École Polytechnique Fédérale de Lausanne

Semester Project

A Semi-Supervised Approach to Citation Matching in the Humanities

Author:
Tao Sun

Supervisor:
Dr. Matteo Romanello

Professor:
Prof. Frédéric Kaplanin

Digital Humanities Laboratory
EPFL

June 8, 2018

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Abstract

Citation matching in the humanities publication is not an easy task. This project adopts a step-by-step approach to match references in the Linked Books dataset. Due to the large scale of data available, the search space is required to be reduced to efferently find the match. The approach starts with matching partial references with full references in one document based on some general rules. Secondly, with the help of hash functions, the full references are grouped together into a global candidate group. Finally, semi-supervised learning method is applied to address the problem of lacking ground truth and classify within the candidate group whether two citations point to the same document. Manually annotating the global candidate group is needed for the evaluation of this approach. The source codes are published on GitHub.

Acknowledgements

A special gratitude I give to my direct supervisor, Dr. Matteo Romanello for the continuous support and generous advices of the semester project. My sincere thanks also goes to Prof. Frédéric Kaplan for offering me the opportunity to work in the Digital Humanities Laboratory for this project. Also, I thank Dr. Maud Ehrmann and other DHLab members with whose company I have spent a happy semester in the laboratory. Last but not the least, I would like to thank Ms. Silvia Ferronato who annotated lots of data for me which is essential for completing the project.

Contents

1 Introduction

1.1 Problem Description

Citation matching (i.e. linking citation strings referencing the same paper) in humanities publications is not a easy task. For humanities research, some very old books and articles are always of great importance, which might have laid the foundation of the field. However, unlike in scientific papers where the citations are well structured, various citation styles have appeared throughout the history in the humanities. How to do citation matching in those old publications is the problem that needs to be solved.

The aim of this project is to develop an approach which can efficiently group together citations pointing to the same documents within the **Linked Books**[1] dataset, which contains a great number of citations in humanities publication. To be more precise, once a citation is chosen, using this approach is able to find all others representing the same document in the dataset.

Given the dataset, challenges are foreseeable:

1. Roughly 4 million citations are available in the dataset, which makes the pairwise comparison unrealistic and impracticable due to quadratic time and space complexity.
2. There is limited annotated data to support an entire supervised learning approach. We don't know for sure whether two citations point to the same document or not.
3. We have to disambiguate citations without any external knowledge base. In the Linked Books project, only those citations that could be linked to the catalogue of Italian libraries, which contains some 15 million records, have been disambiguated so far.

1.2 Linked Books Dataset

The Linked Books dataset contains roughly 4,000,000 citations from a broad corpus of books and journal articles on the history of Venice. The citations have already been parsed into different fields [1]. Within the dataset, there are some subsets available:

References contains the information of each field of each citation, e.g. author, title, abbreviation, publisher, year, the document it comes from, etc.

Disambiguation contains the pair information of one reference and the document it points to, which is manually checked.

[1] https://dhlab.epfl.ch/page-127959-en.html

Articles contains the meta data of articles, including title, author, year, journal information, page number in the journal, etc.

Books contains the meta data of books, including title, author, year, ISBN, publication information, etc.

1.3 Citation Matching

Citation matching has been studied for a long time [2], [3]. It is sometimes divided into two phases: segmentation and entity resolution. The former is to parse a reference string into small fields, like author, title, year, etc. The latter aims to cluster citation representing the same document.

Given the exponentially increasing data, Fedoryszak et al. recently introduced an efficient blocking method into the citation matching [4]. The general idea behind it is *Divide and Conquer*–generating key-value pairs to group together references and process separately in a Map-Reduce way. They considered a citation as a duplicate of referenced document and thus, citation matching problem as deduplication task. In this sense, they call the key as hash and the generating method as hash function which normally is used for detecting duplicated records in a large file.

For instance, for the following reference:

Morassi, Novità e precisazioni sul Tiepolo in « Le Arti », 1942, p. 91.

they would generate keys or hashes, such as morassi#novita#precisazioni, morassi# precisazioni#tiepolo, morassi#1942, and then group together citations sharing the same hashes.

As mentioned above, direct pairwise comparison is impossible in the Linked Books database, thus, such blocking method is useful for the goal of our project: performing pairwise similarity calculation within small candidate group instead of the whole dataset.

1.4 Approach In A Nutshell

Our approach builds upon the idea of [4] as we start from one single document locally and then going to the whole dataset or globally:

Local Clustering To begin with, we do the citation matching job within one document. The matching algorithm is a combination of rule-based matching and pairwise comparison. As in local level the number of citation is small, the time and space complexity of such comparison is thus bearable. The aim of this step is to locally cluster together references to the same publication.

Hash Matching As soon as we get the local cluster, we fuzzily group those clusters together to form a candidate group. Based on the hash idea, several keys for each cluster are generated for matching. After this step, we will have the candidate group in which references are from different documents but share the same hashes.

Global Clustering This step can be regarded as a typical binary classification problem, which classify whether a pair of citations are pointing to the same document or not. Features are defined as the similarity vector between two citations. Due to the lack of ground truth in global-level matching, a semi-supervised learning method is implemented to build the classifier.

We will introduce the step-by-step approach in details with examples selected from our database in Chapter 2. The evaluation method and result of each step will be described and discussed in Chapter 3.

2 Approach

2.1 Local Clustering

Before exploring the whole database to matching citations, we start our approach inside one publication, an article or a book, locally. When citing one document several times in one publication, the author usually do not repeat using one string containing all information of it, a **full citation**, instead, s/he would use the **partial citation** to make the context more concise.

The main task of local clustering is how to deal with the presence of partial citations. Although along the history people cited in different styles, there still exist some general rules of how such partial citation was formed. Thus, a rule-based matching method can be developed. Meanwhile, other situations, like repeatedly using full citations, also need consideration.

2.1.1 Partial Citation

Based on a preliminary phase of data analysis in the dataset and sources from the Internet, some general rules of partial references are observed and summarized as following:

1. idem / id. / eadem / ead.
   - e.g. *Eadem, Storia della ceramica a Venezia, Firenze 1981, pp. 363-405.*
   - Meaning its author is the same as in the immediately preceding reference
   - Always in the beginning of a citation
   - Actually not a "true" partial citation only author's name omitted
2. ibidem / ibid. / supra
   - e.g. *ibid., voi. i, pp. 167-169.*
   - e.g. *Tarabotti, ibid. pp. 202-203.*
   - Meaning it refers to the same publication as in the immediately preceding reference
   - Often in the beginning, sometimes after the author's name
3. ivi.
   - e.g. *ivi col. 212, 24 mag. 1533.*
   - Meaning it refers to the same publication as in the immediately preceding reference

- Often in the beginning

4. op. cit. / op. ctt. / cit.
    - e.g. *E. Ruffoni, Parole pronunciate ecc., op. cit.*
    - e.g. *Quevedo, op. cit.*
    - Meaning it has been cited before
    - In the beginning or after author or after title
5. Using ellipsis
    - e.g. *A. Alverà Bortolotto, Maiolica..., pp. 90-93.*
    - Instead of writing a complete title, using only the first few words followed by "..."

2.1.2 Rule-Based Matching

As shown above, the partial reference always has some kind of connection with the one(s) before it, thus, the order of citations is of importance. In our database, we have the information of which page one citation is and also the order in the page, which enable us to determine the order of citations.

To match partial citations with the full one, several iterations are required to handle different situations, especially different abbreviations.

1. Iterate through all references for the first time:
    (a) Fill the missing author names caused by idem, eadem, etc.
    (b) If "op. cit.", "...", etc.: iterate through all references before from bottom to top to find the best and closest match.
    (c) If "ibidem", "ivi", etc.: match it with the one right before.
2. Iterate through all unmatched references and find the best match pairs

2.2 Hash Matching

2.2.1 Hash Description

As described in 1.4, after we finish local clustering, we need to form a big candidate group using only the full references from all documents available. To generate such meaningful but small enough group, we give each reference a key, which is the combination of several tokens in the reference separated by the hash sign "#". We name such key as "**hash**" and the corresponding generating function as "**hash function**".

One reference will have multiple hashes and other references sharing the same hash will go into the same candidate group. In the project, once one **seed reference** is chosen, we can match all other references sharing the same hash with the seed. We call such process as the references are "**returned by hash**".

2.2.2 Hash Functions

Designing a good hash functions is not a trivial task. Considering the citations in our dataset are already parsed, I choose and modify several hash functions described in [4] to best fit our data. A normalization process and a detailed description of hash functions are presented in the rest of this section.

To give a general idea, the following citation will be used as an example:

Luciano Monzali, Italiani di Dalmazia 1914-1924 , Roma, Le Lettere, 2007, pp. 339-436.

Normalization

- Use lowercase letters
- Remove diacritics and punctuation marks
- Ignore stopwords in all languages[1]
- Keep only tokens longer than 2 characters

Baseline A baseline is served to compare the functionality of different hash functions. Here, I choose to use all the tokens in a citation as hashes. That's also the strategy of manual citation matching: pasting the whole reference string into database and select the best matching among results.

Bigrams of Title We generate all the bigrams in the title as hashes, e.g. "italiani#dalmazia", "dalmazia#1914" for the example reference.

Author - Bigrams of Title Intuitively, the name of author is another choice for hash function. Instead of using name alone, which contains little information, we combine it with bigrams of title, e.g. "luciano#italiani#dalmazia", "monzali#italiani#dalmazia".

Author - Year This function generates hashes in the form "name#year". There are two kinds of this function: one strict version with the exact given year and another blurred version using not only the given one, but also the previous and the next ones to deal with the possible error in the year. For example, "luciano#2007" along with "luciano#2006" and "luciano#2008".

Author - Year - Pages This function generates hashes in the form "name#year#page1#page2#...", e.g. "luciano#2007#339#436". Also, a blurred version for both year and page number is designed.

Author - Year - Num^3 This function takes author with year and top three of other numbers as hash, and the format is "name#year#num$_1$#...#num$_n$", $n \in \{1, 2, 3\}$. "num$_1$..., num$_n$" are sorted; they may be part of title, or the volume number, or just page information.

[1]The stopwords are provided by Natural Language Toolkit (NLTK) [5].

2.3 Global Clustering

When we have the global candidate group, it is now the show time of machine learning. The problem now has been simplified as a binary classification problem. To address the problem, the features of the candidate pair need to be chosen and defined meaningfully. To train and evaluate the classifier, a certain number of labelled data are needed to serve as training and evaluation (or validation) data. However, as described in 1.1, we lack the ground truth of matched citations among documents. That's why we choose to use a semi-supervised method to build the classifier. In this project, a simple logistic regression model is implemented.

2.3.1 Feature Similarity

To define the similarity between different fields is an old but challenging task. There are various ways available. In this project, we mainly implemented some methods mentioned in [6] as following.

Definition Let $ngrams(s)$ be a multiset of ngrams from a string s. We define a *ngram similarity* between strings s and t as

$$sim_n(s,t) = 2\frac{|ngrams(s) \cap ngrams(t)|}{|ngrams(s) \cup ngrams(t)|}$$

Token similarity is defined in a similar manner. Let $tokens(s)$ be a multiset of tokens from a string s. Then

$$sim_{token}(s,t) = 2\frac{|tokens(s) \cap tokens(t)|}{|tokens(s) \cup tokens(t)|}$$

Author Similarity The name of authors may have various forms. Some names may be abbreviated; some others maybe exchange the order. Here, we use one simple approach, computing the ngram (we use $n = 2$) and token similarity. .

Title Similarity Trigram and token similarity are the good choices for title. As in title, there may only be spelling errors after normalization.

Publisher Similarity Publisher names are abbreviated in the most random way. We compute word-based LCS (longest common subsequence) of two strings, divide by the length of shorter one and treat the result as similarity.

Year Similarity The similarity is a 0/1 value indicating if two numbers are equal.

2.3.2 Semi-Supervised Learning

Get labelled data In our database, there is one subset called "disambiguations" where there is some not exactly correct information about groups of references representing the same documents. We select a few examples out and manually check the correctness. The labelled dataset is built as:

- Positive examples: References pointing to same publication
- Negative examples: References pointing to different publications

Incorporate unlabelled data After boosting our classifier with the labelled data, we run the classifier on unlabelled data. Those unlabeled data which have higher confidence in one class can be incorporated into the training instances with the label given by the classifier.

Retrain the classifier Finally, we retrain our classifier on all of the training instances we have acquired. Then we can enter the validation phase to examine the features and try to improve the classifier.

3 Evaluation

3.1 Local Clustering

Thanks to the efforts of annotators, there are some ground truth can be used to evaluate the local clustering method described in 2.1.2. The ground truth contains the pair information of one full reference and its partial references. The pair information is manually collected by 2 annotators and in all we have 1843 full references and 5429 references from 53 articles.

After testing the algorithm with some examples, I found it is hard to evaluate the method with a single metric. Because, for instance, if there is a parsing error in one partial reference, then there would be one more false "full" reference and possibly one more cluster. However, in the ground truth, there are only clusters for those true full references. So, I choose to manually check in each cluster if the matches are correct and if there is anything missing.

The result shows that the method is not elegant enough as it is highly dependant on the parsing results. The errors come from:

1. Some "references" in the dataset are not a reference at all and meanwhile some references are not identified and included in the dataset. As our alogrithm is highly dependant on the order, we may match a partial reference to a meaningless string ahead of it.

2. The abbreviation is not recognized and successfully parsed. There are two common cases:

   (a) the reference is mistakenly extracted from the documents with its context which makes it hard to recognizes each field;

   (b) the abbreviation is just not parsed and labelled. For example:

   Chittolini, I capitoli cit., pp. 676 s.

   the "cit" part is not recognized as an abbreviation but a part of title.

3. Mistakenly parsing title as author or author as title, especially for the case "cit" or "op. cit", etc. For example:

   L'Ester, cit., p. 74

   the first field is mistakenly parsed as author instead of title.

3.2 Hash Matching

3.2.1 Hash Black List

Given the fact that there is no ground truth for global-level matching, the help of annotators is necessary. However, for some hash function described above, a great amount of matches will be produced. Because, for example, some named entities is frequently mentioned in the titles of humanities publications. As a result, a large global-level candidate group will be generated, which makes it quite difficult for the annotators. To handle this issue, we need to **ban** some hashes which are too common to have useful information and build what we called the **black list**.

The black list mainly consists of two parts:

1. Too common hashes with more than 500 matches, e.g. "repubblica#venezia" meaning The Republic of Venice, "santa#maria" which frequently occurs in the name of church, or "paolo#sarpi" who was an famous Italian historian.
2. Meaningless hashes due to parsing error, e.g. hashes beginning with "cura". In Italian, "a cura di" means "edited by". When parsing some references, this phrase is mistakenly considered as part of title and then becomes one bigram hash with many matches.

3.2.2 Annotating Data

In order to generate a few samples of hash matching results for annotators, at first, we selected out of the database a few references which at least contains *author*, *title* and *year* fields as possible "full" references, in total 495339 samples.

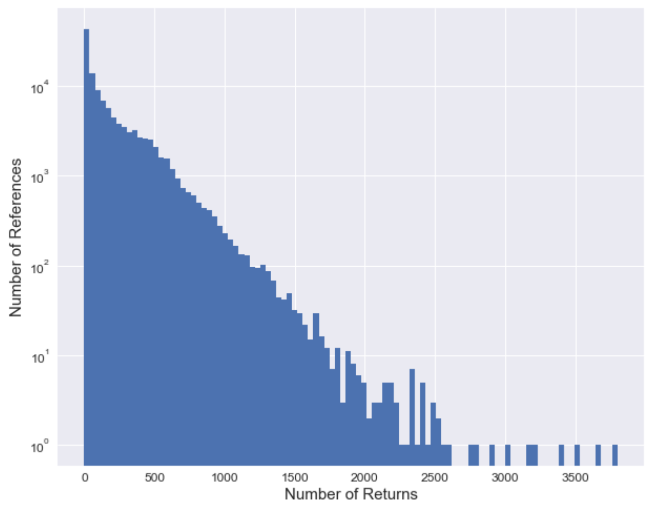

FIGURE 3.1: Hash matching distribution (mean=187, median=83)

For those sampled full references, the matching method described in 2.2 was applied among references from different documents and also the hash black list was used to ban some hashes. As a result, we can get the distribution of the number of returned references. Figure 3.1 presents the distribution of the number of returned references for 117029 seed references, a quarter of all the samples.

As shown above, the number of returned references for most sampled references is less than 500. To ease the burden for annotators, based on the distribution, we choose 30 references out for annotating, 10 with 0-10 returned references, 10 with 10-100 returned references and 10 with 100-500 returned references, of which the number of returned references are shown in Table 3.1.

TABLE 3.1: Number of returned references in each annotating group

No.	0-10	10-100	100-500
0	2	13	116
1	2	13	151
2	3	14	182
3	4	15	185
4	4	21	186
5	4	91	481
6	5	84	464
7	5	83	432
8	6	72	421
9	7	64	415

For each group, information about the number of returned references by each hash function is summarized in Table 3.2. Due the popularity of some bigrams, **Title** hashes have a large number of matches, followed by **Author-Year (Blur)**, **Author-Year** and **Author-Title**. And those hashes with page and number seem not useful for these sampled references as there are limited number of matches.

TABLE 3.2: Number of returned references for each hash function

Hash Function	
Baseline	15
Title	3031
Author-Title	1026
Author-Year	1011
Author-Year (Blur)	1350
Author-Year-Page	2
Author-Year-Page (Blur)	3
Author-Year-Num[3]	217
Combined	3583

3.2.3 Analyzing Results

The recall and precision are calculated as the mean of each seed reference and the results are presented in Table 3.3. For each hash function, we only take into account those references which have such hash. For example, the reference

Vito Fumagalli, Solitudo camis: Vicende del corpo nel Medio- etw.Bologna 1990;

does not have page information, so it will not be taken into account for the calculation of the hash function, **Author-Year-Page**.

TABLE 3.3: Recall and precision (%)

Hash Function	Recall	Precision
Baseline	6.48	27.78
Title	82.44	33.59
Author-Title	73.89	54.74
Author-Year	84.52	62.87
Author-Year (Blur)	86.25	43.82
Author-Year-Page	12.50	12.50
Author-Year-Page (Blur)	25.00	25.00
Author-Year-Num[3]	20.75	31.65
Combined	84.47	25.47

Overall analysis
To begin with, the baseline hash function performs not well both in recall and precision, as in different publications the citation styles are not the same or even have large difference.

The bigrams of title performs really well in terms of recall, but the precision is not good. On the other hand, author-title and author-year hashes perform better with a high recall and a not low precision.

The page and number information seems to have little influence partly because of the parsing error and partly because in one publication author may cite different pages from one document, which treats one document as several parts and cannot provide meaningful match.

Sources of false negative
In this part, we study the sources of false negatives, i.e. true matches are not returned by the hash functions defined above:

- There are in total 54 false negative citations.
- All of them are not included in our "full reference" samples as they do not contain *year* or *author* or *title* field.

The main problem is the assumption we've made about the minimum required fields for a full citation, *author* and *title* and *year*. However, for some full citations, there are only two or less fields.

3.3 Global Clustering

3.3.1 Training Process

As discussed in 2.3.2, the training process starts from a small number of manually checked reference pairs. A logistic regression model is first trained on 100 manually checked pairs, 50 positive and 50 negative examples. On a rolling basis, the model is run on unlabelled data pairs and those pairs with high confidence (we use probability 0.9 as the threshold) are chosen as new training data with a label given by the classifier. As a result, we get 12000 pairs in the training set (6000 negative and 6000 positive examples).

The classifier is then retrained on the new training set. The feature weight of the Logistic Regression model is shown in Figure 3.2. The token similarity of title contributes the largest wight followed by exactly year match and token similarities of author. On the other hand, the Ngram similarities for title and author do not have significant weights in this model.

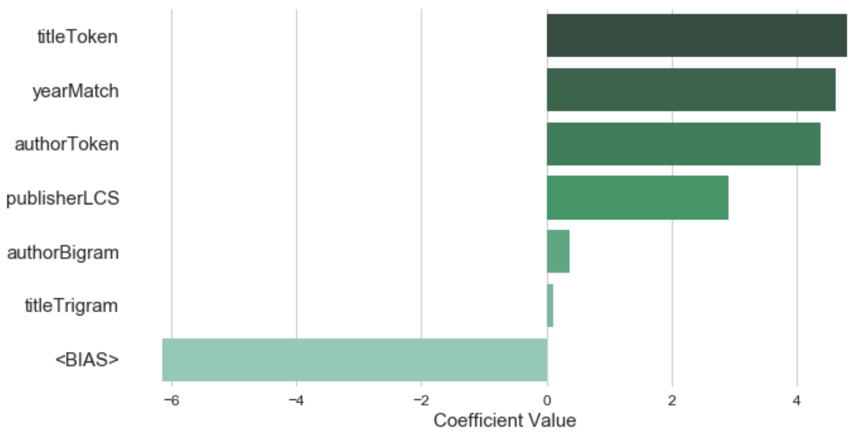

FIGURE 3.2: Feature Weight of Each Feature in the Model

3.3.2 Evaluating Classifier

To evaluate our classifier, we use the same annotated data as 3.2.2 which is of 2890 negative pairs and 752 positive pairs.

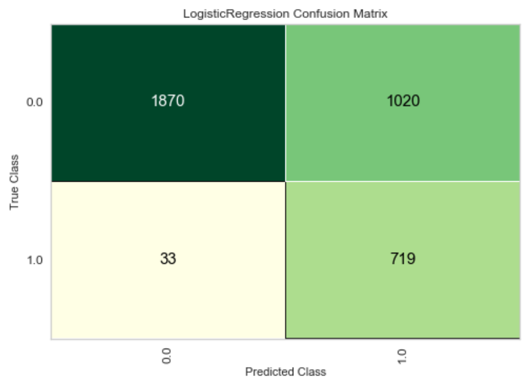

FIGURE 3.3: Confusion Matrix (Recall 96%, Precision 41%)

The confusion matrix of evaluation result is presented in Figure 3.3. From the recall perspective, the model has good performance. As calculated from the confusion matrix, the recall of the model is 96%.

Further analysis on the sources of false negative finds out that a large percent of false negative is due to the extracting or parsing error. One case is that the reference is extracted with its context, which makes a long "title" field and thus low title similarity. Another case is that because of parsing error, title, year or author is not correctly recognized. For example:

- *G. Lo- RENZETTi, Venezia e il suo estuario, Venezia, Bestetti e Tumminelli, 1926, p. 381;*

- *G. LORENZETTI Venezia e il suo estuario Roma [1926].*

The author field of the former one contains hyphen which makes zero author similarity as there is no common tokens. Also, in the latter, year has been mistakenly recognized as title, which makes zero year similarity and also smaller title similarity.

Receiver Operating Characteristic (ROC) curve is a measure of a classifier's predictive quality that compares the tradeoff between the model's sensitivity and specificity. X axis represents the False Positive rate while the Y axis presents the True Positive rates. Thus, the ideal point is the top-left corner of the plot. Another corresponding metric is AUC, area under the ROC curve. Generally speaking, the higher the AUC, the better the model is.

The ROC curves of our model and corresponding AUC values are shown in Figure 3.4. The classifier performs mediocrely as the ROC curve is far away from the top-left corner. Although the AUC reaches 0.9, the curve is not steep enough and therefore the model is not good enough.

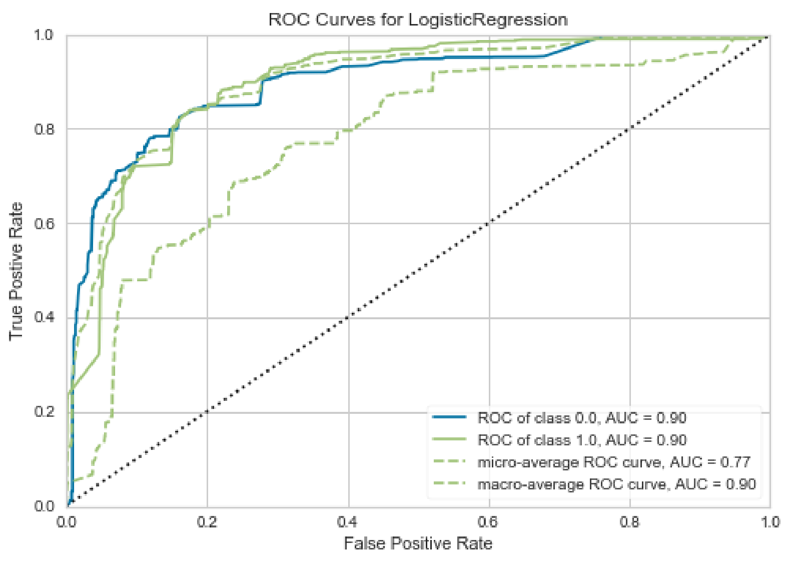

FIGURE 3.4: ROC Curves for Logistic Regression

3.3.3 Further Comments

When looking back to our training set, we found out that most of the negative training data is **0** vector. However, in our evaluation set, most of the negative samples are not **0** vectors. The training set is built from our "disambiguation" dataset. The negative examples in training set are any two citations that do not point to the same documents and they always have less in common. On the other hand, the negative pairs in evaluation set always have something in common as they share the same hashes, which would have non-zero similarity vectors. In other words, there is a difference in the distribution of our negative samples between training and evaluation set, which may be responsible for the mediocre performance of our classifier.

4 Conclusion

This project proposed and implemented a step-by-step approach for efficient citation matching in Linked Books dataset. The large scale of data and the lack of ground truth are two challenges for this project.

The first phase of the approach is to locally matching partial references with corresponding full references. As the designed algorithm is highly dependent on the parsing results, minor errors in identifying references or recognizing abbreviation would have negative impact on clustering.

The second phase is to fuzzily group those full references globally among documents. Hash function is applied to complete the matching task. Samples generated from hash matching are collected and manually labelled by annotators. **Author-Year (Blur)** is the most recommended one as it has an overall highest recall and not bad precision. **Author-Title** and **Author-Year** are also preferable.

Finally a binary classifier is built semi-supervisedly. The classifier takes similarities between two citations as features and classify whether they refers to the same document. Feautres are chosen as bigram-similarity and token-similarity of author, trigram-similarity of title, also similarity of publisher and year. The annotated data in second phase is used for evaluation and the model has a good recall but poor precision. One reason is that most of the negative examples in the training dataset have **0** feature vectors while in the evaluation set every pair share the same hashes and would have non-zero feature vector. So, the way we choose the training set needs further discussion.

The source codes of the project are published on GitHub.

Bibliography

[1] G. Colavizza, M. Romanello, and F. Kaplan, "The references of references: A method to enrich humanities library catalogs with citation data", *International Journal on Digital Libraries*, 2017, ISSN: 1432-1300. DOI: 10.1007/s00799-017-0210-1. [Online]. Available: https://doi.org/10.1007/s00799-017-0210-1.

[2] S. Hitchcock, L. Carr, S. Harris, J. Hey, and W. Hall, "Citation linking: Improving access to online journals", in *Proceedings of the second ACM international conference on Digital libraries*, ACM, 1997, pp. 115–122.

[3] M. Fedoryszak, Ł. Bolikowski, D. Tkaczyk, and K. Wojciechowski, "Methodology for evaluating citation parsing and matching", in *Intelligent Tools for Building a Scientific Information Platform: Advanced Architectures and Solutions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 145–154, ISBN: 978-3-642-35647-6. DOI: 10.1007/978-3-642-35647-6_11. [Online]. Available: https://doi.org/10.1007/978-3-642-35647-6_11.

[4] M. Fedoryszak and Ł. Bolikowski, "Efficient blocking method for a large scale citation matching", *D-Lib Magazine*, vol. 20, no. 11/12, 2014. DOI: 10.1045/november14-fedoryszak. [Online]. Available: http://mirror.dlib.org/dlib/november14/fedoryszak/11fedoryszak.html.

[5] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[6] M. Fedoryszak, D. Tkaczyk, and Ł. Bolikowski, "Large scale citation matching using apache hadoop", *CoRR*, vol. abs/1303.6906, 2013. arXiv: 1303.6906. [Online]. Available: http://arxiv.org/abs/1303.6906.