# Cleaning Noisy and Heterogeneous Metadata for Record Linking across Scholarly Big Datasets

**Athar Sefid,[1] Jian Wu,[3] Allen C. Ge,[2] Jing Zhao,[2] Lu Liu,[2] Cornelia Caragea,[4] Prasenjit Mitra,[2] C. Lee Giles[1,2]**
[1]Computer Science and Engineering, Pennsylvania State University, University Park, PA, 16801
[2]Information Sciences and Technology, Pennsylvania State University, University Park, PA, 16801
[3]Computer Science, Old Dominion University, Norfolk, VA, 23529
[4]Computer Science, University of Illinois at Chicago, Chicago, IL, 60607
azs5955@psu.edu, jwu@cs.odu.edu

## Abstract

Automatically extracted metadata from scholarly documents in PDF formats is usually noisy and heterogeneous, often containing incomplete fields and erroneous values. One common way of cleaning metadata is to use a bibliographic reference dataset. The challenge is to match records between corpora with high precision. The existing solution which is based on information retrieval and string similarity on titles works well only if the titles are cleaned. We introduce a system designed to match scholarly document entities with noisy metadata against a reference dataset. The blocking function uses the classic BM25 algorithm to find the matching candidates from the reference data that has been indexed by ElasticSearch. The core components use supervised methods which combine features extracted from all available metadata fields. The system also leverages available citation information to match entities. The combination of metadata and citation achieves high accuracy that significantly outperforms the baseline method on the same test dataset. We apply this system to match the database of CiteSeerX against Web of Science, PubMed, and DBLP. This method will be deployed in the CiteSeerX system to clean metadata and link records to other scholarly big datasets.

## Introduction

Since the advent of Scholarly Big Data (SBD)(Giles 2013), there has been a growing interest in topics related to this big data instance, such as scholarly article discovery (Wesley-Smith and West 2016), semantic analysis (Al-Zaidy and Giles 2017), recommendation systems (Huang et al. 2015), citation prediction (Liu et al. 2017), scalability improvement (Kim, Sefid, and Giles 2017), and Science of Science (Chen et al. 2018). Major SBD datasets include the Microsoft Academic Graph (MAG), CiteSeerX (Giles, Bollacker, and Lawrence 1998; Wu et al. 2014), DBLP, Web of Science (WoS), and Medline, among which MAG and CiteSeerX are the only two freely available large scale datasets which offer citation graphs. Academic search engines such as Microsoft Academic and CiteSeerX obtain raw PDF files by actively crawling the Web. These PDF documents are then classified into academic and non-academic documents. Metadata, citations, and other types of content are then extracted from these documents. Different from submission-based datasets such as the WoS, a large fraction of documents crawled are pre-prints and manuscripts, which do not necessarily contain unique identifiers, e.g., DOIs. Metadata from these documents may also differ from their official published versions. Also, errors may occur when text is extracted from PDFs, and when metadata is parsed. As a result, metadata extracted is likely to be incomplete and erroneous. This metadata is also heterogeneous since the documents were written by authors using different conventions and templates. This noisy information can propagate through to data analytics and aggregations that can then distort research, making cleaning it a necessity. One common approach is to link document entities to corresponding entities in a cleaned dataset (reference dataset) and then use its records to replace the automatically extracted records.

The biggest challenge of this approach is to find the correct matching entity in the reference dataset. Though different from traditional machine learning (ML) tasks in which there is a trade-off between precision and recalls, the entity matching task must be accomplished with *high precision*. This is because when the dataset is large, a small fraction of false positives may lead to a large number of "false corrections". The problem is even more challenging because there is usually no prior knowledge of which fields are noisy.

Our contributions include: (1) developing a ML-based paper entity matching framework which includes both header and citation information of available scholarly documents; (2) applying the system on CiteSeerX, WoS, DBLP, and PubMed to find overlap between these digital library databases which is used to clean CiteSeerX data. Plus this generates a more accurate citation graph data and links records which can enrich the content of individual document.

## Related Work

There are in general three types of methods in entity matching across bibliographic databases.

**Information Retrieval-based**: This method searches one or multiple attributes of an entity in the target corpus against the index of the reference corpus and rank the candidates using a similarity metric. This approach matches CiteSeerX with DBLP (Caragea et al. 2014). The reference dataset (DBLP) was indexed by Apache Solr. Metadata from the noisy dataset (CiteSeerX) were used to query corresponding fields. Candidates were selected based on similarity scores. It was found that using 3-grams of titles and Jaccard similarity with a threshold of $0.7$ achieves the best F1-measure of $77\%$. Because of the relatively low precision, the approach cannot be directly used in cleaning CiteSeerX data.

**Machine Learning-based**: These methods have been used in entity matching of user entities in online social networks (Peled et al. 2013). The problem is to match user profiles on Facebook and Xing. This work applies pairwise comparison to whole dataset ($\approx 15,000$ records) without applying any blocking function to reduce search spaces, so this method cannot be scaled up to large digital libraries containing tens of millions of records.

**Topical-based**: This method is used to resolve and match entities that are represented by free text, e.g., Wiki articles. The challenge is that different sources may use different languages or terminologies to describe the same topic. A probabilistic model was proposed to integrate the topic extraction and matching into a unified model (Yang et al. 2015). As we don't have access to the full text of the reference datasets, this method is not applicable to our problem.

## Models

### Entity Representation

Throughout below, we refer to a scholarly paper with full information as a *paper entity*. We denote the target corpus which contains noisy data as $T$. This contains $n$ paper entities $t_i, 1 \leq i \leq n$, and a reference corpus $R$ which contains reference data and $m$ paper entities $r_j, 1 \leq j \leq m$. Each entity can be represented by a number of attributes. Our goal is to find a set $M$,

$$M = \{(t, r); t = r, t \in T, r \in R\}.$$

An officially published paper usually is assigned a unique identifier, i.e. DOI. A journal article can also be identified by the journal name, volume, issue number, and the starting page number. However, for a large fraction of open access scholarly papers crawled from the Web, such information is usually not available. Empirically, a paper entity can be uniquely identified by four header fields, (title, authors, year, venue), in which the venue is a conference or a journal name. A citation record parsed from a citation string usually contains the above four fields. So, matching a single citation record can be done in the same manner as matching a paper entity. The abstract is usually a paragraph of text that may contain non-ASCII characters with different encodings, but normalizing abstracts and calculating simhash values takes heavy overhead, so in this work, we use abstracts without normalization. Due to a lack of general venue disambiguator, venue information is not incorporated as matching features. We will show that even without it, the application still achieves high performance.

Paper entity linking can be formalized as a binary classification problem in which the classifier decides whether a candidate pair is a real match or not. Because many digital library datasets do not have citation information, we consider two separate models, one with only header information called Header Matching Model (HMM), and the other with only citation information called Citation Matching Model (CMM). The combination of them is called the Integrated Matching Model (IMM). There is a separate model to evaluate title quality called Title Evaluation Model (TEM).

### Header Matching Model (HMM)

**HMM** is a supervised machine learning model to classify candidate pairs using information from the paper header. We first index all document metadata in the reference corpus using an open source search platform. We use ElasticSearch (ES) because of its relatively low setup overhead and scalability. The default settings are applied for our experiments. The indexed metadata contains header fields including titles, authors, abstracts, and years. For each paper in the target corpus, we query the title against the index, if it contains a minimum of 20 characters. Otherwise, the first author's last name and publication year are used in queries. If the year is not available, only the first author's last name is used. For each field, the query string is segmented into unigrams connected by "OR". The ranking scores are calculated using the Okapi BM25 algorithm (Robertson, Zaragoza, and Taylor 2004). The query algorithm is shown in Algorithm 1.

For each target paper, the top 10 papers from the reference set are retrieved and 10 candidates are formed as matching pairs. The features of each pair include a list of similarities calculated using the header metadata.

1. Title similarity represented by Levenshtein distance of simhashes of normalized titles (Charikar 2002):

$$\text{Sim}_L(\text{title}_t, \text{title}_r) = \text{lev}_{a,b}(|a|, |b|) \tag{1}$$

$$a = \text{Simhash}_{16}(\text{Norm}(\text{title}_t)) \tag{2}$$

$$b = \text{Simhash}_{16}(\text{Norm}(\text{title}_r)) \tag{3}$$

   in which a simhash string contains 16 alphanumeric characters. The titles are normalized so that (1) all letters are lowercased; (2) diacritics are removed, e.g., "á" is converted to "a"; (3) consecutive spaces are collapsed; (4) punctuation marks are trimmed off; (5) single characters "s" and "t" are removed because they are mostly resulted from removing apostrophe from possessives or abbreviations such as "can't".

2. Abstract similarity $\text{Sim}_L(\text{abstract}_t, \text{abstract}_r)$ represented by Levenshtein distance of simhashes of abstracts, calculated in a similar way as Equations (1)–(3) without normalization.

3. Jaccard similarities between normalized titles and original abstracts. For example,

$$\text{Sim}_J(\text{title}_t, \text{title}_r) = \frac{|W_t \cap W_r|}{|W_t \cup W_r|} \tag{4}$$

   in which $W_t$ and $W_r$ represent the token set of the title of the target and the reference paper, respectively.

4. The absolute difference of the years.

5. The **first and the last author's full name similarity**. Author similarities are measured in multiple metrics. An author's full name similarity is represented by a three digit binary $lmf$, representing whether the last name, the middle initial, and the first initial matches, respectively. If a certain name component is missing or it does not match, the binary is set to 0. The decimal value of a binary is used as the full name similarity index. Author names are also normalized before comparison. Diacritics are removed and letters are lowercased. Prefixes, e.g., Prof., and suffixes, e.g., "II", and their variants are removed. For example, if first authors are "Jane C. Huck" and "J. Huck", the binary is 101, which equals to 5 in decimal.
6. The **last name similarities of the first and the last author.** The last name similarity is computed in this way

$$\mathrm{Sim}(\mathrm{N}_t, \mathrm{N}_r) = \begin{cases} 0: \mathrm{N}_t \neq \mathrm{NULL} \wedge \mathrm{N}_r \neq \mathrm{NULL} \wedge \mathrm{N}_t \neq \mathrm{N}_r \\ 1: \mathrm{N}_t = \mathrm{NULL} \vee \mathrm{N}_r = \mathrm{NULL} \\ 2: \mathrm{N}_t \neq \mathrm{NULL} \wedge \mathrm{N}_r \neq \mathrm{NULL} \wedge \mathrm{N}_t = \mathrm{N}_r \end{cases} \quad (5)$$

   in which N stands for the last name and NULL means the value is not available.
7. **All authors' last name Jaccard similarity**

$$\mathrm{Sim}_J\left(L_t, L_r\right) = \frac{|L_t \cap L_r|}{|L_t \cup L_r|} \quad (6)$$

   in which $L_t$ and $L_r$ stands for the set of last names in the target and the reference corpora, respectively.

The pseudocodes of the HMM is in Algorithm 2.

**Algorithm 1** Query Builder

```
1: function QUERY($title, lastName, year$)
2:     if title $\neq$ Null and title.length $> 20$ then
3:         query $\leftarrow$ title
4:     else if lastName $\neq$ Null and year $\neq$ Null then
5:         query $\leftarrow$ lastName and year
6:     else if lastName $\neq$ Null then
7:         query $\leftarrow$ lastName
8:     end if
9:     return query
10: end function
```

**Algorithm 2** Header Matching Model

```
1: function HMM()
2:     T $\leftarrow$ target corpus
3:     R $\leftarrow$ reference corpus
4:     $index_R$ $\leftarrow$ index of reference corpus
5:     matchList $\leftarrow$Ø
6:     for $t \in T$ do
7:         Q $\leftarrow$ Query($t$.title, $t$.firstLastName, $t$.year)
8:         Candidates $\leftarrow$ query Q to $index_R$
9:         for c $\in$ Candidates do
10:            prediction$\leftarrow$Model.predict($t$,c)
11:            if prediction=1 then
12:                matchList.add $(t, c)$
13:                break
14:            end if
15:        end for
16:    end for
17: end function
```

**Algorithm 3** Citation Matching Model

```
1: function CMM()
2:     T $\leftarrow$ target corpus
3:     R $\leftarrow$ reference corpus
4:     matchList $\leftarrow$Ø
5:     $CitationIndex_R$ $\leftarrow$ citations index of reference corpus
6:     for $t \in T$ do
7:         $t\_citations$ $\leftarrow$ citations of $t$.
8:         for $tc_i \in t\_citations$ do
9:             Q $\leftarrow$ Query ($tc_i$.title,$tc_i$.firstLastName, $tc_i$.year)
10:            results $\leftarrow$ query Q to $CitationIndex_R$
11:            for $rc_j \in$ results do
12:                prediction$\leftarrow$Model.predict ($tc_i$, $rc_j$)
13:                if prediction=1 then                    ▷ citations match
14:                    $r_k$ $\leftarrow$ paper that cites $rc_j$
15:                    $r$.title $\leftarrow$ Simhash ($r_k$.title)
16:                    $t$.title $\leftarrow$ Simhash ($t$.title)
17:                    $title\_dist$ = lev ($t$.title, $r$.title)
18:                    if $title\_dist < \theta_{title}$ then
19:                        matchList.add ($t$, $r_k$)
20:                        break
21:                    end if
22:                    $BoW_r$ $\leftarrow$ BoW ($r_k$.referenceTitles)
23:                    $BoW_t$ $\leftarrow$ BoW ($t$.referenceTitles)
24:                    $sim$ $\leftarrow$ Jaccard($BoW_{t_i}$, $BoW_{r_i}$)
25:                    if $sim > \theta_{ref}$ then
26:                        matchList.add ($t$, $r_k$)
27:                        break
28:                    end if
29:                end if
30:            end for
31:        end for
32:    end for
33: end function
```

## Citation Matching Model (CMM)

**CMM** matches paper entities by citations. The paper entity matching problem can benefit from this model when the header metadata is noisy but the references are available. Similar to HMM, a prerequisite is to index all citations in the reference corpus. On average, one paper contains about 20 citations , so the citation index is usually much larger than the document index.

Given a target paper $t$, its citation records $tc_i$ are retrieved from the database. We attempt to find the matching record $rc_j$ in the reference corpus using the query builder in Algorithm 1. Retrieved citations and $rc_i$ are matched by the HMM (Algorithm 2). Citations do not contain abstracts so relevant features are not used. Assuming such $j = 1$ exists (if not, then no matching entity is found) and $rc_1$ is cited by $r_1$, the next step is to compare $r_1$ with $t$. The CMM uses both the *paper title* and the *citation titles* (Algorithm 3). First, the title similarities are calculated using Equations (1)–(3). If this distance is less than a threshold of $\theta_{title}$, $r_1$ is believed to be the matching entity of $t$. Otherwise, CMM extracts the tokens from all the reference titles of $t$, denoted by $BoW_t$, and tokens from all the reference titles of $r_1$, denoted by $BoW_r$. The judgment is made by comparing the Jaccard similarity between $BoW_t$ and $BoW_r$. If the similarity is greater than a threshold $\theta_{ref}$, then $(t, r_1)$ is determined as a matching pair.

Table 1: Features used to train TEM.

| **Character-level features** |
|---|
| #ASCII characters #non-ASCII characters #white spaces #punctuation marks #consecutive punctuation marks #digits Type of the first/last character (punctuation, digit, or letter) |
| **Word-level features** |
| $\#\max(\mathrm{DF}(w)), w \notin \mathbb{S}$[1] $\#\min(\mathrm{DF}(w)), w \notin \mathbb{S}$ $\#\mathrm{median}(\mathrm{DF}(w)), w \notin \mathbb{S}$ #words |
| #Appearance of one of the tokens in the controlled list[2] {Abstract, List, Acknowledgments, Notices, Content, Accepted, Authors, References, Acknowledgments, Null, Chapter, Discussions, Summary} |

[1] DF: document frequency, calculated on all DBLP titles. $\mathbb{S}$ is a set of stopwords adopted from Apache Solr.
[2] The value is set to 1 if the string contains at least one exact match to the controlled list.

Otherwise, the algorithm continues to examine the next paper that shares citation $rc_1$ with $t$. If no papers citing $rc_1$ is found to be a match for $t$, CMM continues and attempts to find the matching record of $tc_2$.

## Title Evaluation Model (TEM)

**TEM** is a light-weight supervised learning model designed to provide a quantitative evaluation of the title quality. The input is a title string, and the output is a probability $\theta$ of how likely the input string looks like a paper title. The title quality is determined to be high if $\theta$ is greater than a threshold $\theta_{tq}$. The TEM exploits the features in Table 1 extracted from the original title string. The TEM is trained on a sample of 8200 title strings containing 6270 high-quality titles and 1930 titles with low quality. Titles are labeled to be of low quality if (1) They are NULL; (2) They have many non-ASCII characters; (3) They include evidently irrelevant information such as authors; (4) They are not in English.

Four supervised models, including Logistic Regression (LR), Support Vector Machine (SVM), Naïve Bayes (NB), and Random Forests (RF), are trained. The LR model achieves the best 10-fold cross-validated F1-score of 0.999, which we adopted.

**IMM** integrates HMM, CMM, and TEM (Figure 1). If HMM is able to find the match of a paper entity, the process continues to the next paper. Otherwise, the paper title quality is evaluated by TEM. If the title quality is high ($\theta \geq \theta_{tq}$), it is likely that there is not a matching entity existing in the reference corpus, otherwise ($\theta < \theta_{tq}$), the matching entity is not found due to the poor title quality. In this situation, we use citation information to match papers.

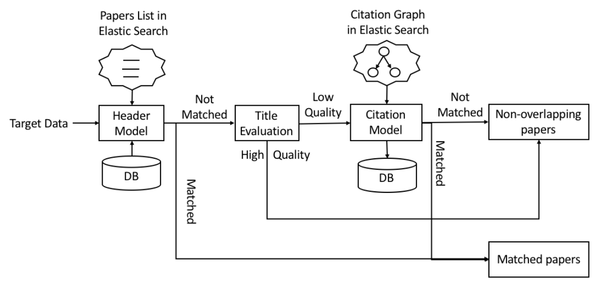

Figure 1: The pipeline of IMM.

# Experiments

## Data

The target data is the CiteSeerX database with about 9 million scholarly papers. The header metadata is extracted by GROBID. References are extracted and parsed by ParsCit (Councill, Giles, and Kan 2008). The reference corpora are described below.

**WoS** is a digital library dataset spanning 230+ academic disciplines with citation indexing. WoS indexing coverage is from 1900 to 2015 with over 20,000 journals, books, and conference proceedings. There are about 45 million WoS papers and 906 million citation records in this corpus.

**DBLP** is a bibliographic dataset covering more than 5,000 conferences and 1,500 journals in computer science. We use the version published in March, 2017 with about 4 million documents. This dataset does not contain citations.

**Medline** is the premier bibliographic dataset released by NCBI with about 24 million academic papers in the area of biomedicine published since 1966. The dataset does not contain citation information.

The **IEEE** corpus is a *subset* of the IEEE Xplore database, containing about 2 million bibliographic records downloaded from IEEE FTP sites. It does not contain citations.

## Ground Truth Labeling

The procedure for labeling process comprises three steps. First, for each paper in the CiteSeerX sample set, a candidate set of 10 papers is retrieved from the reference index using the same manner described in Algorithm 1. Then, to determine the true matches out of candidate papers, other metadata of the papers including authors, abstract, year, venue, keywords, and the number of pages were visually inspected independently by two graduate students. Finally, if it was not possible to decide based on papers profiles, actual PDF files were used side by side to make final decisions. We generated the following ground truth datasets:

**CiteSeerX-IEEE** This dataset, adopted from Wu et al. (2017), is built based on 1000 CiteSeerX papers with 51 true matching pairs found in the IEEE corpus.

**CiteSeerX-DBLP** This dataset, revised based on Caragea et al. (2014), contains 292 matching pairs identified between 1000 CiteSeerX papers and the DBLP dataset.

**CiteSeerX-WoS** This dataset contains 345 matching papers found in WoS out of 533 CiteSeerX papers.

**Combined Sample** The positive sample contains 688 matching pairs. The negative samples are selected using 1845 candidate matching pairs, containing the most similar but unmatched papers.

## Experiment Setups

We trained binary classifiers that decide whether a pair of documents from target and reference corpora is a true match-

Table 2: HMM model 10-fold CV results.

| Model | Precision | Recall | F1-measure |
|---|---|---|---|
| **SVM** | **0.926** | **0.937** | **0.931** |
| LR | 0.794 | 0.968 | 0.872 |
| RF | 0.912 | 0.931 | 0.921 |
| XGBoost | 0.925 | 0.899 | 0.912 |

ing pair. Four machine learning models, SVM, LR, RF, and XGBoost are trained on the Combined Sample. Grid search is applied to tune and find the hyper parameters yielding the best results. Precision, Recall, and F1-score values for 10-fold CV are reported in Table 2.

## Results and Discussion

In four models, SVM achieves the highest F1-measure. RF has a comparable F1-measure but requires a significantly reduced test time ($< 30\%$), so we employ RF for HMM. The information gain (IG) is calculated for each feature indicating that the most informative features are related to titles and the first authors.

To make an even comparison with the method proposed by Caragea et al. (2014), we rerun their experiments on the CiteSeerX-DBLP ground truth with the best parameter settings in which $n = 3$ and the Jaccard similarity threshold $\theta_J = 0.7$. We then compare the results with our RF model trained on the combination of CiteSeerX-WoS and CiteSeerX-IEEE datasets. The HMM outperforms the IR-based model with 14% improvement in precision (100% vs. 86%) and a 3% improvement in the F1 score (91% vs. 88%).

We investigate how the reference Jaccard similarity threshold $\theta_{ref}$ and title Levenshtein distance $\theta_{title}$ affects the performance of CMM (Table 3). A higher value of $\theta_{ref}$ indicates that two papers need more common citations to be considered as a matching pair. The best F1 score is obtained at $\theta_{ref} = 0.5$ and $\theta_{title} = 0.35$.

Table 3: The CMM performance with different $\theta_{ref}$ and $\theta_{title}$.

| $\theta_{ref}$ | $\theta_{title}$ | Precision | Recall | F1 |
|---|---|---|---|---|
| 0.40 | 0.15 | 0.876 | 0.719 | 0.790 |
| | 0.25 | 0.877 | 0.725 | 0.794 |
| | 0.35 | 0.878 | **0.730** | 0.797 |
| | 0.45 | 0.850 | 0.725 | 0.782 |
| 0.50 | 0.15 | 0.968 | 0.690 | 0.797 |
| | 0.25 | 0.969 | 0.714 | 0.822 |
| | 0.35 | 0.965 | 0.728 | **0.830** |
| | 0.45 | 0.927 | 0.725 | 0.814 |
| 0.60 | 0.15 | 0.982 | 0.651 | 0.783 |
| | 0.25 | **0.983** | 0.662 | 0.791 |
| | 0.35 | 0.979 | 0.691 | 0.810 |
| | 0.45 | 0.938 | 0.691 | 0.796 |
| 0.70 | 0.15 | 0.955 | 0.609 | 0.743 |
| | 0.25 | 0.955 | 0.620 | 0.752 |
| | 0.35 | 0.953 | 0.652 | 0.775 |
| | 0.45 | 0. 912 | 0.658 | 0.764 |

Table 4 compares HMM, CMM, and IMM based on the CiteSeerX-WoS dataset (because only this dataset contains citations). In the first column, as the threshold $\theta_{tq}$ increases from 0.01 to 0.2, the testing corpus encloses more papers with higher quality titles, which results in a better performance of HMM. The CMM alone is getting better with remarkably high precision but poor recalls. The integrated model achieves both high recall and precision. This indicates that (1) CMM tend to be more useful when the title quality is low; (2) The integrated model significantly increases the overall performance, especially for papers with low quality titles.

One result in Table 4 that is counter-intuitive is the HMM consistently achieves high performance when the title quality is low. To answer this question, we trained a RF classifier on papers with low-quality titles only. IG of the new model reveals that the most important features in the absence of good titles are First author features, Jaccard similarity of all authors' last names, and Abstract features, implying that when title quality is low, accurate author information can also provide accurate matches.

## Error Analysis

Although the combination of HMM and CMM achieve superior performance, the recall of CMM alone is poor (Table 4). This could be due to two reasons: (1) Citation parsing errors. For example, more than 1 million papers in CiteSeerX contain less than 5 citations; (2) Null title citations. About 17% of citation records in WoS and 8.4% of citations in CiteSeerX have null titles.

The citation-based model is slow because (1) the large number of citations (906 million) slows down the search process and (2) the candidate set for each CiteSeerX citation could be huge for highly-cited papers. The integrated model only applies citation model to papers with low-quality titles to improve recall.

## Application and Conclusion

We applied HMM on CiteSeerX documents against DBLP, WoS, and Medline. The result indicates that **the current CiteSeerX dataset includes about 3 million WoS documents, 1.62 million Medline papers, and about 1.35 million DBLP papers**. The matching process by HMM is done in 11 days on a machine with following specifications: 32 logical cores of Intel Xeon CPU E5-2630 v3 @ 2.40GH; 330 GB RAM. The result reveals that there is still a large number of papers that CiteSeerX should index. The unmatched document metadata can aid CiteSeerX crawler to find relevant resources.

Previous studies (Caragea et al. 2014; Wu et al. 2017) used only metadata in the header of scholarly articles for paper entity linking. In reality, the quality of a header is not always that good. Hence, we investigated leveraging both header and citation information to match paper entities between two digital library datasets when the target corpus contains noisy data. We proposed an approach that integrates header and citation information for paper entity matching. Compared with the IR-based method, header matching model improves precision by at least 13% and F1 by about 3% for papers with low quality titles. The integrated model with header and citation information achieves an F1 as high as 0.992 and precision as high as 0.984. We

Table 4: Comparisons of HMM, CMM, and IMM performances using the CiteSeerX-WoS dataset with different title quality thresholds. $T/s$ stands for testing time in seconds.

| $\theta < \theta_{tq}$ | **Data Portion** | **HMM** | | | | **CMM** | | | | **IMM** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | P | R | F1 | $T/s$ | P | R | F1 | $T/s$ | P | R | F1 | $T/s$ |
| $\theta < 0.01$ | 16.1 % | 0.971 | 0.872 | 0.919 | 10 | 1.0 | 0.513 | 0.678 | 12229 | 0.975 | 1.0 | 0.987 | 9082 |
| $\theta < 0.02$ | 17.8 % | 0.973 | 0.857 | 0.911 | 12 | 1.0 | 0.524 | 0.688 | 12761 | 0.977 | 1.0 | 0.988 | 9791 |
| $\theta < 0.10$ | 21.20% | 0.978 | 0.833 | 0.900 | 14 | 1.0 | 0.593 | 0.745 | 13732 | 0.982 | 1.0 | 0.991 | 10206 |
| $\theta < 0.20$ | 24.39% | 0.981 | 0.869 | **0.922** | 14.5 | 1.0 | 0.59 | 0.742 | 14823 | 0.984 | 1.0 | **0.992** | 11490 |

show that CiteSeerX has a huge overlap with WoS, DBLP, and Medline, which can be used for metadata correction, and that there are still a large number of scientific documents to be crawled and indexed. The framework developed can be used to match records between any bibliographic databases with or without citations. The idea of combining ML and IR is in general applicable to many information retrieval and data linking problems. We will apply this framework to clean the CiteSeerX metadata. This will generate high quality large-scale datasets that can enable development and implementation of many graph-based AI applications.

The software implementation of this framework is available on GitHub at
https://github.com/SeerLabs/entity-matching.

## Acknowledgements

We gratefully acknowledge partial support from the National Science Foundation.

## References

Al-Zaidy, R. A., and Giles, C. L. 2017. A machine learning approach for semantic structuring of scientific charts in scholarly documents. In *AAAI*, 4644–4649.

Caragea, C.; Wu, J.; Ciobanu, A.; Williams, K.; Fernández-Ramírez, J.; Chen, H.-H.; Wu, Z.; and Giles, L. 2014. Citeseerx: A scholarly big dataset. In *Advances in Information Retrieval: 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings*, 311–322.

Charikar, M. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, 380–388.

Chen, C.; Wang, Z.; Li, W.; and Sun, X. 2018. Modeling scientific influence for research trending topic prediction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

Councill, I.; Giles, C. L.; and Kan, M.-Y. 2008. Parscit: an open-source crf reference string parsing package. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08).*

Giles, C. L.; Bollacker, K. D.; and Lawrence, S. 1998. Citeseer: An automatic citation indexing system. In *Proceedings of the 3rd ACM International Conference on Digital Libraries, June 23-26, 1998, Pittsburgh, PA, USA*, 89–98.

Giles, C. L. 2013. Scholarly big data: Information extraction and data mining. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, 1–2. New York, NY, USA: ACM.

Huang, W.; Wu, Z.; Chen, L.; Mitra, P.; and Giles, C. L. 2015. A neural probabilistic model for context based citation recommendation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 2404–2410.

Kim, K.; Sefid, A.; and Giles, C. L. 2017. Scaling author name disambiguation with cnf blocking. *arXiv preprint arXiv:1709.09657.*

Liu, X.; Yan, J.; Xiao, S.; Wang, X.; Zha, H.; and Chu, S. M. 2017. On predictive patent valuation: Forecasting patent citations and their types. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 1438–1444.

Peled, O.; Fire, M.; Rokach, L.; and Elovici, Y. 2013. Entity matching in online social networks. In *International Conference on Social Computing, SocialCom 2013, SocialCom/PASSAT/BigData/EconCom/BioMedCom 2013, Washington, DC, USA, 8-14 September, 2013*, 339–344.

Robertson, S.; Zaragoza, H.; and Taylor, M. 2004. Simple bm25 extension to multiple weighted fields. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, 42–49. New York, NY, USA: ACM.

Wesley-Smith, I., and West, J. D. 2016. Babel: A platform for facilitating research in scholarly article discovery. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, 389–394. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee.

Wu, J.; Williams, K.; Chen, H.; Khabsa, M.; Caragea, C.; Ororbia, A.; Jordan, D.; and Giles, C. L. 2014. Citeseerx: AI in a digital library search engine. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 2930–2937.

Wu, J.; Sefid, A.; Ge, A. C.; and Giles, C. L. 2017. A supervised learning approach to entity matching between scholarly big datasets. In *Proceedings of the Knowledge Capture Conference*, K-CAP 2017, 42:1–42:4. New York, NY, USA.

Yang, Y.; Sun, Y.; Tang, J.; Ma, B.; and Li, J. 2015. Entity matching across heterogeneous sources. In Cao, L.; Zhang, C.; Joachims, T.; Webb, G. I.; Margineantu, D. D.; and Williams, G., eds., *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, 1395–1404. ACM.