Content-Based Citation Recommendation

Chandra Bhagavatula
Allen Institute for AI
chandrab@allenai.org

Sergey Feldman
Data Cowboys [*]
sergey@data-cowboys.com

Russell Power
Independent Researcher [†]
russell.power@gmail.com

Waleed Ammar
Allen Institute for AI
waleeda@allenai.org

Abstract

We present a content-based method for recommending citations in an academic paper draft. We embed a given query document into a vector space, then use its nearest neighbors as candidates, and rerank the candidates using a discriminative model trained to distinguish between observed and unobserved citations. Unlike previous work, our method does not require metadata such as author names which can be missing, e.g., during the peer review process. Without using metadata, our method outperforms the best reported results on PubMed and DBLP datasets with relative improvements of over 18% in F1@20 and over 22% in MRR. We show empirically that, although adding metadata improves the performance on standard metrics, it favors self-citations which are less useful in a citation recommendation setup. We release an online portal for citation recommendation based on our method,[1] and a new dataset `OpenCorpus` of 7 million research articles to facilitate future research on this task.

1 Introduction

Due to the rapid growth of the scientific literature, conducting a comprehensive literature review has become challenging, despite major advances in digital libraries and information retrieval systems. Citation recommendation can help improve the quality and efficiency of this process by suggesting published scientific documents as likely citations for a query document, e.g., a paper draft to be submitted for ACL 2018. Existing citation recommendation systems rely on various information of the query documents such as author names and publication venue (Ren et al., 2014; Yu et al.,

2012), or a partial list of citations provided by the author (McNee et al., 2002; Liu et al., 2015; Jia and Saule, 2017) which may not be available, e.g., during the peer review process or in the early stage of a research project.

Our method uses a neural model to embed all available documents into a vector space by encoding the textual content of each document. We then select the nearest neighbors of a query document as candidates and rerank the candidates using a second model trained to discriminate between observed and unobserved citations. Unlike previous work, we can embed new documents in the same vector space used to identify candidate citations based on their text content, obviating the need to re-train the models to include new published papers. Further, unlike prior work (Yang et al., 2015; Ren et al., 2014), our model is computationally efficient and scalable during both training and test time.

We assess the feasibility of recommending citations when some metadata for the query document is missing, and find that we are able to outperform the best reported results on two datasets while only using papers' textual content (i.e. its title and abstract). While adding metadata helps further improve the performance of our method on standard metrics, we found that it introduces a bias for self-citation which might not be desirable in a citation recommendation system. See §5 for details of our experimental results.

Our main contributions are:
- a content-based method for citation recommendation which remains robust when metadata are missing for query documents,
- large improvements over state of the art results on two citation recommendation datasets despite omitting the metadata,
- a new dataset of seven million research papers, addressing some of the limitations in

---

Phase 1:
candidate selection

d_1 cites d_4
d_1 cites d_5
d_3 cites d_7
...

query
document
d_q

document
embeddings

Phase 2:
reranking

nearest neighbors of d_q:

	NNRank	
d_2 d_q	NNRank →	**0.7**
d_6 d_q	NNRank →	**0.8**
d_3 d_q	NNRank →	0.5
d_4 d_q	NNRank →	0.3

cited in nearest neighbors:

| d_7 d_q | NNRank → | **0.9** |

reranked
list

d_7
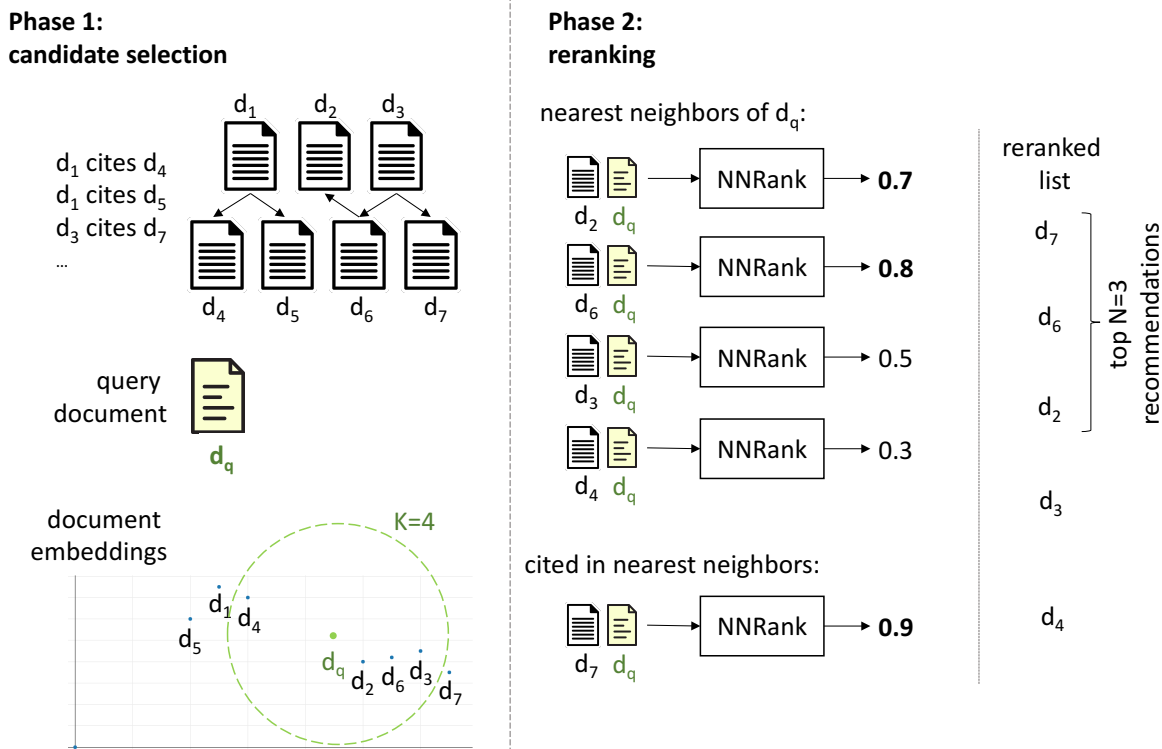d_6 top N=3 recommendations
d_2

d_3

d_4

Figure 1: An overview of our Citation Recommendation system. In Phase 1 (NNSelect), we project all documents in the corpus (7 in this toy example) in addition to the query document d_q into a vector space, and use its (K=4) nearest neighbors: d_2, d_6, d_3, and d_4 as candidates. We also add d_7 as a candidate because it was cited in d_3. In Phase 2 (NNRank), we score each pair $(d_q, d_2), (d_q, d_6), (d_q, d_3), (d_q, d_4)$, and (d_q, d_7) separately to rerank the candidates and return the top 3 candidates: d_7, d_6 and d_2.

previous datasets used for citation recommendation, and

- a scalable web-based literature review tool based on this work.[2]

2 Overview

We formulate citation recommendation as a ranking problem. Given a query document d_q and a large corpus of published documents, the task is to rank documents which should be referenced in d_q higher than other documents. Following previous work on citation recommendation, we use standard metrics (precision, recall, F-measure and mean reciprocal rank) to evaluate our predictions against gold references provided by the authors of query documents.

Since the number of published documents in the corpus can be large, it is computationally expensive to score each document as a candidate reference with respect to d_q. Instead, we recommend citations in two phases: (i) a fast, recall-oriented candidate selection phase, and (ii) a feature rich,

precision-oriented reranking phase. Figure 1 provides an overview of the two phases using a toy example.

Phase 1 - Candidate Selection: In this phase, our goal is to identify a set of candidate references for d_q for further analysis without explicitly iterating over all documents in the corpus.[3] Using a trained neural network, we first project all published documents into a vector space such that a document tends to be close to its references. Since the projection of a document is independent of the query document, the entire corpus needs to be embedded only once and can be reused for subsequent queries. Then, we project each query document d_q to the same vector space and identify its nearest neighbors as candidate references. See §3 for more details about candidate selection.

---

[2] https://github.com/allenai/citeomatic

[3] In order to increase the chances that all references are present in the list of candidates, the number of candidates must be significantly larger than the total number of citations of a document, but also significantly smaller than the number of documents in the corpus.

Phase 2 - Reranking: Phase 1 yields a manageable number of candidates making it feasible to score each candidate d_i by feeding the pair (d_q, d_i) into another neural network trained to discriminate between observed and unobserved citation pairs. The candidate documents are sorted by their estimated probability of being cited in d_q, and top candidates are returned as recommended citations. See §4 for more details about the reranking model and inference in the candidate selection phase.

3   Phase 1: Candidate Selection (`NNSelect`)

In this phase, we select a pool of candidate citations for a given query document to be reranked in the next phase. First, we compute a dense embedding of the query document d_q using the document embedding model (described next), and select K nearest neighbor documents in the vector space as candidates.[4] Following Strohman et al. (2007), we also include the outgoing citations of the K nearest neighbors as candidates.

The output of this phase is a list of candidate documents d_i and their corresponding scores `NNSelect`(d_q, d_i), defined as the cosine similarity between d_q and d_i in the document embedding space.

Document embedding model. We use a supervised neural model to project any document d to a dense embedding based on its textual content. We use a bag-of-word representation of each textual field, e.g., $d[\text{title}] = \{\text{'content-based'}, \text{'citation'}, \text{'recommendation'}\}$, and compute the feature vector:

$$\mathbf{f}_{d[\text{title}]} = \sum_{t \in d[\text{title}]} w_t^{\text{mag}} \frac{\mathbf{w}_t^{\text{dir}}}{\|\mathbf{w}_t^{\text{dir}}\|_2}, \qquad (1)$$

where $\mathbf{w}_t^{\text{dir}}$ is a dense <u>dir</u>ection embedding and w_t^{mag} is a scalar <u>mag</u>nitude for word type t.[5] We then normalize the representation of each field and compute a weighted average of fields to get the document embedding, $\mathbf{e}_d$. In our experiments, we use the title and abstract fields of a document d:

$$\mathbf{e}_d = \lambda^{\text{title}} \frac{\mathbf{f}_{d[\text{title}]}}{\|\mathbf{f}_{d[\text{title}]}\|_2} + \lambda^{\text{abstract}} \frac{\mathbf{f}_{d[\text{abstract}]}}{\|\mathbf{f}_{d[\text{abstract}]}\|_2},$$

---

[4] We tune K as a hyperparameter of our method.
[5] The magnitude-direction representation is based on Salimans and Kingma (2016) and was found to improve results in preliminary experiments, compared to the standard "direction-only" word representation.

where λ^{title} and $\lambda^{\text{abstract}}$ are scalar model parameters.

Training. We learn the parameters of the document embedding model (i.e., $\lambda^*, w_*^{\text{mag}}, \mathbf{w}_*^{\text{dir}}$) using a training set $\mathcal{T}$ of triplets $\langle d_q, d^+, d^- \rangle$ where d_q is a query document, d^+ is a document cited in d_q, and d^- is a document not cited in d_q. The model is trained to predict a high cosine similarity for the pair (d_q, d^+) and a low cosine similarity for the pair (d_q, d^-) using the per-instance triplet loss (Wang et al., 2014):

$$\text{loss} = \max \big(\alpha + s(d_q, d^-) - s(d_q, d^+), 0 \big), \quad (2)$$

where $s(d_i, d_j)$ is defined as the cosine similarity between document embeddings $\text{cos-sim}(\mathbf{e}_{d_i}, \mathbf{e}_{d_j})$. We tune the margin α as a hyperparameter of the model (see Appendix B for more details). Next, we describe how negative examples are selected.

Selecting negative examples. Defining positive examples is straight-forward; we use any (d_q, d^+) pair where a document d_q in the training set cites d^+. However, a careful choice of negative training examples is critical for model performance. We use three types of negative examples:

1. **Random:** any document not cited by d_q.
2. **Negative nearest neighbors:** documents that are close to d_q in the embedding space, but are not cited in it.[6]
3. **Citation-of-citation:** documents referenced in positive citations of d_q, but are not cited directly in d_q.

Negative examples belong to at least one of these types that serve different, and complementary purposes. Selecting a paper from the corpus at random as a negative example typically results in easy negative examples. Selecting nearest neighbor documents in the embedding space used for candidate selection enables the re-ranking phase (described in §4) to fix some of the mistakes made in the candidate selection step. Finally, using citations-of-citations as negative examples is based on the assumption that the authors would have included them as positive examples if they were relevant for the query paper. In Appendix §A, we describe the number of negative examples of each type used for training. Next, we describe how to rerank the candidate documents.

---

[6] Since the set of approximate neighbors depend on model parameters, we recompute a map from each query document to its K nearest neighbors before each epoch while training the document embedding model.
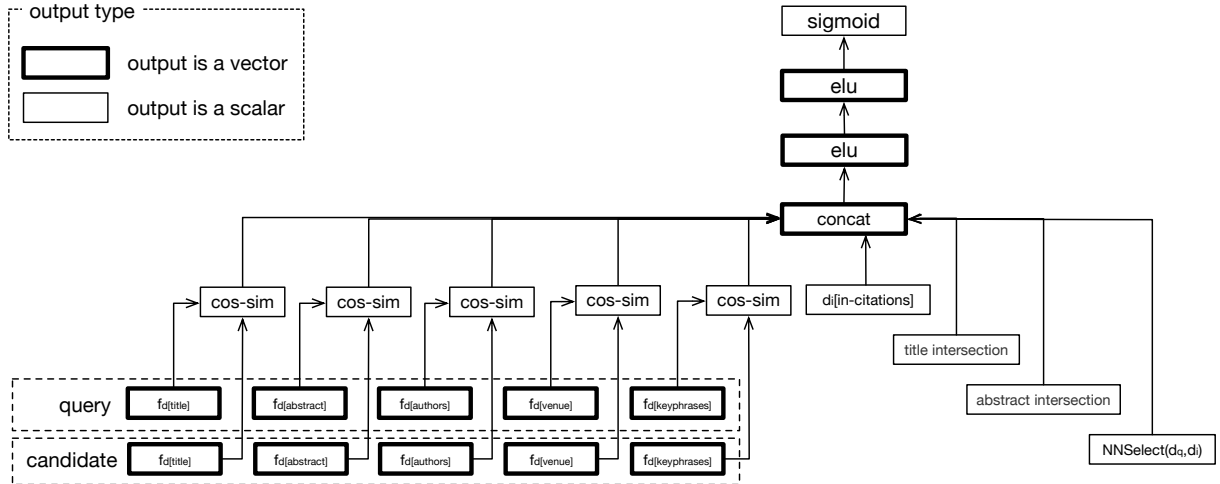
Figure 2: `NNRank` architecture. For each of the textual and categorical fields, we compute the cosine similarity between the embedding for d_q and the corresponding embedding for d_i. Then, we concatenate the cosine similarity scores, the numeric features and the summed weights of the intersection words, followed by two dense layers with ELU non-linearities The output layer is a dense layer with sigmoid non-linearity, which estimates the probability that d_q cites d_i.

4 Phase 2: Reranking Candidates (`NNRank`)

In this phase, we train another model which takes as input a pair of documents (d_q, d_i) and estimates the probability that d_i should be cited in d_q.

Input features. A key point of this work is to assess the feasibility of recommending citations without using metadata, but we describe all features here for completeness and defer this discussion to §5. For each document, we compute dense feature vectors $\mathbf{f}_{d[\text{field}]}$ as defined in Eq. 1 for the following fields: title, abstract, authors, venue and keyphrases (if available). For the title and abstract, we identify the subset of word types which appear in both documents (intersection), and compute the sum of their scalar weights as an additional feature, e.g., $\sum_{t \in \cap \text{title}} w_t^{\cap}$. We also use *log* number of times the candidate document d_i has been cited in the corpus, i.e., $log(d_i[\text{in-citations}])$. Finally, we use the cosine similarity between d_q and d_i in the embedding space, i.e., cos-sim$(\mathbf{e}_{d_q}, \mathbf{e}_{d_i})$.

Model architecture. We illustrate the `NNRank` model architecture in Figure 2.

The output layer is defined as:

$$s(d_i, d_j) = \text{FeedForward}(\mathbf{h}), \qquad (3)$$

$$\mathbf{h} = \Big[\mathbf{g}_{\text{title}}; \mathbf{g}_{\text{abstract}}; \mathbf{g}_{\text{authors}}; \mathbf{g}_{\text{venue}};$$
$$\mathbf{g}_{\text{keyphrases}}; \text{cos-sim}(\mathbf{e}_{d_q}, \mathbf{e}_{d_i});$$
$$\sum_{t \in \cap_{\text{title}}} w_t^{\cap}; \sum_{t \in \cap_{\text{abstract}}} w_t^{\cap};$$
$$d_i[\text{in-citations}] \Big],$$

$$\mathbf{g}_{\text{field}} = \text{cos-sim}(\mathbf{f}_{d_q[\text{field}]}, \mathbf{f}_{d_i[\text{field}]}),$$

where 'FeedForward' is a three layer feed-forward neural network with two exponential linear unit layers (Clevert et al., 2015) and one sigmoid layer. ';' indicates concatenation.

Training. The parameters of the `NNRank` model are $w_*^{\text{mag}}, \mathbf{w}_*^{\text{dir}}, w_*^{\cap}$ and parameters of the three dense layers in 'FeedForward'. We reuse the triplet loss in Eq. 2 to learn these parameters, but redefine the similarity function $s(d_i, d_j)$ as the sigmoid output described in Eq. 3.

At test time, we use this model to recommend candidates d_i with the highest $s(d_q, d_i)$ scores.

5 Experiments

In this section, we describe experimental results of our citation recommendation method and compare it to previous work.

Datasets. We use the DBLP and PubMed datasets (Ren et al., 2014) to compare with previous work on citation recommendation. The DBLP

dataset contains over 50K scientific articles in the computer science domain, with an average of 5 citations per article. The PubMed dataset contains over 45K scientific articles in the medical domains, with an average of 17 citations per article. In both datasets, a document is accompanied by its title, abstract, venue (i.e. journal or conference where the document was published), authors, citations (i.e. other documents in the corpus that are referenced in the given document) and keyphrases (i.e. phrases considered important by automated extraction methods). We replicate the experimental setup of Ren et al. (2014) by excluding papers with fewer than 10 citations and using the standard train, dev and test splits.[7]

We also introduce `OpenCorpus`,[8] a new dataset of 7 million scientific articles primarily drawn from the computer science and neuroscience domain. Due to licensing constraints, documents in the corpus do not include the full text of the scientific articles, but include the title, abstract, year, author, venue, keyphrases and citation information. The mutually exclusive training, development, and test splits were selected such that no document in the development or test set has a publication year less than that of any document in the training set. Papers with zero citations were removed from the development and test sets. We describe the key characteristics of `OpenCorpus` in Table 1.

Statistic	Value
# of documents in corpus	6.9 million
# of unique authors	8.3 million
# of unique keyphrases	823,677
# of unique venues	23,672
avg. # of incoming citations	7.4 ($\pm$ 38.1)
avg. # of outgoing citations	8.4 ($\pm$ 14.4)
size of training set [years 1991 to 2014]	5.5 million
size of dev set [years 2014 to 2015]	689,000
size of test set [years 2015 to 2016]	20,000

Table 1: Characteristics of the `OpenCorpus`.

Baselines. We compare our method to two baseline methods for recommending citations: ClusCite and BM25. ClusCite (Ren et al., 2014) clusters nodes in a heterogeneous graph of terms, authors and venues in order to find related documents which should be cited. We use the ClusCite results

as reported in Ren et al. (2014), which compared it to several other citation recommendation methods and found that it obtains state of the art results on the PubMed and DBLP datasets. The BM25 results are based on our implementation of the popular ranking function Okapi BM25 used in many information retrieval systems. See Appendix §D for details of our BM25 implementation.

Evaluation. We use Mean Reciprocal Rank (MRR) and F1@20 to report the main results in this section. In Appendix §F, we also report additional metrics (e.g., precision and recall at 20) which have been used in previous work. We compute F1@20 as the harmonic mean of the corpus-level precision and recall at 20 (P@20 and R@20). Following (Ren et al., 2014), precision and recall at 20 are first computed for each query document then averaged over query documents in the test set to compute the corpus-level P@20 and R@20.

Configurations. To find candidates in `NNSelect`, we use the approximate nearest neighbor search algorithm `Annoy`[9], which builds a binary-tree structure that enables searching for nearest neighbors in $O(\log n)$ time. To build this tree, points in a high-dimensional space are split by choosing random hyperplanes. We use 100 trees in our approximate nearest neighbors index, and retrieve documents using the cosine distance metric.

We use the `hyperopt` library[10] to optimize various hyperparameters of our method such as size of hidden layers, regularization strength and learning rate. To ensure reproducibility, we provide a detailed description of the parameters used in both `NNSelect` and `NNRank` models, our hyperparameter optimization method and parameter values chosen in Appendix §A.

Main results. Table 2 reports the F1@20 and MRR results for the two baselines and three variants of our method. Since the `OpenCorpus` dataset is much bigger, we were not able to train the ClusCite baseline for it. Totti et al. (2016) have also found it difficult to scale up ClusCite to larger datasets. Where available, we report the mean $\pm$ standard deviation based on five trials.

The first variant, labeled "`NNSelect`," only uses the candidate selection part of our method (i.e., phase 1) to rank candidates by their cosine

---

[7] The dataset characteristics reported here are different from those in Table 3 in (Ren et al., 2014) because we report the size of the filtered datasets while they report the size of the datasets before filtering.
[8] http://labs.semanticscholar.org/corpus/

[9] https://github.com/spotify/annoy
[10] https://github.com/hyperopt/hyperopt

Method	DBLP		PubMed		OpenCorpus	
	F1@20	MRR	F1@20	MRR	F1@20	MRR
BM25	0.119	0.425	0.209	0.574	0.058	0.218
ClusCite	0.237	0.548	0.274	0.578	–	–
NNSelect	0.282±0.002	0.579±0.007	0.309±0.001	0.699±0.001	0.109	0.221
+ NNRank	0.302±0.001	0.672±0.015	0.325±0.001	0.754±0.003	**0.126**	0.330
+ metadata	**0.303**±0.001	**0.689**±0.011	**0.329**±0.001	**0.771**±0.003	0.125	**0.330**

Table 2: F1@20 and MRR results for two baselines and three variants of our method. BM25 results are based on our implementation of this baseline, while ClusCite results are based on the results reported in Ren et al. (2014). "NNSelect" ranks candidates using cosine similarity between the query and candidate documents in the embedding space (phase 1). "NNSelect + NNRank" uses the discriminative reranking model to rerank candidates (phase 2), without encoding any of the metadata features. "+ metadata" encodes the metadata features (i.e., keyphrases, venues and authors), achieving the best results on all datasets. Mean and standard deviations are reported based on five trials.

similarity to the query document in the embedding space as illustrated in Fig. 1. Although the document embedding space was designed to efficiently select candidates for further processing in phase 2, recommending citations directly based on the cosine distance in this space outperforms both baselines.

The second variant, labeled "NNSelect + NNRank," uses the discriminative model (i.e., phase 2) to rerank candidates selected by NNSelect, without encoding metadata (venues, authors, keyphrases). Both the first and second variants show that improved modeling of paper text can significantly outperform previous methods for citation recommendation, without using metadata.

The third variant, labeled "NNSelect + NNRank + metadata," further encodes the metadata features in the reranking model, and gives the best overall results. On both the DBLP and PubMed datasets, we obtain relative improvements over 20% (for F1@20) and 25% (for MRR) compared to the best reported results of ClusCite.

In the rest of this section, we describe controlled experiments aimed at analyzing different aspects of our proposed method.

Choice of negative samples. As discussed in §3, we use different types of negative samples to train our models. We experimented with using only a subset of the types, while controlling for the total number of negative samples used, and found that using negative nearest neighbors while training the models is particularly important for the method to work. As illustrated in Table 3, on the PubMed dataset, adding negative nearest neighbors while training the models improves the F1@20 score

from 0.306 to 0.329, and improves the MRR score from 0.705 to 0.771. Intuitively, using nearest neighbor negative examples focuses training on the harder cases on which the model is more likely to make mistakes.

	F1@20	Δ	MRR	Δ
Full model	0.329		0.771	
without intersection	0.296	**0.033**	0.653	**0.118**
without -ve NNs	0.306	0.016	0.705	0.066
without numerical	0.314	0.008	0.735	0.036

Table 3: Comparison of PubMed results of the full model with model without (i) intersection features, (ii) negative nearest neighbors in training samples, and (iii) numerical features.

Valuable features. We experimented with different subsets of the optional features used in NNRank in order to evaluate the contribution of various features. We found intersection features, NNSelect scores, and the number of incoming citations to be the most valuable feature. As illustrated in Table 3, the intersection features improves the F1@20 score from 0.296 to 0.329, and the MRR score from 0.653 to 0.771, on the PubMed dataset. The numerical features (NNSelect score and incoming citations) improve the F1@20 score from 0.314 to 0.329, and improves the MRR score from 0.735 to 0.771. This shows that, in some applications, feeding engineered features to neural networks can be an effective strategy to improve their performance.

Performance across venues We studied the variability of performance of our model for papers from different venues. Figure 3 shows the F1@20 score of NNRank for papers belonging to the top
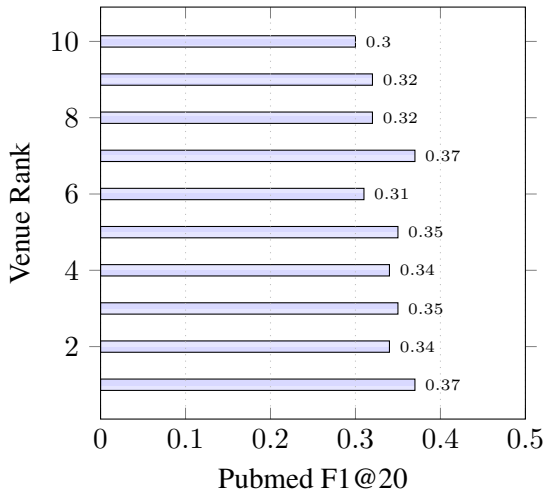
Figure 3: F1@20 of `NNRank` on the Pubmed dataset across top ten venues

# of neighbors	R@20	P@20	Time(ms)
1	0.123	0.079	131
5	**0.142**	**0.080**	144
10	0.138	0.069	200
50	0.081	0.040	362

Table 4: `OpenCorpus` results for `NNSelect` step with varying number of nearest neighbors on 1,000 validation documents.

ten venues (by their paper count) in the Pubmed corpus. `NNRank`'s performance is robust across venues.

Encoding textual features. We also experimented with using recurrent and convolutional neural network to encode the textual fields of query and candidate documents, instead of using a weighted sum as described in Eq. 1. We found that recurrent and convolutional encoders are much slower, and did not observe a significant improvement in the overall performance as measured by the F1@20 and MRR metrics. This result is consistent with previous studies on other tasks, e.g., Iyyer et al. (2015).

Number of nearest neighbors. As discussed in §3, the candidate selection step is crucial for the scalability of our method because it reduces the number of computationally expensive pairwise comparisons with the query document at runtime. We did a controlled experiment on the `OpenCorpus` dataset (largest among the three datasets) to measure the effect of using different numbers of nearest neighbors, and found that both P@20 and R@20 metrics are maximized when `NNSelect` fetches five nearest neighbors using the approximate nearest neighbors index (and their out-going citations), as illustrated in Table 4.

Self-citation bias. We hypothesized that a model trained with the metadata (e.g., authors) could be biased towards self-citations and other well-cited authors. To verify this hypothesis, we compared two `NNRank` models – one with meta-

data, and one without. We measured the mean and max rank of predictions that had at least one author in common with the query document. This experiment was performed with the `OpenCorpus` dataset.

A lower mean rank for `NNRank` + Metadata indicates that the model trained with metadata tends to favor documents authored by one of the query document's authors. We verified the prevalence of this bias by varying the number of predictions for each model from 1 to 100. Figure 4 shows that the mean and max rank of the model trained with metadata is always lower than those for the model that does not use metadata.
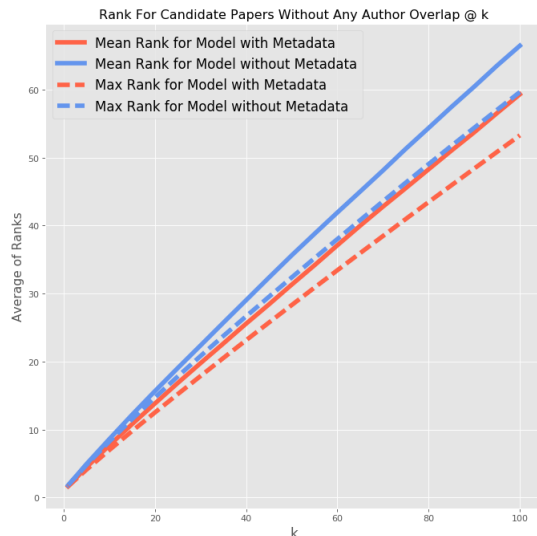

Figure 4: Mean and Max Rank of predictions with varying number of candidates.

6 Related Work

Citation recommendation systems can be divided into two categories – *local* and *global*. A local citation recommendation system takes a few sentences (and an optional placeholder for the candidate citation) as input and recommends citations based on the local context of the input sen-

244

tences (Huang et al., 2015; He et al., 2010; Tang and Zhang, 2009; Huang et al., 2012; He et al., 2011). A global citation recommendation system takes the entire scholarly article as input and recommends citations for the paper (McNee et al., 2002; Strohman et al., 2007; Nallapati et al., 2008; Kataria et al., 2010; Ren et al., 2014). We address the global citation recommendation problem in this paper.

A key difference of our proposed method compared to previous work is that our method is content-based and works well even in the absence of metadata (e.g. authors, venues, key phrases, seed list of citations). Many citation recommendation systems crucially rely on a query document's metadata. For example, the collaborative filtering based algorithms of McNee et al. (2002); Jia and Saule (2017); Liu et al. (2015) require seed citations for a query document. (Ren et al., 2014; Yu et al., 2012) require authors, venues and key terms of the query documents to infer interest groups and to extract features based on paths in a heterogeneous graph. In contrast, our model performs well solely based on the textual content of the query document.

Some previous work (e.g. (Ren et al., 2014; Yu et al., 2012)) have addressed the citation recommendation problem using graph-based methods. But, training graph-based citation recommendation models has been found to be expensive. For example, the training complexity of the ClusCite algorithm (Ren et al., 2014) is cubic in the number of edges in the graph of authors, venues and terms. This can be prohibitively expensive for datasets as large as `OpenCorpus`. On the other hand our model is a neural network trained via batched stochastic gradient descent that scales very well to large datasets (Bottou, 2010).

Another crucial difference between our approach and some prior work in citation prediction is that we build up a document representation using its constituent words only. Prior algorithms (Huang et al., 2015, 2012; Nallapati et al., 2008; Tanner and Charniak, 2015) learn an explicit representation for each training document separately that isn't a deterministic function of the document's words. This makes the model effectively transductive since a never-before-seen document does not have a ready-made representation. Similarly, Huang et al. (2012)'s method needs a candidate document to have at least one in-coming cita-

tion to be eligible for citation – this disadvantages newly published documents. Liu et al. (2015) form document representations using citation relations, which are not available for unfinished or new documents. In contrast, our method does not need to be re-trained as the corpus of potential candidates grows. As long as the new documents are in the same domain as that of the model's training documents, they can simply be added to the corpus and are immediately available as candidates for future queries.

While the citation recommendation task has attracted a lot of research interest, a recent survey paper (Beel et al., 2016) has found three main concerns with existing work: (i) limitations in evaluation due to strongly pruned datasets, (ii) lack of details for re-implementation, and (iii) variations in performance across datasets. For example, the average number of citations per document in the DBLP dataset is 5, but Ren et al. (2014) filtered out documents with fewer than 10 citations from the test set. This drastically reduced the size of the test set. We address these concerns by releasing a new large scale dataset for future citation recommendation systems. In our experiments on the `OpenCorpus` dataset, we only prune documents with zero outgoing citations. We provide extensive details of our system (see Appendix §A) to facilitate reproducibility and release our code[11]. We also show in experiments that our method consistently outperforms previous systems on multiple datasets.

Finally, recent work has combined graph node representations and text-based document representations using CCA (Gupta and Varma, 2017). This sort of approach can enhance our text-based document representations if a technique to create graph node representations at test-time is available.

7 Conclusion

In this paper, we present a content-based citation recommendation method which remains robust when metadata is missing for query documents, enabling researchers to do an effective literature search early in their research cycle or during the peer review process, among other scenarios. We show that our method obtains state of the art results on two citation recommendation datasets, even without the use of metadata available to the

---

[11] https://github.com/allenai/citeomatic

baseline method. We make our system publicly accessible online. We also introduce a new dataset of seven million scientific articles to facilitate future research on this problem.

Acknowledgements

References

Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. 2016. Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries* 17(4):305–338.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, Springer, pages 177–186.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence* 7(04):669–688.

Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014a. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1435–1446.

Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014b. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *EMNLP*. volume 14, pages 1435–1446.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* .

Shashank Gupta and Vasudeva Varma. 2017. Scientific article recommendation by using distributed representations of text and graph. In *WWW*.

Qi He, Daniel Kifer, Jian Pei, Prasenjit Mitra, and C. Lee Giles. 2011. Citation recommendation without author supervision. In *WSDM*.

Qi He, Jian Pei, Daniel Kifer, Prasenjit Mitra, and Lee Giles. 2010. Context-aware citation recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, pages 421–430.

Wenyi Huang, Saurabh Kataria, Cornelia Caragea, Prasenjit Mitra, C Lee Giles, and Lior Rokach. 2012. Recommending citations: translating papers into references. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, pages 1910–1914.

Wenyi Huang, Zhaohui Wu, Liang Chen, Prasenjit Mitra, and C Lee Giles. 2015. A neural probabilistic model for context based citation recommendation. In *AAAI*. pages 2404–2410.

Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daumé. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

Haofeng Jia and Erik Saule. 2017. An analysis of citation recommender systems: Beyond the obvious. In *ASONAM*.

Saurabh Kataria, Prasenjit Mitra, and Sumit Bhatia. 2010. Utilizing context in generative bayesian models for linked corpus. In *AAAI*.

Haifeng Liu, Xiangjie Kong, Xiaomei Bai, Wei Wang, Teshome Megersa Bekele, and Feng Xia. 2015. Context-based collaborative filtering for citation recommendation. *IEEE Access* 3:1695–1703.

Patrice Lopez and Laurent Romary. 2010. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, pages 248–251.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL*.

Sean M McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K Lam, Al Mamunur Rashid, Joseph A Konstan, and John Riedl. 2002. On the recommending of citations for research papers. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*. ACM, pages 116–125.

Ramesh Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. 2008. Joint latent topic models for text and citations. In *KDD*.

Xiang Ren, Jialu Liu, Xiao Yu, Urvashi Khandelwal, Quanquan Gu, Lidan Wang, and Jiawei Han. 2014. Cluscite: Effective citation recommendation by information network-based clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 821–830.

Tim Salimans and Diederik P. Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *CoRR* abs/1602.07868. http://arxiv.org/abs/1602.07868.

Trevor Strohman, W. Bruce Croft, and David D. Jensen. 2007. Recommending citations for academic papers. In *SIGIR*.

Jie Tang and Jing Zhang. 2009. A discriminative approach to topic-based citation recommendation. In *PAKDD*.

Chris Tanner and Eugene Charniak. 2015. A hybrid generative/discriminative approach to citation prediction. In *HLT-NAACL*.

Luam C. Totti, Prasenjit Mitra, Mourad Ouzzani, and Mohammed J. Zaki. 2016. A query-oriented approach for relevance in citation networks. In *WWW*.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity with deep ranking. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Washington, DC, USA, CVPR '14, pages 1386–1393. https://doi.org/10.1109/CVPR.2014.180.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina*. pages 2111–2117.

Xiao Yu, Quanquan Gu, Mianwei Zhou, and Jiawei Han. 2012. Citation prediction in heterogeneous bibliographic networks. In *SDM*.

A Hyperparameter Settings

Neural networks are complex and have a large number of hyperparameters to tune. This makes it challenging to reproduce experimental results. Here, we provide details of how the hyperparameters of the `NNSelect` and `NNRank` models were chosen or otherwise set. We chose a subset of hyperparameters for tuning, and left the rest at manually set default values. Due to limited computational resources, we were only able to perform hyperparameter tuning on the development split of the smaller DBLP and Pubmed datasets.

For DBLP and PubMed, we first ran Hyperopt[12] with 75 trials. Each trial was run for five epochs of 500,000 triplets each. The ten top performing of these models were trained for a full 50 epochs, and the best performing model's hyperparameters are selected. Hyperparameters for `NNSelect` were optimized for Recall@20 and those for the `NNRank` model were optimized for F1@20 on the development set. The selected values for DBLP are reported in Table 6 and for PubMed are reported in Table 7.

`OpenCorpus` hyperparameters were set via informal hand-tuning, and the results are in Table 9. A few miscellaneous parameters (not tuned) that are necessary for reproducibility are in Table 8.

We briefly clarify the meaning of some parameters below:

- **Margin Multiplier** - The triplet loss has variable margins for the three types of negatives: $0.1\gamma, 0.2\gamma$, and 0.3γ. We treat γ as a hyperparameter and refer to it as the margin multiplier.

- **Use Siamese Embeddings** - For the majority of our experiments, we use a *Siamese* model (Bromley et al., 1993). That is, the textual embeddings for the query text and abstract share the same weights. However, we had a significantly larger amount of data to train `NNRank` on `OpenCorpus`, and found that non-Siamese embeddings are beneficial.

- **Use Pretrained** - We estimate word embeddings on the titles and abstracts of `OpenCorpus` using Word2Vec implemented by the gensim Python package[13].

---

[12]https://github.com/hyperopt/hyperopt
[13]https://radimrehurek.com/gensim/

B Margin Loss Details

When computing the margins for the triplet loss, we use a boosting function for highly cited documents. The full triplet loss function is as follows:

$$\max\big(\gamma\alpha(d^-) \\ + s(d_q, d^-) + B(d^-) \\ - s(d_q, d^+) - B(d^+) \\ , 0\big)$$

where γ is the margin multiplier, and $\alpha(d^-)$ varies based on the type of negative document:

- $\alpha(d^-) = 0.3$ for random negatives

- $\alpha(d^-) = 0.2$ for nearest neighbor negatives

- $\alpha(d^-) = 0.1$ for citation-of-citation negatives.

The boosting function is defined as follows:

$$B(d) = \frac{\sigma\left(\frac{d[\text{in-citations}]}{100}\right)}{50}$$

where σ is the sigmoid function and $d[\text{in-citations}]$ is the number of times document d was cited in the corpus. The boosting function allows the model to slightly prefer candidates that are cited more frequently, and the constants were set without optimization.

C Nearest Neighbors for Training Details

When obtaining nearest neighbors for negative examples during training, we use a heuristic to find a subset of the fetched nearest neighbors that are sufficiently wrong. That is, these are non-citation samples that look dissimilar in the original text but similar in the embedding space. This procedure is as follows for each training query:

1. Compute the Jaccard similarities between a training query and all of its true citations using the concatenation of title and abstract texts.

2. Compute the bottom fifth percentile Jaccard similarity value. I.e. the value below which only the bottom 5% most least textually similar true citations fall. For example, if the Jaccard similarities range from 0.2 to 0.9, the fifth percentile might plausibly be 0.3.

3. Use the Annoy index computed at the end of the previous epoch to fetch nearest neighbors for the query document.

4. Compute the textual Jaccard similarity between all of the nearest neighbors and the query document.

5. Retain nearest neighbors that have a smaller Jaccard similarity than the fifth percentile. Using the previous example, retain the nearest neighbors that have a lower Jaccard similarity than 0.3.

D   BM25 Details

BM25 Implementation	DBLP		PubMed	
	F@20	MRR	F@20	MRR
Ren et al. (2014)	0.111	0.411	0.153	0.497
Our approximation	0.119	0.425	0.209	0.574

Table 5: Results of our BM25 implementation on DBLP and Pubmed datasets.

Okapi-BM25 is a popular ranking function. We use BM25 as an IR-based baseline for the task of citation recommendation. For the DBLP and Pubmed datasets, BM25 performance is provided in Ren et al. (2014). To create a competitive BM25 baseline for `OpenCorpus`, we first created indexes for the DBLP and Pubmed datasets and tuned the query to approximate the performance reported in previous work. We used Whoosh[14] to create an index. We extract the key terms (using Whoosh's `key_terms_from_text method`) from the title and abstract of each query document. The key terms from the document are concatenated to form the query string. Table 5 shows that our BM25 is a close approximation to the BM25 implementation of previous work and can be reliably used as a strong IR baseline for `OpenCorpus`. In Table 2, we report results on all three datasets using our BM25 implementation.

E   Key Phrases for `OpenCorpus`

In the `OpenCorpus` dataset, some documents are accompanied by automatically extracted key phrases. Our implementation of automatic key phrase extraction is based on standard key phrase extraction systems – e.g. (Caragea et al., 2014a,b; Lopez and Romary, 2010). We first extract noun phrases using the Stanford CoreNLP package (Manning et al., 2014) as candidate key phrases. Next, we extract corpus level and document level features (e.g. term frequency, document frequency, n-gram probability etc.) for each candidate key phrase. Finally, we rank the candidate key phrases using a ranking model that is trained on author-provided key phrases as gold labels.

F   Detailed Experimental Results

Table 10 compares `NNRank` with previous work in detail on DBLP and Pubmed datasets. Clus-Cite (Ren et al., 2014) clusters nodes in a heterogeneous graph of terms, authors and venues in order to find related documents which should be cited. ClusCite obtains the previous best results on these two datasets. L2-LR (Yu et al., 2012) uses a linear combination of meta-path based linear features to classify candidate citations. We show that `NNRank` (with and without metadata) consistently outperforms ClusCite and other baselines on all metrics on both datasets.

---

[14]https://pypi.python.org/pypi/Whoosh/

	NNSelect		NNRank	
Hyperparameter	**Range**	**Chosen Value**	**Range**	**Chosen Value**
learning rate	[1e-5, 1e-4, ..., 1e-1]	0.01	[1e-5, 1e-4, ..., 1e-1]	0.01
l2 regularization	[0, 1e-7, 1e-6, ..., 1e-2]	0	[0, 1e-7, 1e-6, ..., 1e-2]	1e-3
l1 regularization	[0, 1e-7, 1e-6, ..., 1e-2]	1e-7	[0, 1e-7, 1e-6, ..., 1e-2]	1e-4
word dropout	[0, 0.05, 0.1, ..., 0.75]	0.60	[0, 0.05, 0.1, ..., 0.75]	0.35
margin multiplier	[0.5, 0.75, 1.0, 1.25, 1.5]	1.0	[0.5, 0.75, 1.0, 1.25, 1.5]	0.5
dense dimension	[25, 50, ..., 325]	300	[25, 50, ..., 325]	175
metadata dimension	-	-	[5, 10, ..., 55]	45
use pretrained	[true, false]	true	[true, false]	false
finetune pretrained	[true, false]	true	[true, false]	-
number ANN neighbors	-	10	-	-
triplets per batch size	-	256	-	256
triplets per epoch	-	500000	-	500000
triplets per training	-	2500000	-	2500000
use Siamese embeddings	-	true	-	true

Table 6: DBLP hyperparameter tuning results. Note that the dense dimension when using pretrained vectors is fixed to be 300. A '-' indicates that the variable was not tuned.

	NNSelect		NNRank	
Hyperparameter	**Range**	**Chosen Value**	**Range**	**Chosen Value**
learning rate	[1e-5, 1e-4, ..., 1e-1]	0.001	[1e-5, 1e-4, ..., 1e-1]	0.001
l2 regularization	[0, 1e-7, 1e-6, ..., 1e-2]	0	[0, 1e-7, 1e-6, ..., 1e-2]	0
l1 regularization	[0, 1e-7, 1e-6, ..., 1e-2]	1e-6	[0, 1e-7, 1e-6, ..., 1e-2]	1e-6
word dropout	[0, 0.05, 0.1, ..., 0.75]	0.55	[0, 0.05, 0.1, ..., 0.75]	0.1
margin multiplier	[0.5, 0.75, 1.0, 1.25, 1.5]	0.5	[0.5, 0.75, 1.0, 1.25, 1.5]	1.5
dense dimension	[25, 50, ..., 325]	325	[25, 50, ..., 325]	150
metadata dimension	-	-	[5, 10, ..., 55]	40
use pretrained	[true, false]	false	[true, false]	false
finetune pretrained	[true, false]	-	[true, false]	-
number ANN neighbors	-	10	-	-
triplets per batch size	-	256	-	256
triplets per epoch	-	500000	-	500000
triplets per training	-	2500000	-	2500000
use Siamese embeddings	-	true	-	true

Table 7: PubMed hyperparameter tuning results. Note that the dense dimension when using pretrained GloVe vectors is fixed to be 300. A '-' indicates that the variable was not tuned.

Hyperparameter	PubMed/DBLP Value	OpenCorpus Value
title/abstract vocabulary size	200000	200000
maximum title length	50	50
maximum abstract length	500	500
training triplets per query	6	6
min # of papers per author included	1	10
min # of papers per venue included	1	10
min # of papers per keyphrases included	5	10
max authors per document	8	8
max keyphrases per document	20	20
minimum true citations per document	2/1	2
maximum true citations per document	100	100
optimizer	LazyAdamOptimizer*	Nadam**
use magnitude-direction embeddings	true	true
reduce learning rate upon plateau	false	true

Table 8: Per-dataset parameters. These were hand-specified. *LazyAdamOptimizer is part of TensorFlow. **Nadam is part of Keras.

Hyperparameter	NNSelect Value	NNRank Value
learning rate	0.001	0.001
l2 regularization	1e-5	1e-5
l1 regularization	1e-7	1e-7
word dropout	0.1	0.1
margin multiplier	1.0	1.0
dense dimension	75	75
metadata dimension	-	25
use pretrained	false	false
number ANN neighbors	5	-
triplets per batch size	256	32
triplets per epoch	2500000	2500000
triplets per training	25000000	100000000
use Siamese embeddings	true	false

Table 9: Hyperparameters used for OpenCorpus

Method	DBLP					PubMed				
	P@10	P@20	R@20	F1@20	MRR	P@10	P@20	R@20	F1@20	MRR
BM25	0.126	0.0902	0.1431	0.11	0.4107	0.1847	0.1349	0.1754	0.15	0.4971
L2-LR	0.2274	0.1677	0.2471	0.200	0.4866	0.2527	0.1959	0.2504	0.2200	0.5308
ClusCite	0.2429	0.1958	0.2993	0.237	0.5481	0.3019	0.2434	0.3129	0.274	0.5787
NNSelect	0.287	0.230	0.363	0.282	0.579	0.388	0.316	0.302	0.309	0.699
+ NNRank	0.339	0.247	0.390	0.302	0.672	0.421	0.332	0.318	0.325	0.754
+ metadata	**0.345**	**0.247**	**0.390**	**0.303**	**0.689**	**0.429**	**0.337**	**0.322**	**0.329**	**0.771**

Table 10: Comparing NNRank with ClusCite. (Ren et al., 2014) have presented results on several other topic-based, link-based and network-based citation recommendation methods as baselines. For succinctness, we show results for the best system, Cluscite, and two baselines BM25 and L2-LR.