PENNSTATE

1855

ODU

Cleansing Noisy and Heterogeneous Metadata for Record Linking Across Scholarly Big Datasets

Athar Sefid, Jian Wu,

Allen C. Ge, Jing Zhao, Lu Liu, Cornelia Caragea, Prasenjit Mitra, C. Lee Giles

Pennsylvania State University

Old Dominion University

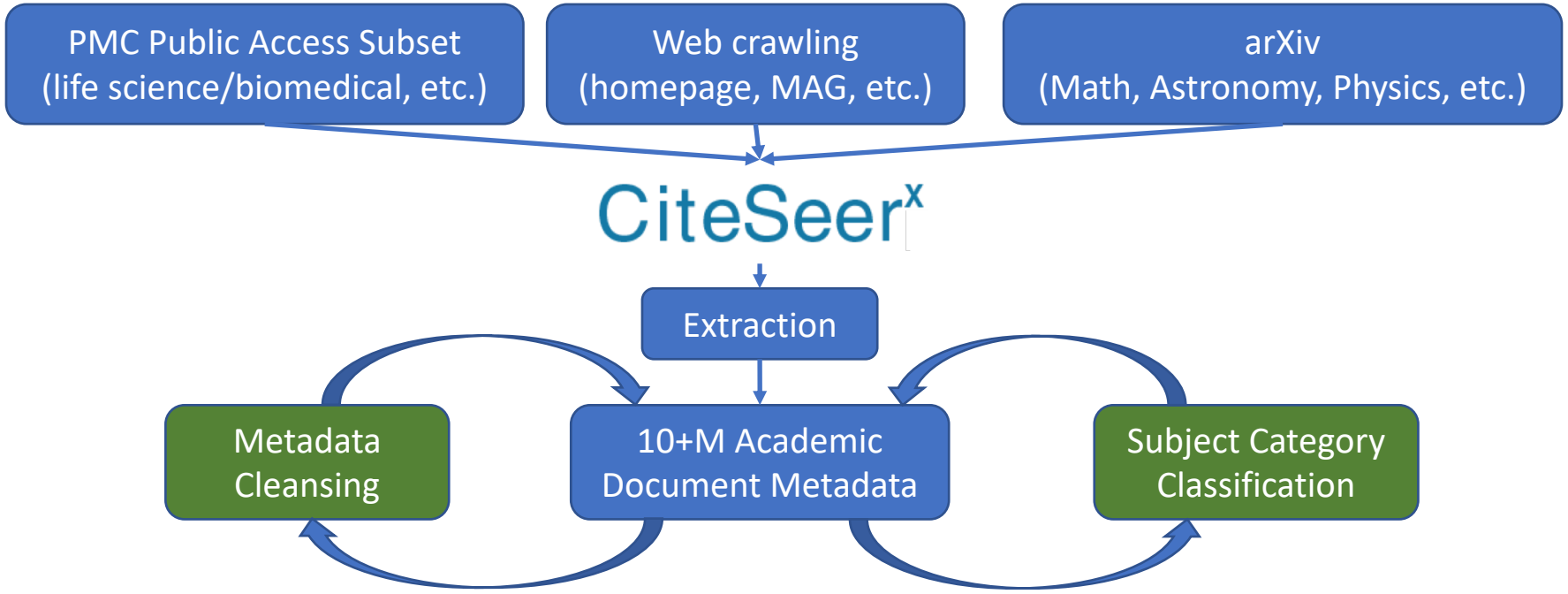
Problem: Erroneous Metadata

- Automatically extracted metadata from scholarly PDF files is usually noisy.
  - **Incomplete fields**
  - **Erroneous values**

Automatic metadata extraction

How to fix the errors?

- Step 1: match noisy dataset to the reference datasets
  - Reference datasets: high accuracy, Input by human
  - Matching must be done with high **precision** to avoid **”false correction”**
- Step 2: replace metadata of noisy data set

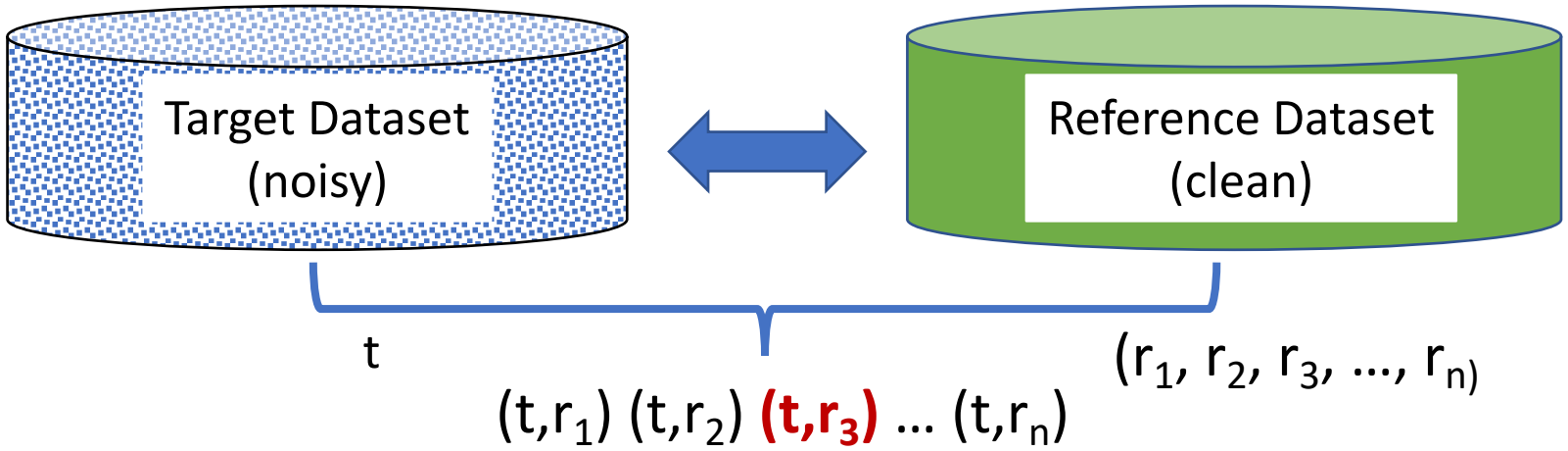


Matching Mechanisms

1. Match by title + authors: values may be wrong, cannot be used as a unique identifier
2. Unique identifier (DOI, journal, volume, page): not always available
3. Look at all available fields
    1. Header: Title, Authors, Venue, Year, Abstract, etc.
    2. Citations
    3. Full text?

Entity Matching Problem

- Target corpus T: containing n paper entities t_i; $1 < i < n$
- Reference corpus R: containing m paper entities r_j; $1 < j < m$
- Goal is to find a set $\mathrm{M} = \{(\mathrm{t}, \mathrm{r}) | \mathrm{t} = \mathrm{r}, \mathrm{t} \in \mathrm{T}, \mathrm{r} \in R\}$
- Reduced to a binary classification problem



Challenges

- Noisy data could be in any field – Examples:

correct title: A New Metric for Banking Integration in Europe
incorrect title: A New Metric for Banking Integration in Europe 1

correct venue: Mol Cancer Res
incorrect venue: Deregulated expression of the PER1, PER2 and PER3 genes in

- Datasets can be big: pairwise comparison does not scale

Target Dataset	
CiteSeerX	10 million

Reference Datasets	
Web of Science	45 million
DBLP	4 million
Medline	24 million

Binary classification

- Header: what information can we use?
  - Title: single string, may contain non-ASCII characters
  - Authors: multiple names, multiple components, may contain non-ASCII characters
  - Venue: single string, multiple forms (journal, conference, full name-acronym mapping) – need external database
  - Year: integer
  - Abstract: long text, may contain non-ASCII characters

Binary classification

- Header: what information can we use?
  - Title: single string, may contain non-ASCII characters
  - Authors: multiple names, multiple components, may contain non-ASCII characters
  - Venue: single string, multiple forms (journal, conference, full name-acronym mapping) – need external database
  - Year: integer
  - Abstract: long text, may contain non-ASCII characters
- HMM: Header Matching Model

HMM classifier features

- Title similarity
  - Normalized → Simhashed → Levenshtein distance
  - Jaccard similarity of normalized titles
- Abstract similarity
  - Simhashed →Levenshtein distance
  - Jaccard similarity of original abstracts
- Year similarity
  - Absolute difference

Author similarity

- Full name similarity: lmf
  - Jane C. Huck vs. J. Huck
  - First initial-match: $f = 1$
  - Middle initial—not match: $m = 0$
  - Last name-match: $l = 1$
  - $lmf = 101$ – decimal value = 5
  - Applied to the first and last author
- Last name similarity: 0/1/2
  - 0: not match
  - 1: at least one unavailable
  - 2: both available and match
  - Applied to the first and last author
- Author list similarity: Jaccard
  - Only look at last names
  - Paper 1: Wu, Sefid, Huck
  - Paper 2: Wu, Sefid, Giles
  - $\text{Jaccard similarity} = \frac{2}{4}$

Jian Wu: jwu@cs.odu.edu; Athar Sefid: azs5955@psu.edu

Algorithm 1 Query Builder

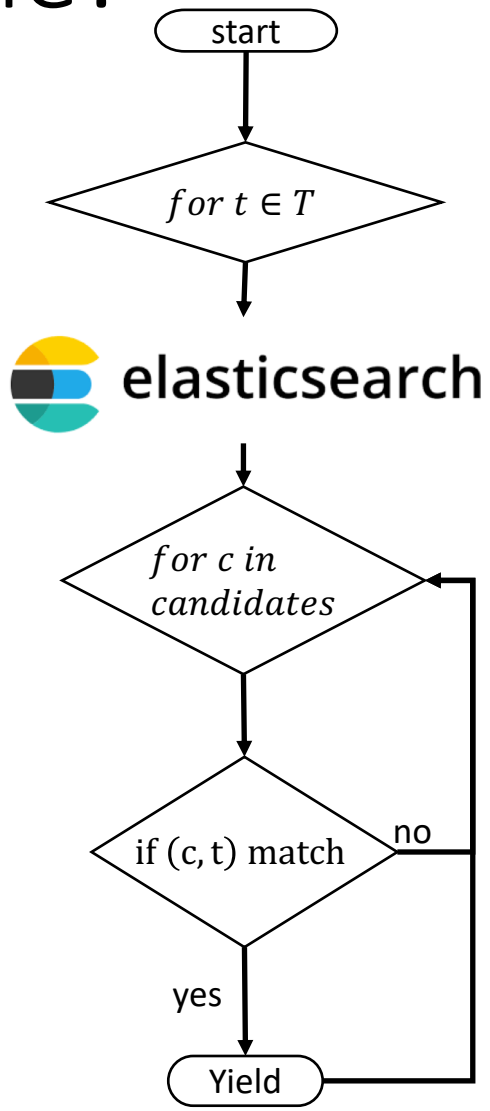
```
1:  function QUERY(title, lastName, year)
2:      if title ≠ Null and title.length > 20 then
3:          query ← title
4:      else if lastName ≠ Null and year ≠ Null then
5:          query ← lastName and year
6:      else if lastName ≠ Null then
7:          query ← lastName
8:      end if
9:   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

Algorithm 2 Header Matching Model

```
1: function HMM()
2:     T ← target corpus
3:     R ← reference corpus
4:     index_R ← index of reference corpus
5:     matchList ←Ø
6:     for t ∈ T do
7:         Q ← Query(t.title, t.firstLastName, t.year)
8:         Candidates ← query Q to index_R
9:         for c ∈ Candidates do
10:            prediction←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
```

Pair-wise comparison does not scale!

- Solution: use search engines to reduce search space
  - Reference metadata is indexed (by Elastic Search)
  - titles, authors, abstracts, and year
- Query title and authors of target papers against the index
  - Top 10 papers are retrieved – candidate set



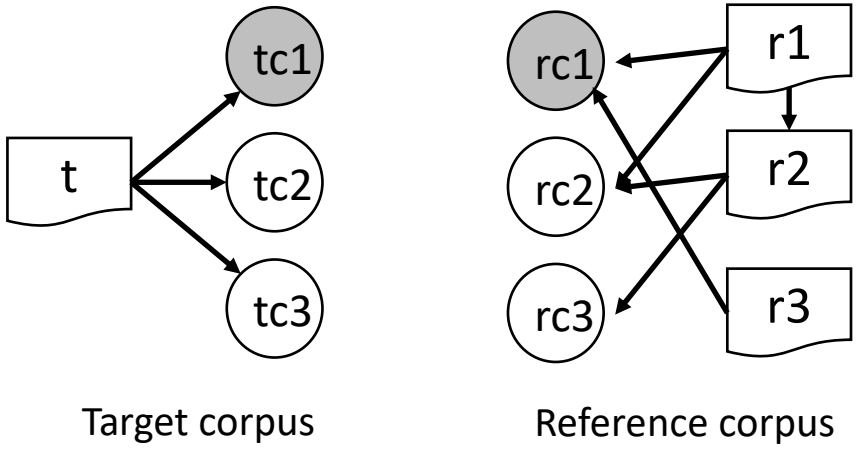
Limitation of the Header Matching Model

- Title may be noisy – a serious problem
- Solution: use citations (references)
- Assumption: different papers have different reference lists

CMM citation matching model

- References must be available.
- Algorithm:

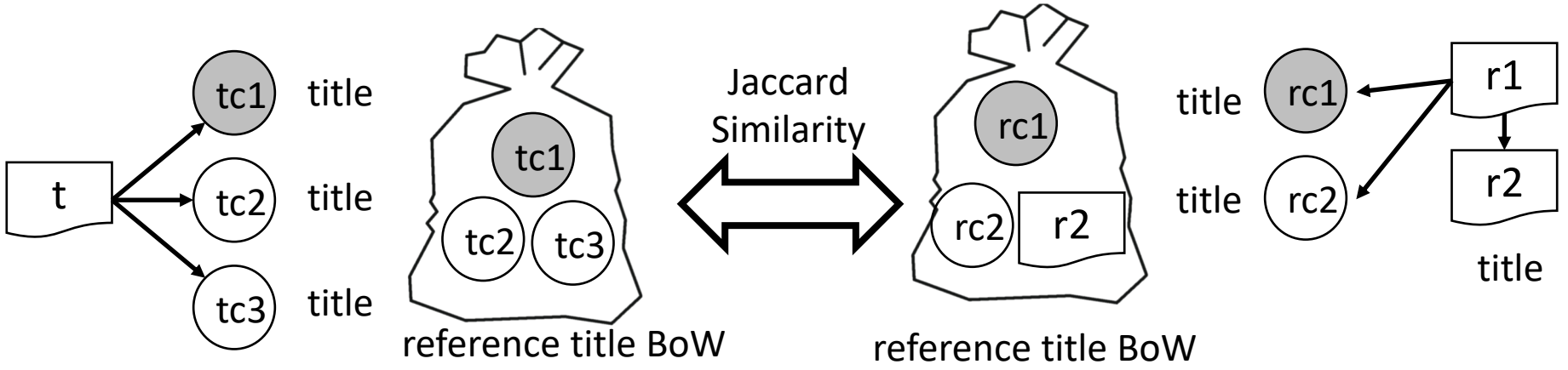
1. All citations in the reference corpus are indexed
2. For each target paper t, its citation records tc_i are retrieved



CMM citation matching model

- References must be available.
- Algorithm:

1. All citations in the reference corpus are indexed
2. For each target paper t, its citation records tc_i are retrieved



Algorithm 3 Citation Matching Model

```
function CMM()
    T ← target corpus
    R ← reference corpus
    matchList ←Ø
    CitationIndex_R ← citations index of reference corpus
    for t ∈ T do
        t_citations ← citations of t.
        for tc_i ∈ t_citations do
            Q ← Query (tc_i.title,tc_i.firstLastName, tc_i.year)
            results ← query Q to CitationIndex_R
            for rc_j ∈ results do
                prediction←Model.predict (tc_i, rc_j)
                if prediction=1 then                    ▷ citations match
                    r_k ← paper that cites rc_j
                    r.title ← Simhash (r_k.title)
                    t.title ← Simhash (t.title)
                    title_dist = lev (t.title, r.title)
                    if title_dist < θ_title then
                        matchList.add (t, r_k)
                        break
                    end if
                    BoW_r ← BoW (r_k.referenceTitles)
                    BoW_t ← BoW (t.referenceTitles)
                    sim ← Jaccard(BoW_ti, BoW_ri)
                    if sim > θ_ref then
                        matchList.add (t, r_k)
                        break
```

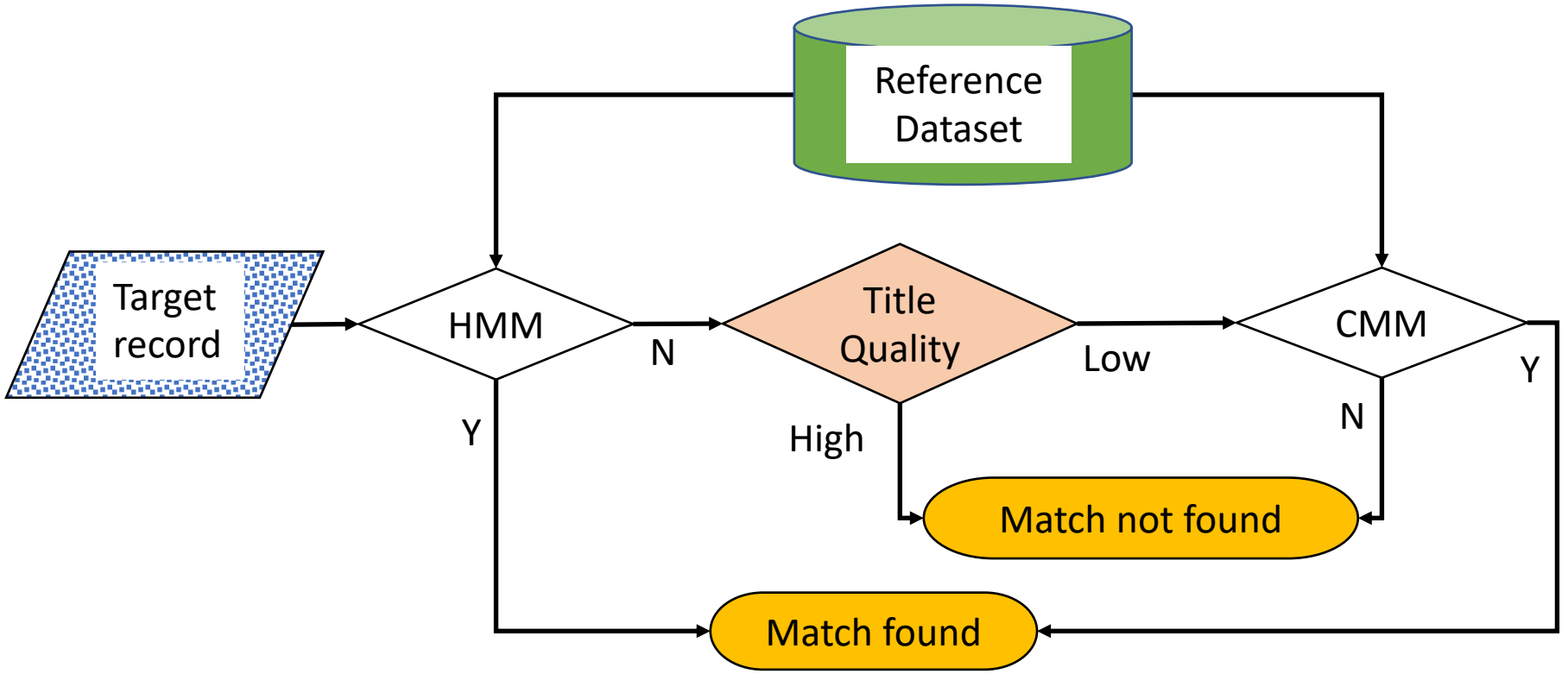
Table 1: Features used to train TEM.

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 $(\mathrm{DF}(w)), w \notin \mathbb{S}$ #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.

IMM: Integrate HMM with CMM



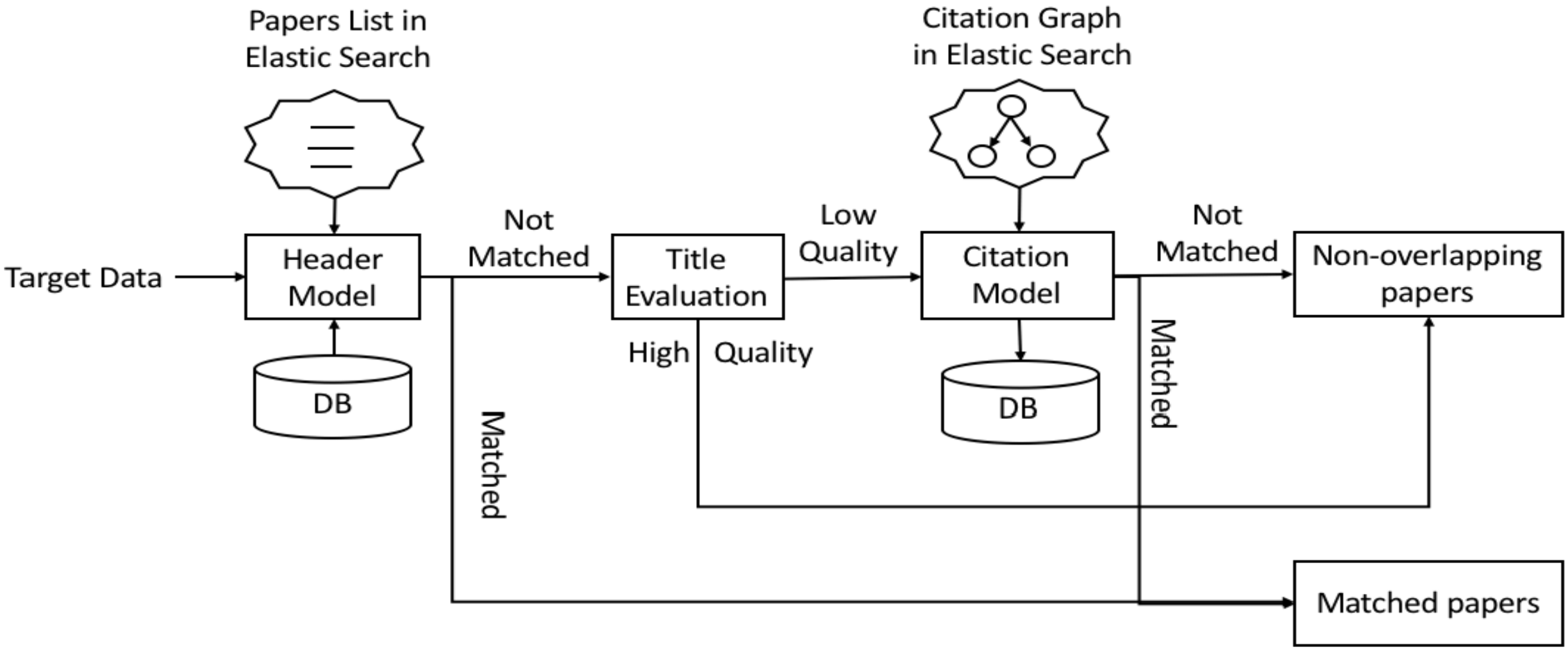
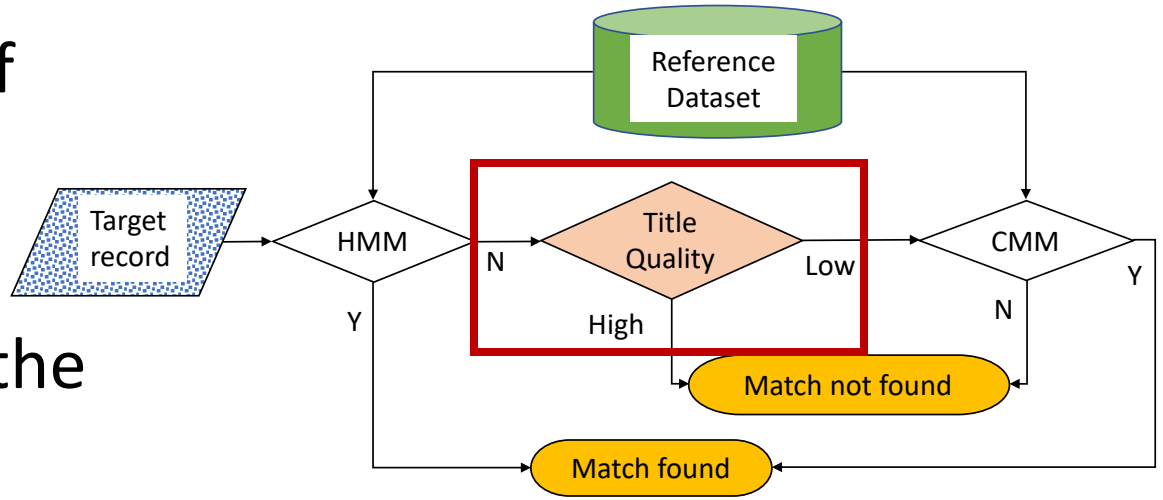


Figure 1: The pipeline of IMM.

TEM title evaluation model

- A simple supervised model to provide a quantitative evaluation of the title quality.
- Input: a title string
- Output: a probability of how likely the input string looks like a paper title.



HMM Alone

- Ground truth: about 700 matching pairs and the same number of unmatched but most similar pairs
- HMM alone:
  - Most informative features are titles and the first authors based on information gain(IG)
  - The SVM achieves the highest F1-measure. But, Random Forest has a comparable F1-measure with 30% reduced test time,
  - We employ Random Forest.

10-fold cross validation 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

HMM vs. CMM vs. IMM

Evaluation Data: CiteSeerX-WoS dataset

Higher title quality

$\theta < \theta_{tq}$	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

- CMM model has high remarkable precision but poor recall.
- IMM achieves both high recall and precision.
  - CMM tend to be more useful when the title quality is low
  - The integrated model significantly increases the overall performance, especially for papers with low quality titles.

Table 3: The CMM performance with different θ_{ref} and θ_{title}.

θ_{ref}	θ_{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

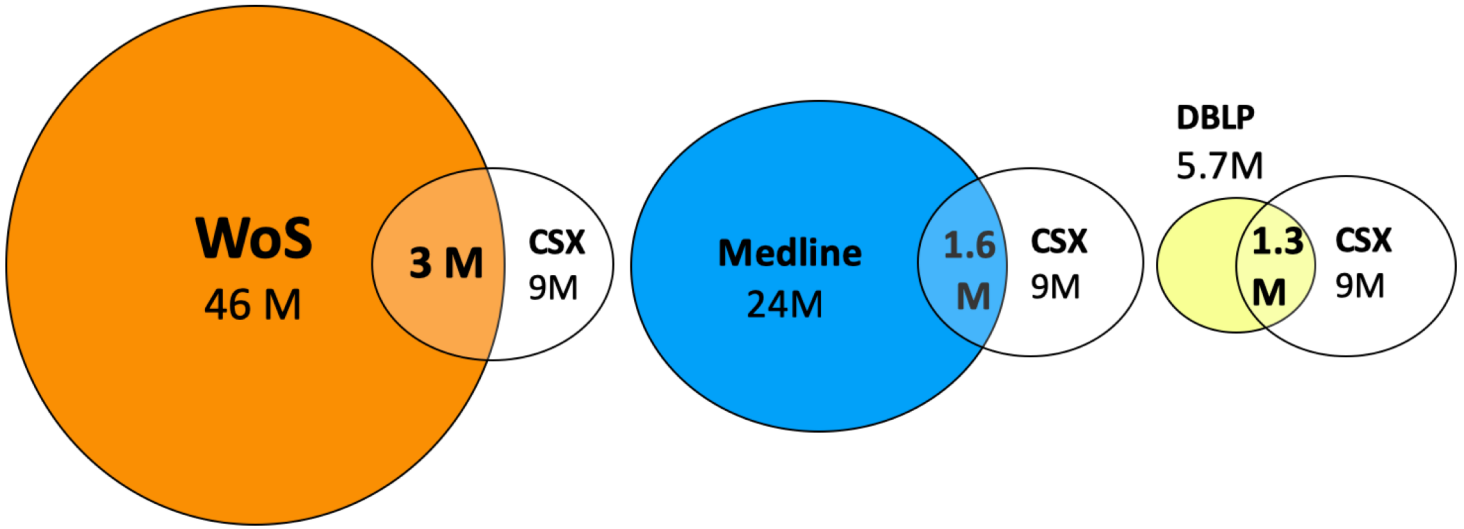
Table 4: Comparisons of HMM, CMM, and IMM performances using the CiteSeerX-WoS dataset with different title quality thresholds. T/s stands for testing time in seconds.

$\theta < \theta_{tq}$	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

Error Analysis

- Why CMM has how recall?
  1. Papers with very few references:
     - 1 million CiteSeerX papers have less than 5 references (parsing error?)
  2. Citations without titles:
     1. 17% of citation records in WoS have null titles
     2. 8.4% of citations in CiteSeerX have null titles
- Why CMM model is slow?
  1. Huge number of citations to indexed: 906 million (hardware can help?)
  2. Candidate set for each CiteSeerX citation could be huge for highly-cited papers

CiteSeerX overlap with reference sets



In total 4.2 million CiteSeerX documents can be cleaned.

Compare with previous work

- Previous studies used only metadata in the header of scholarly articles for paper entity linking.
- Quality of a header is not always good. Hence, we investigated leveraging both header and citation information to match paper entities
- Compared with the previous IR-based method, HMM improves precision by at least 13% and F1 by about 3% for papers with low quality titles.
- The best results by IMM:
  - F1= 0.992
  - Precision=0.984