

Resolving citations in a paper repository

Sunita Sarawagi Sumana Srinivasan
V.G.Vinod Vydiswaran Kapil Bhudhia
I.I.T. Bombay
{sunita,sumana,vgvinodv,kapil}@it.iitb.ac.in

ABSTRACT

In this paper, we describe our process of creating a citation graph from a given repository of physics publications in L^AT_EX format. The task involved a series of information extraction, data cleaning, matching and ranking steps. This paper describes the challenges we faced along the way and the issues involved in resolving them.

1. INTRODUCTION

Our input was a collection of about 35,000 L^AT_EX formatted papers from the hep-ph portion of arXiv, an online repository of research publications from several related areas of physics. The goal was to re-create from these papers, a citation graph where nodes represent papers and an edge from paper i to paper j denotes that paper i cites j . In this paper, we present our experiences in accomplishing this task. Our approach can be partitioned into the following five steps:

1. **Information Extraction**, where we extracted structured fields like citations, paper title, and author names at the first level, and finer-grained fields like lastnames from citations and paper headers, and year of publications at the second level.
2. **Data Cleaning**, where the extracted fields are cleaned of irrelevant L^AT_EX commands and stop-words.
3. **Candidate Generation**, where we use an inverted index on paper headers and query them using fields in citations to generate candidate (citation, paper) pairs.
4. **Additional Filtering**, where we apply additional filters derived from date, names, and frequency to prune the set of (citation, paper) candidates returned from the previous stage.
5. **Final Ranking**, where we rank candidate pairs based on word similarity and select the top match.

We elaborate each of these steps in the rest of the paper.

2. INFORMATION EXTRACTION

In this section, we describe the problems encountered during the extraction of citations and paper headers.

2.1 Citation Extraction

This involved identifying the beginning and end of the citation section and extracting individual citations from the part in between. Majority of the papers had citations bracketed within `\begin{bibliography} ... \end{bibliography}`. Most of the remaining started the citation section with the special keyword “References” appearing in different formats, but without any marker to denote the end of the section. In such cases, we identified a set of keywords like `\lref`, `\nref`, and `\ref` to mark the beginning of each citation entry.

The extraction of individual citations was not easy when commands other than `\bibitem` was used to start a citation. We used the following algorithm to automatically discover the variety of commands used to mark the start of a citation. We first counted the frequency of occurrence of the first word in a line following a `\` in the citation section, ignoring standard L^AT_EX font words like `\bf` and `\it`. If `\bibitem` appeared in this list of words with a frequency greater than 2, we selected that. Otherwise, we chose the most frequent word as the separator. This revealed an interesting range of latex commands like `\bib`, `\bi`, `\nref`.

Another complexity was due to multiple citations within one citation entry. This practice, uncommon in computer science publications, was found to be fairly common in physics publications. Fortunately, the extraction of individual citations turned out to be simple, since they were separated by semicolons and this delimiter was rarely used in non-compound citations.

2.2 Name and date extraction

Citations typically comprised of a list of authors, followed by the name and year of the journal and an article number. Surprisingly, most citations did not state the title of the cited paper. We extracted two kinds of structured fields from each citation: lastnames of all authors and year of publication, as shown in a typical citation below:

```
J.A.O11er, E.Oset, Nucl. Phys. A620 (1997)438
Extracted fields...Lastnames:O11er,Oset Year:1997
```

For this task, we wrote hand tuned regular expressions that exploited single letter author initials before and after the words denoting lastnames. For date extraction, we applied a succession of rules in the following order: numbers between years 1950 and 2000 within “()”, without “()”, and two digits between 50 and 99 within “()”.

2.3 Title and Author extraction

We next analyzed paper headers to identify the section of

the paper that contained the author and title. Then, from these sections we extracted author lastnames. The first part was trivial for about half of the papers that had both `\author{}` and `\title{}`. For the rest, we tried to detect tags that denote the beginning of the body of the paper, such as `\begin{Abstract}` and `\begin{Introduction}` and their variants. We extracted the part of the paper that appeared before these as the likely author-title section. In those cases where even this information was absent, we used the words from the top 15% of the papers.

The author-title section is later indexed (Section 4) and matched to citations. Therefore, including spurious words in this section does not hurt quality provided these extra words do not include names other than the authors of this paper. Many papers contained citations at the beginning and/or body of the paper. When extracting the author-title section it is important to avoid these. We ensure this by running the author-title section extractor only after removing all citations from the paper.

We next identified last names from these headers. The task here was significantly more difficult than in citations, where lastnames had a lot of uniformity in their location and formatting. We therefore employed the following trick. We created a dictionary of lastnames by collating the names obtained from all citations. These were very likely correct. We then wrote a hand-tuned set of rules that combined various sources of evidence from typical name formats, density of occurrence, and match in the dictionary to extract lastnames of all authors of the paper.

3. DATA CLEANING

Some of the data cleaning tasks that were performed at various stages, particularly before the next indexing phase, were:

- The papers and citation entries were stripped of \LaTeX commands so as to avoid spurious hits.
- Common words such as “Phys.,” “Math.,” “Intl.,” “Proceedings”, etc. were filtered out from the citation string so as to reduce the number of hits pertaining to these stop-words.
- Numerical tokens and special symbols were filtered out of the citation.
- Some of the citations that referred to a collaboration of authors instead of listing individual author names were removed from the citations list. The citations containing the words “COLLAB” or “COLLABORATION” were filtered out.

4. GENERATION OF CANDIDATE (CITATION, PAPER) PAIRS

We created an inverted index of words from the extracted author-title section of the paper header. For this purpose we used Lucene¹, a full-featured text search engine. For each citation, we generated a list of likely papers it refers to by querying the index using words in the citation. The query is an “OR” of all words from the *cleaned* citation but makes it *mandatory* to include all lastnames that we had extracted. An earlier version without the mandatory lastnames gave

¹<http://jakarta.apache.org/lucene/docs/index.html>

unmanageably large and poor quality results. We select a maximum of top 15 hits returned by Lucene’s ranking algorithm and also record the total number of hits that matched the query.

5. ADDITIONAL FILTERING

We pruned any (citation c , paper p) pairs that did not satisfy any of the following three filters:

1. **Date Filter:** Each paper’s earliest year of publication is estimated to be the maximum of the year extracted from each citation in the paper. If year of a paper p is less than the year of a citation c , we prune that (c, p) pair.
2. **Name Filter:** Here we make use of lastnames extracted for each paper. Any (c, p) pair where all of p ’s lastnames do not appear in c ’s list of last names and in the same order, are removed.
3. **Frequency Filter:** We filtered out all citations that generated a very large number of hits in the above index search. After pruning the dataset, we found that there were few papers that were authored by collaborations. These papers contained a long list of author names, and hence appeared in top 15 hits of many citations. These papers were also filtered out.

6. FINAL RANKING

Our initial goal was to compute several elaborate features from various metrics like TF-IDF scores and position of respective terms in the citations and papers, and do this separately for the title, author, and content part. Our intention was to input these features into a classifier, where examples of matched pairs are created using active learning. However, computing this information for the entire citation corpus was found to be prohibitively time consuming and could not be completed in the limited time we had. We therefore resorted to the following simpler scheme.

For each (c, p) pair, we measured the frequency of occurrence of each non-stopword in p ’s title in the body of the paper where c appears, and calculate a TF-IDF score. For each c , we then use this score to sort all papers with which it is matched. If the difference between the top two scores was substantial, then the top hit was selected as the valid link in the citation graph. On the other hand, if the scores of the top hits were too close, then none of the citations were predicted as valid so as to avoid labeling false citation links as edges in the graph.

7. CONCLUSION

Due to lack of tools in the area, we developed highly specific perl scripts to perform regular expression searches along with pattern matching code written in JAVA. None of these can be reused in a different but similar scenario. We felt the need for a pattern specification and learning tool that could perhaps drastically cut down on the time taken to perform these tedious tasks. However, a limited time competition prevented us from experimenting with these.