CiteSeer^x: A Scholarly Big Dataset

Cornelia Caragea[1,4], Jian Wu[2,5], Alina Ciobanu[3,6], Kyle Williams[2,5], Juan Fernández-Ramírez[1,7], Hung-Hsuan Chen[1,5], Zhaohui Wu[1,5], and Lee Giles[1,2,5]

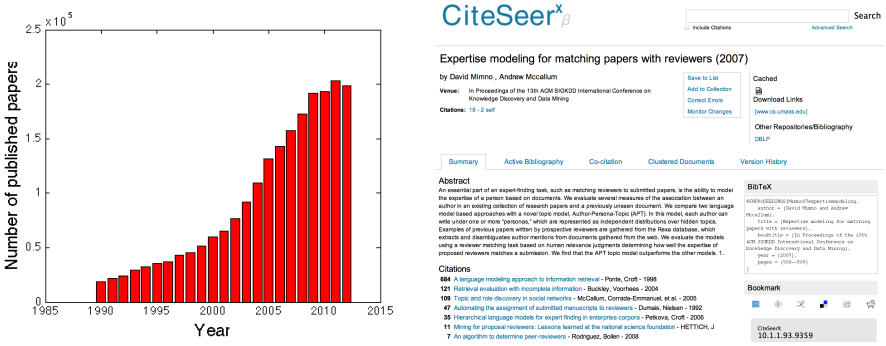
[1]Computer Science and Engineering, [2]Information Sciences and Technology, [3]Computer Science, [4]University of North Texas, Denton, TX, USA, [5]Pennsylvania State University, University Park, PA, USA, [6]University of Bucharest, Bucharest, Romania, [7]University of the Andes, Bogota, Colombia
ccaragea@unt.edu; jxw394@ist.psu.edu; alina.ciobanu@my.fmi.unibuc.ro; kwilliams@psu.edu; jp.fernandez29@uniandes.edu.co; hhchen@psu.edu; zzw109@psu.edu; giles@ist.psu.edu

Abstract. The CiteSeer^x digital library stores and indexes research articles in Computer Science and related fields. Although its main purpose is to make it easier for researchers to search for scientific information, CiteSeer^x has been proven as a powerful resource in many data mining, machine learning and information retrieval applications that use rich metadata, e.g., titles, abstracts, authors, venues, references lists, etc. The metadata extraction in CiteSeer^x is done using automated techniques. Although fairly accurate, these techniques still result in noisy metadata. Since the performance of models trained on these data highly depends on the quality of the data, we propose an approach to CiteSeer^x metadata cleaning that incorporates information from an external data source. The result is a subset of CiteSeer^x, which is substantially cleaner than the entire set. Our goal is to make the new dataset available to the research community to facilitate future work in Information Retrieval.

Keywords: CiteSeer^x; Scholarly Big Data; Record Linkage.

1 Introduction

As science advances, scientists around the world continue to produce large numbers of research articles, which provide the technological basis for worldwide dissemination of scientific discoveries. Online digital libraries such as DBLP, CiteSeer^x, Microsoft Academic Search, ArnetMiner, arXiv, ACM Digital Library, Google Scholar, and PubMed that store research articles or their metadata, have become a medium for answering questions such as: how research ideas emerge, evolve, or disappear as a topic; what is a good measure of quality of published works; what are the most promising areas of research; how authors connect and influence each other; who are the experts in a field; and what works are similar. Unfortunately, our ability to manually process and filter the huge amount of information available in digital libraries lags far behind the number of research articles available today. Figure 1(a) shows the growth in the number of research articles published between 1990 and 2011, extracted from the DBLP dataset.



(a) DBLP Growth (b) CiteSeer^{x} Metadata Example

Fig. 1: (a) The growth in the number of research papers published between 1990 and 2011, extracted from DBLP; (b) An example of CiteSeer^{x} metadata.

Recent developments in data mining, machine learning, and information retrieval made it possible to transform the way we analyze research articles on a web-wide scale. CiteSeer^{x} [1] has been proven as a powerful resource in many applications such as text classification [2,3,4], collective classification [5], document and citation recommendation [6,7,8,9,10], author name disambiguation [11], expert search [12], collaborator recommendation [13], paper and slides alignment [14], author influence [15], joint modeling of documents' content and interests of authors [16], and entity resolution [17].

To extract metadata from each article, the CiteSeer^{x} project uses automated techniques [18,19]. An example of metadata for an entry in CiteSeer^{x} (i.e., a research article) is shown in Figure 1(b). The metadata contains information such as the title of an article, the authors, venue where the article was published, the year of publication, article's abstract, and the references list. Among the automated techniques for metadata extraction used in CiteSeer^{x}, Han et al. [18] employed a Support Vector Machine based classification method to extract information from the header of research papers. Councill et al. [19] used Conditional Random Fields to segment reference strings from plain text, and used heuristic rules to identify reference strings and citation contexts. Although fairly accurate, these automated techniques still make mistakes during the metadata extraction process. Thus, the metadata inherently contains noise, e.g., errors in the extraction of the year of publication.

In contrast, DBLP provides *manually* curated metadata. However, DBLP metadata is not as rich as that provided by CiteSeer^{x}. For example, DBLP does not provide an article's abstract or references strings, which are crucial in applications such as citation recommendation and topic evolution, that use either the citation graph and/or textual information.

In this paper, we present a record linkage based approach to building a *scholarly big dataset* that uses information from DBLP to automatically remove noise

in CiteSeer^x. Inspired by the work on record linkage by Bhattacharya and Getoor [20], we study the usage of both the similarity between paper titles and the similarity between the authors' lists of "similar" papers and their number of pages.

The contributions of the research described here are two-fold. First, we present and study an approach to building a scholarly dataset derived from CiteSeer^x that is substantially cleaner than the entire set. Second, the new dataset will be made available to the research community and can benefit many research projects that make use of rich metadata such as a citation graph and textual information available from the papers' abstracts. The dataset will be maintained and updated regularly to compile the dynamic changes in CiteSeer^x.

The rest of the paper is organized as follows. We present related work in Section 2. In Section 3, we describe our approach to CiteSeer^x data cleaning. The evaluation and the characteristics of the new dataset are presented in Section 4. We conclude the paper with a summary and discussion of limitations of our approach, and directions for future work in Section 5.

2 Related Work

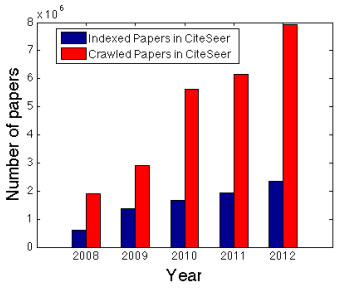
Record linkage refers to the problem of identifying duplicate records that describe the same entity across multiple data sources [21] and has applications to data cleaning, i.e., discovering and removing errors from data to improve data quality [22], and information integration, i.e., integrating multiple heterogeneous data sources to consolidate information [23].

Many approaches to record linkage and its variants, e.g., duplicate detection or deduplication, co-reference or entity resolution, identity uncertainty, and object identification, have been studied in the literature. Several research directions considered are: the classification of record pairs as "match" or "not match" [21,24,25,26]; design of approximate string matching algorithms [27], adaptive algorithms that learn string similarity measures [24,25], iterative algorithms that use attribute as well as linked objects similarities [17,20]; object identification and co-reference resolution [23,28,29]; near-duplicates detection by identifying f-bit fingerprints that differ from a given fingerprint in at most k-bit positions [30]; and algorithms for reference matching in the bibliometrics domain [28].

Our work builds upon previous works on record linkage and uses information from DBLP to improve the quality of CiteSeer^x metadata records. An approach to building a citation network dataset using DBLP was proposed by ArnetMiner[1]. However, our approach results in a scholarly dataset that contains richer metadata compared with ArnetMiner's DBLP citation dataset, e.g., it contains the *citation contexts* for a paper's references list.

To our knowledge, this is the first attempt to generate a *scholarly big dataset* that consists of cleaner CiteSeer^x metadata, that will be made available to the research community, and will aim at facilitating future work in Information Retrieval. Through the merger of CiteSeer^x and DBLP, errors in CiteSeer^x metadata

[1] http://arnetminer.org/DBLP_Citation



(a) CiteSeer^x Growth

Total number of indexed documents:	2,356,568
Unique documents (after clustering):	1,957,448
Authors in collection:	7,511,558
Unique authors (identified by name):	2,447,515
Disambiguated authors:	304,833
Total number of citations:	52,185,298
Unique records (citations+documents):	15,375,541

(b) CiteSeer^x Statistics

Fig. 2: (a) The growth in the number of crawled documents as well as in the number of documents (or research papers) indexed by CiteSeer^x between 2008 and 2012; (b) Statistics of the entire CiteSeer^x dataset.

fields will be corrected. Only clean CiteSeer^x records with a match in DBLP will be retained in the dataset, whereas the rest will be filtered out.

3 Approach to Data Cleaning

We first present the characteristics of the entire collection of articles indexed in CiteSeer^x, and then describe our approach to data cleaning and its evaluation.

3.1 Characteristics of the Entire CiteSeer^x Dataset

The CiteSeer^x dataset is rapidly growing in size. Figure 2(a) shows the increase in both the number of crawled documents as well as the number of documents (or research papers) indexed by CiteSeer^x during the last five years. As can be seen from the figure, the number of crawled documents has increased from less than two million to almost eight million, whereas the number of indexed documents has increased from less than one million to more than two million. Note that because CiteSeer^x only crawls open-access documents (e.g., those available from authors' webpages), many of the crawled documents are manuscripts.

As of May 2013, the total number of documents indexed in CiteSeer^x is ≈2.35M. After clustering to remove multiple versions of a paper, there are ≈1.9M unique papers. The number of authors in collection, i.e., authors (with repetition) from all documents is ≈7.5M, with the number of unique authors being ≈2.4M (i.e., authors without repetition), and the number of disambiguated authors being ≈300K (e.g., "Andrew McCallum", "Andrew K. McCallum" and "A. McCallum" are disambiguated to the same author, whereas "Wei Hong" is disambiguated to multiple authors by their affiliations). The number of citations in collection, i.e., all papers with repetition that occur in the references list of all documents is ≈52M, and the number of unique records, i.e., unique papers and citations, is ≈15M. The exact numbers are summarized in Figure 2(b).

Algorithm 1 CiteSeer^x Data Cleaning

Input:
$\mathcal{C} \leftarrow$ CiteSeer^x metadata entries; $\mathcal{D} \leftarrow$ DBLP metadata entries;
$\theta \leftarrow$ a predefined threshold; **boolean** checkAuthors, checkPageCount.
Output: A subset of CiteSeer^x cleaned using DBLP, $\mathcal{C}^{\mathcal{D}}$.
Index the fields in $\mathcal{D}$ into an inverted index;
$\mathcal{C}^{\mathcal{D}} = \phi$;
for all entries $e \in \mathcal{C}$ **do**
  $t_e \leftarrow$ `getTitle`(e); // Query t_e = title of entry e
  $\mathcal{D}_e \leftarrow$ `retrieve`(t_e); // $\mathcal{D}_e$ contains the hits from DBLP for the query title t_e
  for all $d \in \mathcal{D}_e$ **do**
    $t_d \leftarrow$ `getTitle`(d); // t_d = title of d
    $s \leftarrow sim(t_d, t_e)$;
    if $s \geq \theta$ **then**
      if (checkAuthors = false) && (checkPageCount = false) **then**
        $m \leftarrow$ `match`(d, e); // write the fields from d to e
      end if
      $a_e \leftarrow$ `getAuthors`(e); $a_d \leftarrow$ `getAuthors`(d);
      if (checkAuthors = true) && (checkPageCount = false) && $(a_e \subseteq a_d)$ **then**
        $m \leftarrow$ `match`(d, e); // write the fields from d to e
      end if
      $p_e \leftarrow$ `PageCount`(e); $p_d \leftarrow$ `PageCount`(d);
      if (checkAuthors = false) && (checkPageCount = true) && $(p_e \approx p_d)$ **then**
        $m \leftarrow$ `match`(d, e); // write the fields from d to e
      end if
      $\mathcal{C}^{\mathcal{D}} = \mathcal{C}^{\mathcal{D}} \cup m$;
    end if
  end for
end for
return $\mathcal{C}^{\mathcal{D}}$ // The subset of CiteSeer^x cleaned using DBLP

3.2 Building a Cleaner CiteSeer^x Dataset

Our approach to building a cleaner CiteSeer^x dataset is to merge metadata information from CiteSeer^x and DBLP. The procedure for merging these two sources of information is shown in Algorithm 1. The input of the algorithm is given by two sets $\mathcal{C}$ and $\mathcal{D}$ of CiteSeer^x and DBLP metadata entries, respectively, and a threshold θ. The output is $\mathcal{C}^{\mathcal{D}}$, a merged dataset of CiteSeer^x and DBLP entries, obtained by performing record linkage.

The algorithm starts by indexing the entries in $\mathcal{D}$ into an inverted index. Specifically, titles and authors from DBLP are indexed using Solr 4.3.0, an indexing platform that is built using Apache Lucene[2]. Next, the algorithm iterates through all the entries e in $\mathcal{C}$ and treats each CiteSeer^x title as a query that is used to search against the DBLP indexed entries. For each query title t_e, a candidate set $\mathcal{D}_e$ of DBLP entries is retrieved in the following way: we extract the n-grams from the query title and retrieve all entries from DBLP that have at least one n-gram in common with the query (an n-gram is defined as a sequence of n contiguous tokens in a title). More precisely, for the bi-gram "expertise modeling" in the query "Expertise modeling for matching papers with reviewers.", we retrieve all DBLP entries that contain the bi-gram in their titles. In exper-

[2] http://lucene.apache.org

iments, we used various values of n for the length of an n-gram ($n = 1, 2, 3, 4$, and $|t_e|$, where $|t_e|$ represents the length, in the number of tokens, of t_e).

In our record linkage algorithm, we studied the effect of using only title similarity on the matching performance as well as using the similarity between other attributes of entries such as papers' author lists *or* their page count, in addition to title similarity. After the candidate set $\mathcal{D}_e$ is retrieved, the algorithm computes the similarity between the query title t_e and the titles t_d of entries in $\mathcal{D}_e$. If the similarity is greater than θ and if only title information is required, the algorithm performs the match between t_e and t_d and adds the match to $\mathcal{C}^{\mathcal{D}}$. Otherwise, if additional information is required besides title similarity, the algorithm checks one of the following conditions: (1) if the list of authors a_e of e is included in the list of authors a_d of d, or (2) if the page count p_e of e is approximately the same as the page count p_d of d. If the condition is satisfied, a match between e and d is performed. More precisely, the fields available in DBLP overwrite those in CiteSeer$^{\mathtt{x}}$ (the fields in CiteSeer$^{\mathtt{x}}$ with no equivalent in DBLP are kept the same). If more matches are found for a CiteSeer$^{\mathtt{x}}$ entry, which satisfy the condition, the DBLP entry with the highest similarity score with the query title is returned as a match. If no DBLP match is found, the CiteSeer$^{\mathtt{x}}$ record is not added to $\mathcal{C}^{\mathcal{D}}$. The algorithm terminates with the set $\mathcal{C}^{\mathcal{D}}$ of CiteSeer$^{\mathtt{x}}$ records that are merged with their equivalent entries from DBLP.

In our implementation, for condition (1), we considered only the authors' last names to avoid ambiguities caused by various uses of authors' first names in the two data sources (e.g., "A. McCallum" vs. "Andrew McCallum" in CiteSeer$^{\mathtt{x}}$ and DBLP, respectively). Author disambiguation applied to both CiteSeer$^{\mathtt{x}}$ and DBLP will be considered in future, for further versions of $\mathcal{C}^{\mathcal{D}}$.

For condition (2), we extracted the page count from DBLP from the field: $<$ `pages` $>$ `page_start` $-$ `page_end` $<$ /`pages` $>$. For CiteSeer$^{\mathtt{x}}$ entries, we used PDFBox[3] to determine the page count directly from the pdf file of each document (note that the pdf is available for each document in CiteSeer$^{\mathtt{x}}$).

4 Evaluation of the Proposed Approach

To evaluate our approach, we randomly sampled 1000 documents from CiteSeer$^{\mathtt{x}}$ and manually identified their matching records in DBLP. We describe the manual labeling process first and then present the experimental design and results.

4.1 Manual Labeling of the Random CiteSeer$^{\mathtt{x}}$ Sample

For each of the records in the random sample, we searched DBLP for a "true match". Specifically, we used the *actual* paper title found from the pdf of the paper to query the Solr index and retrieve a candidate set of DBLP records (we again used Solr 4.3.0 to index the DBLP fields). In determining the true DBLP match for a CiteSeer$^{\mathtt{x}}$ document, we also used information about authors, year, and venue, obtained from the profile page of the document (through

[3] http://pdfbox.apache.org

the CiteSeer^x web service), as well as the number of pages, obtained either by checking the pdf of the document or by using PDFBox. If a matching decision was difficult to make (e.g., due to a one-to-many relation, one CiteSeer^x record and multiple DBLP search results), we downloaded the corresponding pdfs and made a side-by-side comparison. The general criteria for decision making are:

1. Matched papers must have the same title and author lists;
2. The order of importance for other metadata is venue > year > page count;
3. If only title and authors are available, the original papers are downloaded and compared side-by-side.

After completing the manual labeling of all documents in the sample, we found 236 documents had true matches in DBLP. Since this sample is randomly selected in the DOI space, it reflects the average properties of the entire CiteSeer^x sample.

4.2 Experimental Design

Our experiments are designed around the following questions. *How does the matching performance vary when we vary the threshold θ on the similarity between a query title and the DBLP entries relevant to the query and what is the effect of the similarity measure on the matching performance?* High thresholds impose high overlaps of words between two titles, whereas low thresholds allow for low overlaps, which can handle CiteSeer^x titles that are wrongly extracted (e.g., when an author name is included in the title, or when only the first line is extracted for a title that spans multiple lines). In experiments, we varied the threshold θ from 0.5 to 0.9, in steps of 0.1. We also experimented with two similarity measures: Jaccard and cosine, and found Jaccard to perform better than cosine (see the Results section). In subsequent experiments, we used Jaccard.

The next question is: *Is the proposed approach to data cleaning computationally efficient?* The most expensive part of Algorithm 1 is the Jaccard similarity calculation (Jaccard similarity is given by the number of common words between two titles divided by the total number of unique words). The larger the size of the retrieved document set is (i.e., the size of $\mathcal{D}_e$), the more Jaccard similarity computations are needed. We experimented with several techniques for query construction to control the size of $\mathcal{D}_e$. We compared the matching performance for all tested query construction techniques to determine what is the *most time efficient* and *highly accurate* among these techniques. Specifically, the query construction techniques to retrieve the candidate set $\mathcal{D}_e$ of DBLP entries for a query t_e are n-gram queries, with $1 \leq n \leq |t_e|$, defined as follows:

– n-gram query: $\mathcal{D}_e = \{d \in \text{DBLP} \mid d \text{ has } \textit{at least} \text{ one } n\text{-gram in common with } t_e\}$, $1 \leq n \leq |t_e|$, where $|t_e|$ is the length of t_e, in the number of tokens.

The use of $|t_e|$-gram (called AND) query results in a small size of the set $\mathcal{D}_e$, whereas the use of the unigram (called OR) query results in a large size of the set. The n-gram query ($2 \leq n \leq |t_e| - 1$) presents a tradeoff between AND and OR. For each query type above, we also experimented with the setting where we

note that here AND connects all tokens, e.g., review AND support AND vector AND machine.
when n is less than |t_e|, the query results of different n-grams are *always* merged (UNION, an OR operation)

θ	AND				3-gram				OR			
	Prec.	Recall	F1-score	time	Prec.	Recall	F1-score	time	Prec.	Recall	F1-score	time
0.5j	0.694	0.691	0.692	55	0.627	0.826	0.713	106	0.519	0.826	0.637	3722
0.6j	0.744	0.691	0.716	42	0.713	0.809	0.758	65	0.692	0.809	0.746	4100
0.7j	0.756	0.682	0.717	44	0.746	0.797	0.770	67	0.740	0.797	0.767	4088
0.8j	0.768	0.674	0.718	43	0.765	0.746	0.755	64	0.762	0.746	0.754	4064
0.9j	0.772	0.661	0.712	43	0.769	0.678	0.721	65	0.769	0.678	0.721	3616
0.5c	0.652	0.699	0.675	42	0.484	0.847	0.616	63	0.274	0.843	0.414	4158
0.6c	0.679	0.699	0.689	43	0.561	0.839	0.672	66	0.410	0.835	0.55	4102
0.7c	0.688	0.691	0.689	44	0.648	0.818	0.723	67	0.580	0.818	0.678	4201
0.8c	0.733	0.686	0.709	43	0.726	0.809	0.766	66	0.715	0.809	0.759	4116
0.9c	0.763	0.669	0.713	43	0.763	0.725	0.743	67	0.763	0.725	0.743	4178

Table 1: The matching performance, using only titles, for various values of θ, for Jaccard (j) and cosine (c) similarities, and AND, 3-gram, and OR query types.

removed the stop words from titles, which significantly reduces the size of $\mathcal{D}_e$. In experiments, we found that the matching performance with and without stop words are similar or the same, whereas the time spent to compute the Jaccard similarities is significantly reduced, by a factor of 3, when stop words are removed (the size of $\mathcal{D}_e$ is much smaller). Hence, we report the results without stop words.

Our last question is: *What is the effect of using the similarity between other attributes such as author lists or page count, in addition to title similarity?* We experimented with the following settings for match identification: title only, title + authors, and title + page count.

To evaluate the performance of Algorithm 1 for matching CiteSeer$^{\mathtt{x}}$ and DBLP entries, we report precision, recall and F1-score on the manually annotated sample of 1000 randomly selected CiteSeer$^{\mathtt{x}}$ entries. Precision gives the fraction of matches correctly identified by the algorithm among all matches identified by the algorithm, whereas recall gives the fraction of matches correctly identified by the algorithm among all actual matches. F1-score gives the harmonic mean of precision and recall. With the best setting that we obtained on the manually annotated sample, we then ran experiments on the entire CiteSeer$^{\mathtt{x}}$. We report the characteristics of the newly constructed dataset, i.e., the resulting set from the merger of CiteSeer$^{\mathtt{x}}$ and DBLP.

4.3 Results

The effect of varying the threshold θ, the similarity measure, as well as the query type on the matching performance. Table 1 shows the comparison of CiteSeer$^{\mathtt{x}}$-DBLP matching performance for various values of the threshold θ on the similarity between a query and a DBLP title, ranging from 0.5 to 0.9 in steps of 0.1, using two similarity measures, Jaccard (j) and cosine (c), on the manually labeled sample of 1000 CiteSeer$^{\mathtt{x}}$ entries. The results are shown for three query types, AND, 3-gram, and OR, with stop words removed, using only

title similarity (i.e., no author or page count information is considered). The time (in seconds) taken by each experiment to finish is also given in the table. As can be seen from the table, as we increase θ, and thus, impose a higher word overlap between a query and a DBLP title, the recall decreases, whereas the precision increases, in most cases, for all query types and for both Jaccard and cosine similarities. Hence, lower θ can handle better wrongly extracted titles (as shown by the higher recall), but too low θ allows matching entries that have some word overlap between their titles, which in fact are not true matches (as shown by the lower precision). Moreover, it can be seen from the table that, in most cases, Jaccard similarity yields better results compared with cosine similarity.

From the table, we can also see that the matching performance for the 3-gram query is generally better compared with both AND and OR queries. Although the time (in seconds) spent by 3-gram queries is slightly worse than that of AND queries, it is significantly shorter than that of OR queries. The average number of DBLP hits for a CiteSeer^{x} query, i.e., the size of $\mathcal{D}_e$, is $81,625$ for OR, and it drops substantially to 212 for 3-gram, and to 131 for AND. Hence, much less computations are needed for the 3-gram and AND queries compared with OR. For the 3-gram query, we performed experiments when the stop words were not removed and found that the matching performance remains the same with that of removing stop words. However, the time (in seconds) spent to finish an experiment increased by a factor of 5. The value $n = 3$ for the n-gram query was empirically selected based on comparisons with 2-gram and 4-gram queries. These results, using Jaccard similarity, are shown in Table 2. As can be seen from the table, for $\theta = 0.7$, the 3-gram query results in the highest F1-score of 0.77, with a slight increase in time compared with 4-gram query.

Although the size of the candidate set $\mathcal{D}_e$ is significantly reduced through an AND query, its use seems to be quite limiting since it requires that every word in the title extracted by CiteSeer^{x} must be matched for a document to be retrieved and added to $\mathcal{D}_e$. While the AND query will not affect retrieval for incomplete titles, however, if the extractor mistakenly appended an author name to a title, no DBLP hits are found, and hence, the DBLP cannot be used to fix the error in metadata. The AND query increases the precision, but results in a decrease in recall. The OR query overcomes the limitations of the AND query, however, the size of retrieved DBLP documents for which Jaccard similarity needs to be computed increases significantly. The 3-gram queries provide a good tradeoff between the size of the retrieved documents and the matching performance.

The effect of using author and page count on the matching performance. Table 3 shows, using a 3-gram query and Jaccard similarity, the comparison of the matching performance when only title information is used (Title Only) with the matching performance when additional information is used, i.e., papers' author lists (Title+Authors) or page count (Title+Pages). As can be seen from the table, the recall is fairly high using Title Only compared with Title+Authors and Title+Pages, but the precision drops. Title+Pages generally achieves the highest precision, e.g., 0.904 for $\theta = 0.9$. Thus, as more information is used, the precision increases, however, at the expense of decreasing recall. A

θ	**2-gram**				**3-gram**				**4-gram**			
	Prec.	Recall	F1-score	time	Prec.	Recall	F1-score	time	Prec.	Recall	F1-score	time
0.5j	0.560	0.826	0.668	156	0.627	0.826	0.713	106	0.662	0.805	0.726	53
0.6j	0.695	0.809	0.748	149	0.713	0.809	0.758	65	0.722	0.792	0.755	50
0.7j	0.743	0.797	0.769	152	0.746	0.797	0.770	67	0.747	0.779	0.763	51
0.8j	0.762	0.746	0.754	140	0.765	0.746	0.755	64	0.766	0.737	0.751	51
0.9j	0.769	0.678	0.721	143	0.769	0.678	0.721	65	0.769	0.677	0.720	52

Table 2: The matching performance, using only titles, for $2-4$-gram queries.

θ	**Title Only**				**Title+Authors**				**Title+Pages**			
	Prec.	Recall	F1-score	time	Prec.	Recall	F1-score	time	Prec.	Recall	F1-score	time
0.5j	0.627	0.826	0.713	106	0.819	0.631	0.713	66	0.802	0.551	0.653	64
0.6j	0.713	0.809	0.758	65	0.835	0.623	0.714	65	0.869	0.534	0.661	64
0.7j	0.746	0.797	0.770	67	0.847	0.610	0.709	67	0.875	0.534	0.663	66
0.8j	0.765	0.746	0.755	64	0.873	0.581	0.697	66	0.888	0.504	0.643	66
0.9j	0.769	0.678	0.721	65	0.868	0.530	0.658	66	0.904	0.479	0.626	66

Table 3: The matching performance, using title only, title+authors, and title+page count information.

potential explanation for low recall could be noisy extraction of authors' names in CiteSeer$^{\text{x}}$ or the mismatch between the page count. The page count is an additional evidence for record matching, which is independent of the metadata extraction quality. If two records have the same page count in addition to *similar* titles, they are likely to refer to the same document. However, during the manual inspection, we found that many matched papers did not have the same page count (e.g., often an extra page is added as a cover page from the institution or research lab). The CiteSeer$^{\text{x}}$ version was generally a manuscript. In experiments, we allowed ± 1 from the page count. Further investigation of this will be considered in future.

If the page count or authors' list for a paper cannot be extracted in one of the data sources, the CiteSeer$^{\text{x}}$ entry is skipped. If there is a one-to-many relationship between CiteSeer$^{\text{x}}$ and DBLP, the algorithm matches the CiteSeer$^{\text{x}}$ entry with the one from DBLP with the highest Jaccard similarity score.

5 Summary, Discussion, and Future Directions

We presented an approach to CiteSeer$^{\text{x}}$ metadata cleaning that uses information from DBLP to clean metadata in CiteSeer$^{\text{x}}$. In addition to using title similarity to perform record linkage between CiteSeer$^{\text{x}}$ and DBLP, we studied the use of additional information such as papers' author lists or page count. Results of experiments on a random sample of 1000 CiteSeer$^{\text{x}}$ entries show that the proposed approach is a promising solution to CiteSeer$^{\text{x}}$ metadata cleaning.

One of the major limitations of our approach is that it assumes the titles in CiteSeer^x are extracted correctly. However, there are many titles that are wrongly extracted. For example, we found: (i) titles that contain tokens that do not occur at all in the actual title[4]; (ii) titles that contain only one stop word, "and" or "the"[5]; (iii) titles that are incomplete[6]; and (iv) titles that contain other tokens besides the title tokens, such as author, venue, or year[7]. The algorithm will fail to find a matching record in DBLP for (i) and (ii) since the retrieved candidate DBLP set is not relevant to the actual title or the Jaccard similarity does not exceed the predefined threshold θ. For (iii) and (iv), it is possible that the algorithm will find a match in DBLP if θ is too low. Author or page count could help improve precision, however at the expense of decreasing recall (potentially due to noise in authors' name extraction or difference in page count). In future, we plan to use information for other external data sources such as IEEE and Microsoft Academic Search to improve data quality in CiteSeer^x. Additional information, e.g., venue names will be investigated in future as well.

5.1 Dataset Characteristics, Sharing and Maintenance

We generated a *scholarly big dataset* of cleaner CiteSeer^x metadata records. The dataset is made available to the research community, along with our Java implementation at http://www.cse.unt.edu/∼ccaragea/citeseerx. We ran our implementation on the entire CiteSeer^x with the setting that had the highest performance on the random sample of 1000 documents (i.e., title only, 3-gram query, $\theta = 0.7$, Jaccard). The total number of matches found between CiteSeer^x and DBLP is 630,351 and the total time taken to finish is 184,749 seconds. The entries in the newly constructed dataset are xml files that contain articles' metadata. Regular updates will be done to integrate new articles crawled and indexed by CiteSeer^x.

In addition to the newly constructed dataset, we will provide the xml files for the entire CiteSeer^x data repository and let the researchers decide what setting they prefer to use for the dataset generation, using our Java implementation.

References

1. Giles, C.L., Bollacker, K., Lawrence, S.: Citeseer: An automatic citation indexing system. In: Digital Libraries '98. (1998) 89–98
2. Lu, Q., Getoor, L.: Link-based classification. In: ICML. (2003)
3. Peng, F., Schuurmans, D.: Combining naive bayes and n-gram language models for text classification. In: ECIR, Springer-Verlag (2003) 335–350
4. Caragea, C., Silvescu, A., Kataria, S., Caragea, D., Mitra, P.: Classifying scientific publications using abstract features. In: SARA. (2011)
5. Sen, P., Namata, G.M., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine **29**(3) (2008) 93–106

[4] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.239.1803
[5] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.63.1066
[6] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.169.7994
[7] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.267.8099

6. Zhou, D., Zhu, S., Yu, K., Song, X., Tseng, B.L., Zha, H., Giles, C.L.: Learning multiple graphs for document recommendations. In: Proc. of WWW '08. (2008)
7. Caragea, C., Silvescu, A., Mitra, P., Giles, C.L.: Can't see the forest for the trees? a citation recommendation system. In: Proceedings of JCDL '13. (2013)
8. Huang, W., Kataria, S., Caragea, C., Mitra, P., Giles, C.L., Rokach, L.: Recommending citations: translating papers into references. In: CIKM. (2012)
9. Küçüktunç, O., Saule, E., Kaya, K., Çatalyürek, Ü.V.: Diversified recommendation on graphs: pitfalls, measures, and algorithms. In: WWW. (2013)
10. Nallapati, R.M., Ahmed, A., Xing, E.P., Cohen, W.W.: Joint latent topic models for text and citations. In: Proceedings of KDD '08. (2008)
11. Treeratpituk, P., Giles, C.L.: Disambiguating authors in academic publications using random forests. In: Proc. of JCDL. JCDL '09 (2009)
12. Gollapalli, S.D., Mitra, P., Giles, C.L.: Similar researcher search in academic environments. In: JCDL. (2012)
13. Chen, H.H., Gou, L., Zhang, X., Giles, C.L.: Collabseer: a search engine for collaboration discovery. In: Proceedings of JCDL '11. (2011)
14. Kan, M.Y.: Slideseer: a digital library of aligned document and presentation pairs. In: Proceedings of JCDL '07. (2007)
15. Kataria, S., Mitra, P., Caragea, C., Giles, C.L.: Context sensitive topic models for author influence in document networks. In: Proceedings of IJCAI'11. (2011)
16. Rosen-Zvi, M., Griffiths, T., Steyvers, M., Smyth, P.: The author-topic model for authors and documents. In: Proceedings of UAI '04. (2004)
17. Bhattacharya, I., Getoor, L.: A latent dirichlet model for unsupervised entity resolution. In: SDM. (2006)
18. Han, H., Giles, C.L., Manavoglu, E., Zha, H., Zhang, Z., Fox, E.A.: Automatic document metadata extraction using support vector machines. In: JCDL. (2003)
19. Councill, I.G., Giles, C.L., yen Kan, M.: Parscit: An open-source crf reference string parsing package. In: Intl. Language Resources and Evaluation. (2008)
20. Bhattacharya, I., Getoor, L.: Iterative record linkage for cleaning and integration. In: DMKD. (2004)
21. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association **64** (1969) 1183–1210
22. Rahm, E., Do, H.H.: Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin **23** (2000) 2000
23. Tejada, S., Knoblock, C.A., Minton, S.: Learning object identification rules for information integration. Journal Information Systems (2001)
24. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of KDD '03. (2003)
25. Cohen, W.W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: Proceedings of KDD '02. (2002)
26. Winkler, W.E.: Methods for record linkage and bayesian networks. Technical report, Statistical Research Div., U.S. Bureau of the Census (2002)
27. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: IJCAI. (2003) 73–78
28. Pasula, H., Marthi, B., Milch, B., Russell, S., Shpitser, I.: Identity uncertainty and citation matching. In: NIPS, MIT Press (2003)
29. McCallum, A., Wellner, B.: Toward conditional models of identity uncertainty with application to proper noun coreference. In: IIWeb. (2003)
30. Manku, G.S., Jain, A., Das Sarma, A.: Detecting near-duplicates for web crawling. In: Proc. of WWW '07. (2007)