authors that only had last names and first initials. After disambiguation, we reduced our sample to 430,000 unique authors. We generated author ids for each author.

4.3 Author Impact Factor Heuristic

One heuristic we explored to generate author features originates from the belief that authors who already possess a large number of citations are more likely to have their work (present and future) cited more than an author in the same field with a fewer number of citations. To define a heuristic to measure the 'impact factor' of each author, we construct an adjacency matrix, where each node is a paper, and edges in this DAG represent the authors on that paper who have also contributed to the papers in the incoming nodes. We utilize the implicit hashing function used in a hashmap to ensure unique authors and maintain constant lookup time.

Next, we perform breadth-first search for every paper (because ultimately we are trying to enrich features for any particular paper in our dataset), to see what other papers the authors on the original paper have also authored. As a simple heuristic to approximate the impact factor of that paper and its authors, we measure the number of 1st and 2nd descendants of the original paper, and include these numbers as a 2-d feature vector for each paper. Qualitatively, the greater the number of papers each author in a paper has authored, the more likely this paper is to have a large number of citations and thus possibly more likely to be cited than another paper whose authors are less well-connected. We used this as a straightforward method to incorporate author relationships in our dataset.

4.3.1 Title Embedding Generation

One key source of information is in the titles of each paper. Authors would generally describe the main content of their paper in the title, and so it follows that papers with similar content will have similar titles. Our objective to engineer title features is to compute title similarity based on vectorized titles.

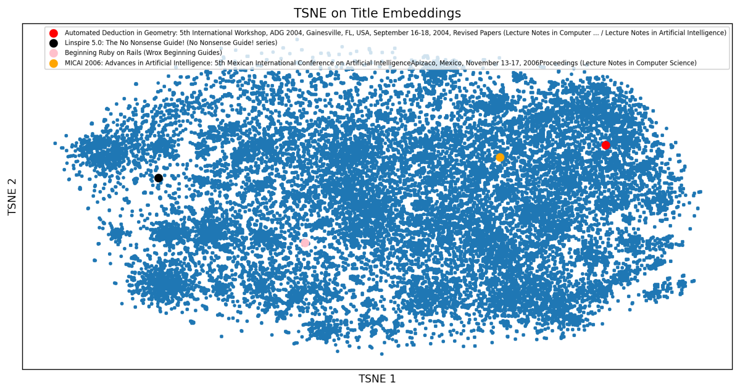


Figure 2: TSNE visualization of title embeddings

To accomplish this, we use a distilled pre-trained language transformer model [15] that was fine-tuned for a contrastive learning objective. The model was trained with non-specific text, so the learned embeddings were not fine-tuned for academic titles or language. This is something we attempted to explore, but ultimately decided it was infeasible due to computational constraints.

Each title was first truncated to a pre-defined number of characters, in case the length of the title was unusually long. Next, the title words were tokenized and inputted to the transformer model. From there, each title would be transformed into a feature vector of length 384. We used this feature vector as a node feature for each of our citations in the network.

To evaluate the effectiveness of this embedding method, and particularly to visualize whether it was able to identify similarities among papers with common foci, we performed TSNE on the entire embedding space, as is see in Figure 2. We can see here that titles with similar content (such as AI papers) are closer together than papers with different aims or objectives.

4.3.2 Creation of Citation Network

We used networkx python package to create the entire citation network. Afterwards, we created some graph structure based features to be used in our models. We initially tested some first order and second order heuristics (e.g, jaccard co-efficient, resource allocation index) to extract graph structure features which did not show much promise. A more elegant solution to extract structure based features determining the 'importance' of certain citations was performed through the PageRank algorithm [10]. After we construct an adjacency matrix for the citation network, we use the ranking of each node (from the PageRank algorithm) as an additional feature we can use to determine the importance of each paper. This is one method of predicting potential links between two nodes, as it is more likely for a node to cite an important node (ie. a paper that has been cited more than others). However, such features cannot be used for unseen nodes in the graph, and has been kept as optional.

4.4 Models

As models that extensively leverages graph structure representations, we used deep graph neural networks. As models that emphasizes on feature associated only with the binary classification problem, we used standard classifiers, e.g, k-nearest neighbor, logistic regression, and random forest classifiers.

4.4.1 Graph Neural Networks

We use the entire training graph and all node features (title embeddings, years, author features) to create our graph models. Each node is represented as a vector v where, $v \in \mathcal{R}^{1\times 388}$. To represent the graph, we used Deep Graph Library [14] package. We used a sequential Graph Convolutional Networks from GraphSAGE [7] to learn the inductive representation of each node (papers) in the graph. Graph Convolutional Networks are used to aggregate the node feature informations across each node's neighborhood in the model. These features are further fed into a 3 layer fully connected neural network model to predict whether two nodes will have an edge between them or not. In other words, node embeddings of two nodes are used to see whether one will cite the other. The model is trained in a supervised fashion where binary cross-entropy is used as loss function. Binary cross-entropy loss function can be expressed as follows:

$$\mathcal{L} = -\sum_{(u,v)\in\mathcal{G}} y_{(u,v)} \log \hat{y}_{(u,v)} + (1 - y_{(u,v)}) \log(1 - \hat{y}_{(u,v)})$$

The entire model was implemented in pytorch with gpu acceleration.

4.4.2 Standard Classifiers

We used only citation link associated features to train standard classifiers, e.g., logistic regression, k-nearest neighbor and random forest classifier. For each citation, we used the publication years ($\mathcal{R}^1$)of both end nodes, author features ($\mathcal{R}^2$) of both end nodes, **similarity** (dot product) between title embeddings of two end nodes ($\mathcal{R}^1$) and optional pagerank features ($\mathcal{R}^1$) of two end nodes as features for prediction. Hence each citation edge (positive or negative) is presented as a $\mathcal{R}^{1\times 9}$ dimensional vector. For logistic-regression we used elastic net with varying L1-L2 rations. However, no significant change was observed with the change of L1-L2-ratios and in our evaluation, we report the models tested with L2 penalty. For k-nearest neighbor, we defined k as 5. For random forest classifier, we used 100 decision tree estimators where trees are expanded nodes are expanded until all leaves are pure or until all leaves contain less than 2 samples. The classifiers were implemented using python's scikit-learn library.

5 Results

5.1 Evaluation Metrics

There exists several evaluation metrics, e.g, F1-score, P5, P10, Mean Average Precision (MAP), etc. in the literature. However, F1-score, precision, and recall are easily interpretable and are most widely used by the machine learning community. To this end, we use F1-score, precision and recall in our binary classification problem (predicting whether an edge exist or not). The evaluation metrics are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$
$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

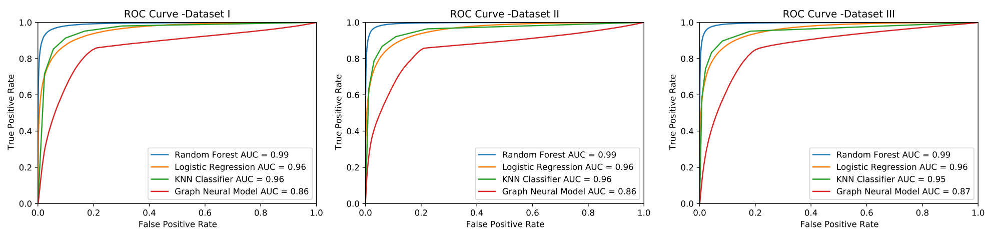
Here, TP, FP, FN means True Positive, False Positive, and False Negative respectively.

5.2 Quantitative Validation

The evaluation scores against test datasets of Dataset-I, Dataset-II, and Dataset-III are presented in Table 1. In each cases, the models are trained with the training data of the corresponding dataset types. From the table, it is evident that edge feature based standard classifiers outperformed the deep node representation learning based classifier. Even the liner logistic regression classifier demonstrated a better accuracy. This indicates that graph structure based features does not come very useful in the prediction task in our dataset. The AUC-ROC curve in Figure 3 further validates this claim. However, in all cases, random forest classifier shows excellent performance.

Table 1: F1-score, precision, recall values for methods tested against the datasets using the engineered features. F1-score, precision, and recall are represented with F1, P, and R respectively in the table.

Method	**Dataset-I**			**Dataset-II**			**Dataset-III**		
	F1	P	R	F1	P	R	F1	P	R
Graph Neural Network	0.83	0.761	0.857	0.749	0.631	0.794	0.65	0.518	0.621
Logistic Regression	0.887	0.849	0.875	0.844	0.776	0.815	0.816	0.724	0.772
K-NN Classifier	0.908	0.868	0.913	0.873	0.807	0.867	0.849	0.764	0.833
Random Forest	**0.952**	**0.941**	**0.933**	**0.956**	**0.929**	**0.952**	**0.944**	**0.908**	**0.936**

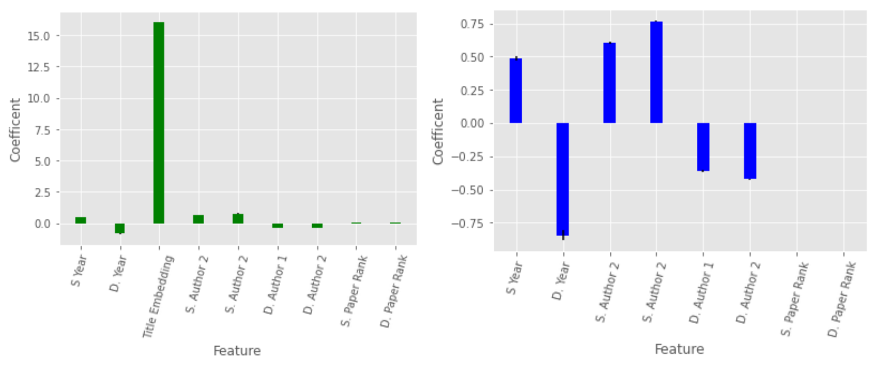


(a) (b) (c)

Figure 3: ROC-AUC curves against the three test datasets used in our method.

5.3 Feature Importance Analysis

We further investigated the feature importance leveraged by our classifier. Figure 4 shows the coefficients for the various features across our logistics regression models. All features had been scaled to facilitate this analysis. As a result, these coefficients shows the relative importance of each feature in prediction. We averaged the coefficients across the 3 analyses with differing negatives nodes. The figure includes standard deviations across these 3 values. The standard errors are fairly small, suggesting that increased sparsity did not impact feature importance. The title embedding feature was by far the most important feature for prediction. Other features had similar levels of importance, except for the graph features had very little importance. This fact again testifies that graph-structure based features barely contribute to prediction in our specific problem.



(a) (b)

Figure 4: Feature importance by logistic regression co-efficients against the datasets. The error bar represents the deviation of feature co-efficient values across the dataset.

6 Discussion and Analysis

One key result of our research suggests that the title embedding feature was the most important feature in our model. This makes intuitive sense because papers likely cite work on similar topics. This result also has important implications for predicting the likelihood that a new article should cite an old article. A new article would be a terminal node of citation network since it cannot be cited by others. As a result, one will need to predict the paper's first edges into the network. The title embedding feature has just high importance in prediction ensures that this would be possible.

We also found that the Random Forest Classifier performed the best. It was also less sensitive to changes in sparsity compared to other classifiers. Counter to what we expected, the graphic neural network model performed the worst. One possible explanation is that given the sparsity of edges, the GNN may struggle to construct the local subgraph. We hope our findings will bring intelligent insights in such prediction task to the community.

References

[1] Ali Abrishami and Sadegh Aliakbary. Predicting citation counts based on deep neural network learning techniques. *Journal of Informetrics*, 13(2):485–499, 2019.

[2] Xiaomei Bai, Fuli Zhang, and Ivan Lee. Predicting the citations of scholarly paper. *Journal of informetrics*, 13(1):407–418, 2019.

[3] Ertan Bütün, Mehmet Kaya, and Reda Alhajj. A supervised learning method for prediction citation count of scientists in citation networks. ASONAM '17, New York, NY, USA, 2017. Association for Computing Machinery.

[4] Wei Cheng and Bruce A Weinberg. Marginalized and overlooked? minoritized groups and the adoption of new scientific ideas. Working Paper 29179, National Bureau of Economic Research, August 2021.

[5] Ali Daud, Waqas Ahmed, Tehmina Amjad, Jamal Nasir, Naif Aljohani, Rabeeh Abbasi, and Ishfaq Ahmad. Who will cite you back? reciprocal link prediction in citation networks. *Library Hi Tech*, 35:00–00, 10 2017.

[6] Eugene Garfield. Citation analysis as a tool in journal evaluation. *Science (American Association for the Advancement of Science)*, 178(4060):471–479, 1972.

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[8] Ijaz Hussain and Sohail Asghar. A survey of author name disambiguation techniques: 2010–2016. *The Knowledge Engineering Review*, 32, 2017.

[9] Marlène Koffi. Gendered citations at top economic journals. *AEA Papers and Proceedings*, 111:60–64, May 2021.

[10] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[11] Nataliia Pobiedina and Ryutaro Ichise. Citation count prediction as a link prediction problem. *Applied Intelligence*, 44:252–268, 2015.

[12] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998, 2008.

[13] Dashun Wang, Chaoming Song, and Albert-Laszlo Barabasi. Quantifying long-term scientific impact. *Science (American Association for the Advancement of Science)*, 342(6154):127–132, 2013.

[14] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, et al. Deep graph library: Towards efficient and scalable deep learning on graphs. 2019.

[15] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. 2020.

[16] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5171–5181, Red Hook, NY, USA, 2018. Curran Associates Inc.