A Knowledge Base of Mathematical Results

Theo Delemazure

Master 2 IASD

PSL University
École normale supérieure

A Knowledge Base of Mathematical Results

Final report

Theo Delemazure
theo.delemazure@ens.psl.fr

March-August 2020

Advisor:
Prof. Pierre Senellart

Ecole normale supérieure
Team VALDA
45, Rue d'Ulm

Abstract

The basic unit of information of use by researchers in theoretical fields are the mathematical results. We aim to build a knowledge base of these results, using information extraction techniques on scholarly documents. We present an algorithm which extracts mathematical results and references to mathematical results from scientific papers, using their PDF or LaTeX sources. We analyse the results of our algorithm on the whole `arXiv` database of scientific papers and explore the resulting graph of mathematical results, which contains more than 6 million results and 4.5 million edges. We present attempts to link theorems of different papers using a TFIDF vectorizer or an autoencoder.

Contents

1 Introduction **4**

2 Related work **6**
- 2.1 Information extraction from scientific publications 6
- 2.2 Scientific digital libraries and open archives 6
- 2.3 References extraction and analysis 9
- 2.4 Knowledge bases of theorems 9

3 The CS-CC database **11**

4 Theorem Matching **13**
- 4.1 The theorem dataset 13
- 4.2 TF-IDF Vectorizer 14
- 4.3 Autoencoder 15

5 The graph extraction algorithm **16**
- 5.1 From PDF to XML 17
- 5.2 Extracting results 18
- 5.3 Associate tags and papers 20
- 5.4 Results on the CS-CC dataset 21

6 The `arXiv` Database **24**
- 6.1 The dataset 24
- 6.2 Quantitative Analysis 25
- 6.3 Qualitative Analysis 31

7 Future work **34**
- 7.1 Improve the extraction algorithm 34
- 7.2 Crowdsourcing 34

1 Introduction

In the context of my IASD Master's degree, I did a 5-month internship with the *VALDA Team* of the *École Normale Supérieure of Paris* on the TheoremKB project.

TheoremKB is a project started by my internship advisor *Pierre Senellart*. The goal of this project is **to transform the way scientists navigate through scientific literature**. Nowadays, access to scientific literature mainly relies on search engines and digital libraries like Google Scholar, DBLP, arXiv, or SemanticScholar. On these platforms, one can search for articles by keywords, authors, title, dates, and other metadata. For some of them, it is also possible to navigate through the graph of citations. However, the basic unit of information is always **the paper itself**, and the PDF associated with it.

But the actual unit of information of use by scientists in mathematical fields (*mathematics, theoretical computer science*) are the **mathematical results** (*theorems, lemmas, claims, propositions, etc.*) contained inside the paper, and how they rely on each other. The goal of TheoremKB is to turn the scientific literature from a collection of papers to **a knowledge base of mathematical results**.

Such a knowledge base could be **very helpful** to understand how different results rely on each other, or to explore classes of results (for instance, get all the known NP-Complete Problems). It can also help authors to understand the dependencies between the results contained in their own papers.

This internship and the one of *Lucas Pluvinage*, another IASD student, come at the beginning of the project. During this internship, our goal was to get a sense of **how dense** the knowledge base mentioned above could be, and **to create a first connected graph** of mathematical results. While I was more involved in finding a way **to detect dependencies** between mathematical results, *Lucas*' task was to extract the results and their proofs from the paper using information extraction techniques.

The remainder of this document is organized as follows. **Section 2** introduces the existing literature that can be linked to the project, and the most advanced tools and platforms currently available to browse the graph of scientific publications. In **Section 3**, I describe the dataset of papers I used for my experiments during the first 4 months of the internship to test my scripts. My attempt at linking theorems from different papers based on the statement associated with them is described in **Section 4**. The main part of my work is detailed in **Sections 5 and 6**, respectively discussing the algorithm we built to find and link mathematical results together, and the analysis of how this algorithm performed on the whole `arXiv` database.

Finally, **Section 7** presents some of the ideas that have been started during my internship, but remain unfinished and need to be pursued.

2 Related work

In this section, I will present **the existing literature and tools** on some topics related to the TheoremKB project and more precisely to my internship.

2.1 Information extraction from scientific publications

There as been a lot of research around the extraction of information from scholarly documents, mainly focused on two main areas of the document: **The header** (title, authors, abstract, etc.) and **citation data**. A few other works have looked at the extraction of other parts of the document, such as *figures* [21] and *tables* [13].

GROBID [2, 15], created by *Patrick Lopez* is a free, open-source tool implementing these techniques. It takes as input the PDF of a scientific article and **returns an XML file** of the paper parsed into different sections, with figures, tables and equations extracted. It relies on PDFAlto [3], another useful tool created by the same team, which can read a PDF and return the position of every single word of the PDF.

This tool proved to be very powerful and useful for our task, even if it contains some bugs, like mathematical results often detected as figures. These bugs are probably due to the fact that GROBID is **trained on a very small corpus of paper**, all of them being manually annotated.

2.2 Scientific digital libraries and open archives

An important part of the project is to **create a knowledge base** storing all relevant information about mathematical results and their proof, and more generally about scientific papers as a whole. It would then be essential to have **a public web platform** on which researchers could easily navigate, and query the graph of scientific papers and mathematical results. I will present in this section some existing **digital libraries** and **open archives** platform that enables to browse the graph of scholarly documents.

Digital libraries Nowadays, a large number of platforms give access to the scientific literature. Most of them are **digital libraries**. Often these services are directly provided by commercial publishers, such as Scopus (by *Elsevier*), Web of Science (by *Clarivate*), or the ACM digital library. Some others are general search engines for scientific publication, the best known being Google Scholar and Microsoft Academic [23].

Some platforms are more independent digital libraries **gathering data from various publishers** on one platform (MathSciNet and EuDML for *mathematics*, DBLP for *computer science*, PubMed for *biology and medicine*, etc.).

All these platforms propose **the PDF file** of the publication, often **metadata** (authors, abstract, venue, etc.) and sometimes extract **figures and tables** from the paper, but they never allow to query directly **the mathematical statements** from the research article. Dissem.in is another digital library, gathering data from various platforms and showing precisely which papers are open access and which are not. It also proposes an API that enables to find the *doi* of a paper given its title and authors.

There also exist **illegal digital libraries**, such as **Sci-Hub** and **Libgen**, which give access to a lot of publication without any *paywall*. We only consider here sources with no legal issues.

Open archives. Another kind of platforms is **open archives**, on which authors can upload **preprints and publications**, together with the sources of these publications. For instance, the project HAL originated by the *CNRS*, counts more than $700,000$ hosted publications. The best known is `arXiv`, created by *Cornell University*, and it is the one we will use in this paper. It contains **more than** 1.7 **million papers** published on various scientific topics (mostly *mathematics, computer science and physics*) since 1991. We can also mix these open archives with a **social network style** and obtain something like ResearchGate, on which researchers can share and comment articles, as we comment posts on *Facebook* or *LinkedIn*.

The S2ORC dataset. SemanticScholar is one of the most interesting of these digital libraries. As a lot of them, it shows the metadata of the article, the references and the citations. They also automatically extract figures, tables and discussed topics in the paper, and they detect related papers. A very interesting feature of this platform is that it **sorts references into three categories** and automatically detect important references. I will talk about this in more detail in **Section 2.3**. The website counts more than **80 millions papers** and propose a very useful API to easily get a JSON file of any article.

Even more interestingly, the *Allen Institute for AI*, at the origin of *SemanticScholar*, released a huge dataset called S2ORC [14]. This contextual citation graph contains 81.1 million academic publications and links between them, but it also contain **the parsed full text** of 8.1 million open access papers. For each publication, they gathered the different versions of the pa-

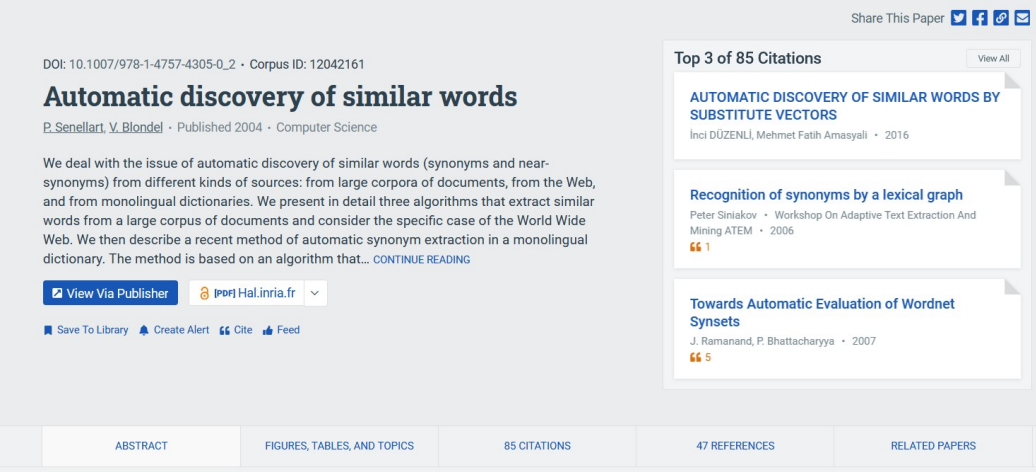

Figure 1: An example of the webpage of one paper on a **digital library**, here *SemanticScholar*.

per and picked the most representative one. Then they used GROBID [2, 15] presented above to **split the paper** in different subsections and **automatically extract** figures, table, formulas from the text. As I explained before, GROBID also extract references to these elements, and to bibliographical references. Finally, they simply linked papers in the bibliography of the PDF files to existing papers in their dataset. **Figure 2** (and its caption), taken from [14], sums up the content of one document in the S2ORC dataset.

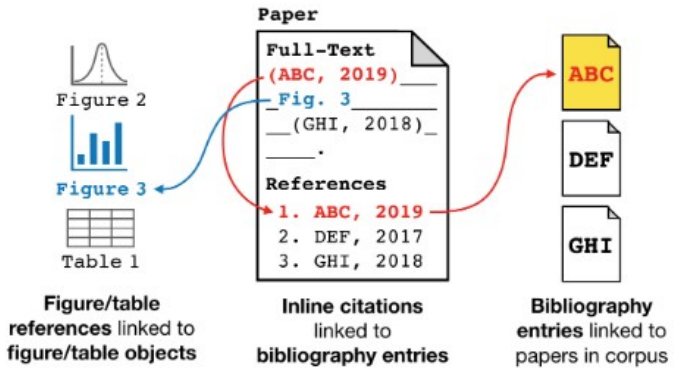

Figure 2: [14] *For each paper,* ***in-line*** *citations are linked to the corresponding bibliography entry within that paper. The bibliography entries are then* ***linked to one of 81.1M candidate papers****. References to* ***figures and tables*** *are also extracted from the full text and linked to figure and table objects within the paper.*

Other citation graphs. This is not the only **contextual citation graph** publicly available. In 2009 was published the *ACL Anthology Network* [19], a graph of scientific publications derived from the *ACL Anthology Corpus* **manually augmented** in order to link references in the bibliography to papers in the corpus (it contains around 25K papers). More recently, *Saier and Färber* introduced a citation graph derived from `arXiv` LaTeX sources in [22].

2.3 References extraction and analysis

Digital libraries and **contextual citation graphs** mentioned in the previous section are directly useful to researchers, but they are also useful indirectly by the mean of various research tasks. We can cite for instance the **citation recommendation task** for authors writing a paper, either at a *paper level* [5, 12] or an *in-line level*, using the context [9]. It can also be used to detect similar papers in order to **build a recommendation system** (for instance, *Microsoft Academic* built is own recommendation system for scientific papers [10]). There is also work done **to generate citations** in a realistic and natural sentence, given two papers (the first one citing the second one) [16].

An interesting task for us is **citation extraction and disambiguation**. *Cohan et al.* proposed in [7] a classifier trained to detect the intent of a citation, given the sentence in which the citation appears. They trained it to detect three main intents: **Background** (for instance, citations in this Section of this report are mainly background citation), **Result** (these citations are the one that we aim to extract with the algorithm presented in **Section 5**), and **Method**. To train their classifier, they used a bidirectional *Long Short-Term Memory* (LSTM) with an attention layer, followed by three independent *Multi-Layer Perceptron* (MLP) with three different tasks: one with the **main task** (guessing the citation intent) and the two other with **scaffolds** (e.g., guessing the title of the section from which the citation is taken).

2.4 Knowledge bases of theorems

Another aspect of this project is that we want a **structured knowledge base of mathematical results**. There has been work done on knowledge bases of Theorems in the context of **automated theorem provers**. Some of these tools include libraries which contain a lot of theorems and their proof. The model *MBASE* [11] proposed in 2001 aimed to create a knowledge base

with formal representations of mathematical results. However, the development of this model seems to have stopped in 2003, without any system demonstration.

Logipedia is one of the most complete **library of mathematical results**, put as formal statements and useable for a theorem prover. However, this is not as *human-readable* as TheoremKB aims to be.

Finally, *Ganesalingam and Gowers* proposed in [8] a very interesting automated model which aims to **write proofs in a human style** by trying to mimic researchers' reasoning.

3 The CS-CC database

Dataset creation. For our experiments, we needed a dataset of scientific papers **dense enough** so that this is likely that there exist *links* between results from different papers. The dataset must **not be too big** so that our scripts will not take too much time. It is possible to download **bulks of papers** from `arXiv` (and other websites), but publications are sorted **by months** and paper published the same month are less likely to quote each other. Moreover, `arXiv` enables to download both the pdf and the source of papers published on it (when available).

Instead, I crawled the export version of the `arXiv` open archives, and downloaded all PDFs and sources of papers published between 2010 and 2020 under the CS-CC category. CS-CC stands for *"Computer Science - Computational Complexity"*. We chose this category because *computational complexity* is a domain with **a lot of mathematical results** and close enough so that papers will very probably **quote each other**.

The dataset I gathered contains 6.000 scientific papers. The second step was to **get the links** between the different articles of the dataset. The first methods I used to obtain them were not very efficient, but it enabled me to familiarize myself with some useful tools.

Extracting citations. Indeed, I first tried to use the *dissem.in* API. If one sends the title of an article, the publication year and the names of the authors to the *dissem.in* API, the API **returns the DOI** of the paper. The idea was to find the DOI of each paper in the dataset, and then the DOI of articles in the bibliography of all papers in the dataset. Once this is done, we just have to find matching DOI. I tried to extract the information required by the API from the LaTeX source code, but, as it is shown in **Table 1**, more than *one fifth* of all papers in the dataset do not have any latex file in their sources (or no sources at all). Only 62% have a file for the bibliography, often a *.bbl* file, which is not ideal because its format can vary from one paper to another. Only 3% of them have a *.bib* file for the bibliography.

.bib	.bbl	.tex
3%	62%	78%

Table 1: Proportion of papers in the CS-CC dataset having a *.bib*, *.bbl* or *.tex* file for the bibliography.

Instead of directly using the source files, I decided to extract information **directly from the PDF** using GROBID [2, 15], a tool based on machine

learning and able to convert a PDF of a scientific paper into an XML file. The returned **XML file** contains structured information on the paper, such as **header data** (*title, authors name, universities, etc.*), **body data** (*sections, figures, equations, references, etc.*) and most importantly **bibliographic data** (*title, authors name and publication year of each quoted article*). However, the results were still not convincing since GROBID and *dissem.in* have their own bugs.

Fortunately, the digital library *SemanticScholar* for scientific literature offers an API which only needs the `arXiv` id of an article and returns a detailed JSON file, containing the `arXiv` id of **every `arXiv` paper referenced in the bibliography**. Thanks to that, I was able to obtain a dense graph of articles and inter-articles dependencies.

Graph analysis. 2.666 of the 6.000 papers are cited by at least one other paper from the corpus, and 3.190 of them are citing another paper of the corpus. **Figure 3** shows the distribution of the number of citations *intra-corpus* per article. This highlights the fact that despite its small size, this dataset is **dense enough** for our experiments.

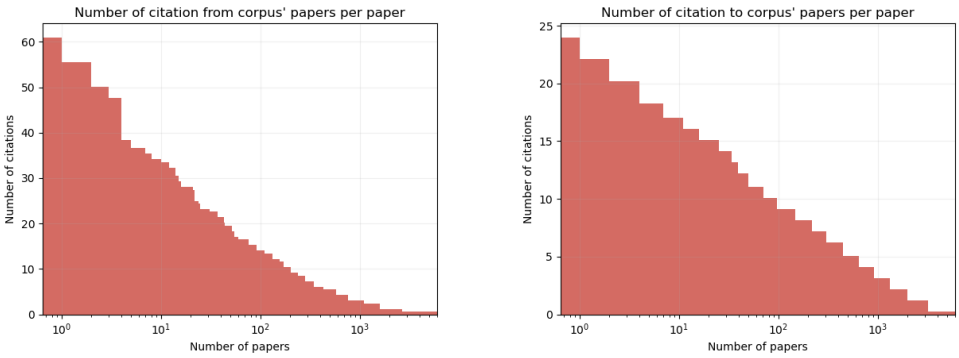

Figure 3: Distributions of the number of citations *from* (left panel) and *to* (right panel) other papers of the corpus.

The **longest path** in this graph is of length 19 and the **biggest connected component** contains 3.167 papers and 8.106 edges between papers. **Figure 4** shows a representation of this component obtained using the *NetworkX* library on Python.

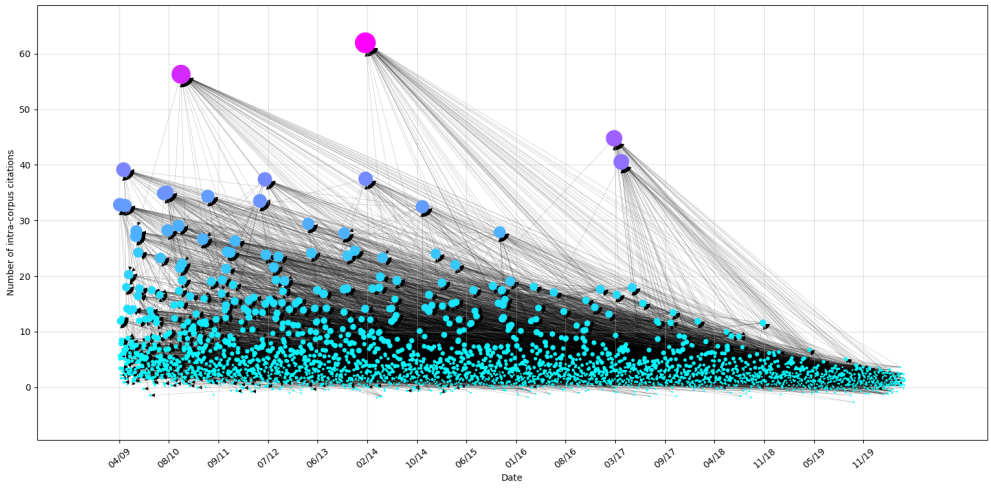

Figure 4: Representation of the graph of dependencies between papers of the corpus. The *x-coordinate* of a point represents its **publication date** (between 2009 and 2020) and the *y-coordinate* represents its **number of citations** from papers in the corpus.

4 Theorem Matching

This section is dedicated to my first attempt at **linking results** from different papers. The idea was to detect pairs of theorems which are similar. For example, it would detect if a theorem in some paper is **a restatement** of a theorem in one of the papers referenced in the bibliography.

4.1 The theorem dataset

For each paper in the CS-CC dataset, I extracted the set of mathematical results and their statement from the LaTeX source files. To do so, I simply searched for each "`\begin{theorem}`" and "`\end{theorem}`" using the `TexSoup` library in *Python*. For each result, a *regexp* detects if some paper is cited inside the result and we check if the cited paper is in the CS-CC dataset. If a citation to another paper is detected, we use **a classifier** to find which particular results of the cited paper is associated to this one. The different classifiers we tried are described in the next sections. Note that we only consider citations which are *inside* the statement and not the ones which are in the proofs. Indeed, in the first case, it is more likely that the result **is a restatement** of a result in the cited paper, and in the second case, the result probably **relies on** a result of the cited paper.

A total of 41,167 results were successfully extracted from LaTeX files and 308 theorems are referencing another paper from the corpus (there might be duplicates). To see the efficiency of the different methods, I selected **a subset** of 39 theorems in which the precise result of the cited paper we want to find is already mentioned in the header (e.g *"Theorem 3 from [4]"*).

The remainder of this section is dedicated to the different methods I tried to detect the similarities between different theorems. **The accuracy** obtained with each method is reported in **Table 2**.

Method	**Found**	**Accuracy**
TF-IDF	29/39	75%
Autoencoder	21/39	54%

Table 2: Efficiency of the different methods for **theorem matching**.

4.2 TF-IDF Vectorizer

The first model I used is based on **sentence vectorization** using the frequencies of words in the sentence. I tried several vectorizers, but the most efficient one was the *TDF-IDF Vectorizer*.

TF-IDF stands for *Term Frequency Inverse Document Frequency*. Given **a set of documents** $\mathcal{D}$, we will associate to each pair of **document** and **term** (d_i, t_j) a weight $\text{tfidf}_{i,j}$. This weight is the product of two numbers:

1. Term frequency : $\text{tf}_{i,j}$ is equal to the number of occurrences of the term j in the document i.

2. Inverse Document Frequency : The idf_j of the term j is a function of the inverse of the proportion of documents containing this precise term. For instance, it can be the log of it.

$$\text{idf}_j = \log \frac{|\mathcal{D}|}{|\{d_i \mid t_j \in d_i\}|}$$

Finally, we can compute the tfidf *weight* for the pair (d_i, t_j) :

$$\text{tfidf}_{i,j} = \text{tf}_{i,j} \times \text{idf}_j$$

In our case, **documents** are **theorems** taken in a bunch of papers, containing the citing paper and the cited paper. We denote the set of theorems $\mathcal{T}$. **Terms** are simply **words** appearing in these theorems. We can

associate to each result a vector $v(t_i) = (\text{tfidf}_{i,1}, \cdots, \text{tfdif}_{1,m})$. I used the `TfidfVectorizer` function from `sklearn` library, with *english stopwords*, which directly returns the vector of every theorem given the set of theorems.

If we want to find the result in a paper $\mathcal{P}_2 \subset \mathcal{T}$ which is **the most similar** to a result t_1 in a paper $\mathcal{P}_1 \subset \mathcal{T}$, we simply take

$$t_{max} = \arg\max_{t_2 \in \mathcal{P}_2} v(t_1)^t v(t_2)$$

This method works well when the two theorems are **almost identical**. However, it gets complicated when there is a lot of theorems in the paper $\mathcal{P}_2$, since it is very likely that theorems of the same paper contain almost the same terms. I also tried to use **N-grams** instead of unique words, but the results were worse.

4.3 Autoencoder

I also tried to create my own theorem vectorizer using **a variational autoencoder**, inspired by [24, 6] and more generally by autoencoders seen in class and in [20].

First, I needed to choose **embeddings** for words. Given that all the theorems share a scientific vocabulary, I tried to train embeddings using the **CBOW method** [17]. However, this did not work well, maybe because the dataset of theorems was too small. I decided to use **pretrained Glove embeddings** [18] with 50 dimensions and 6 billion tokens instead.

In our autoencoder, the input of the *decoder* is a fraction of the statement and the output of the *encoder* and it must guess the masked words of the sentence. The fraction of words which is masked is called the *word dropout*. For instance, when the *word dropout* is equal to 0.3 we obtain the loss shown in **Figure 5**. Both encoder and decoder use a **GRU recurrent neural network**. The best results I obtained using this method are not really convincing (see **Table 2**), and much worse than the results obtained using *TF-IDF*.

I did not investigate more on this topic, because the dataset was too small and it was hard to train a classifier in **an unsupervised fashion**.

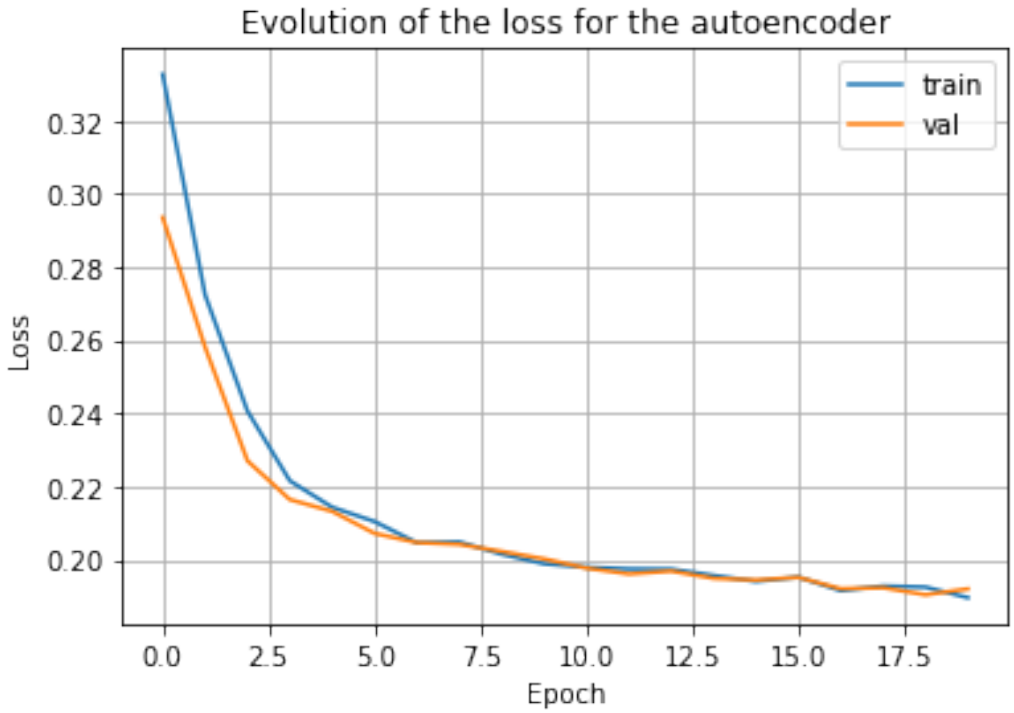

Figure 5: Evolution of the **training loss** and **validation loss** of the autoencoder with a *word dropout* of 0.3.

5 The graph extraction algorithm

In this section, I will present **the main algorithm** of this internship, which enables us to create a first knowledge base of mathematical results. Our goal is to extract **results**, as well as **references** to external and internal mathematical results from PDF files of scholarly documents, and gather everything to form a knowledge base of theorems and proofs. By *internal* reference I mean a reference to another result of the same paper. On the contrary, an *external* reference is a reference to a result in another paper. This extraction is done in **three steps**:

1. First, we want to **convert the PDF into an XML file** containing the positions of the mathematical results and their proofs. One of *Lucas Pluvinage*'s goals during this internship was to build a strong classifier to automatically extract results and proofs. He uses *Conditional Random Fields* (CRF) for that, for more information please see his report.

2. Now that we have boxes and we know they contain either results or

proofs, we can **extract information that interest us**. This includes the *name/number* of each result (or name of the result proven if it is a proof) and more importantly, references to internal and external results. This part is harder than it looks and it was an important part of my work.

3. Finally, we need to **associate external references** to `arXiv` papers using their tags. By tag, I mean the number or word between brackets, associated to a paper in the bibliography (e.g, the tag of [15] is *"14"*).

However, since the *CRF* was not yet ready for the first step, we extracted results and their proofs directly from the LaTeX sources. I explain how we did it in more detail in **Section 5.1**.

Figure 6 sums up the extraction algorithm.

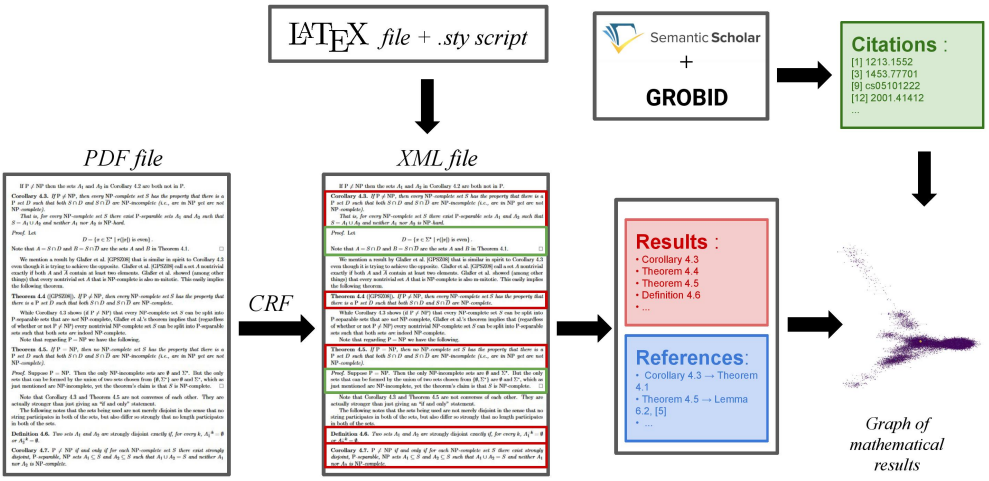

Figure 6: **Flow of the algorithm**. We convert a PDF file into an XML file using a CRF, or we use source files for this purpose, then we extract results and references. Finally, we associate references with papers in the corpus to build a graph of mathematical results.

5.1 From PDF to XML

The first step is to **detect the bounding box** surrounding mathematical results and proofs. *Lucas* was working on a CRF, which would automatically detect these boxes from the PDF file. However, the classifier was not ready

and we had LaTeX sources of a big proportion of the downloaded papers, so we thought of another solution.

Indeed, we use a LaTeX extension script to set **a hyperlink around results and proofs** during compilation. For more information on how this is done, please see *Lucas*' thesis.

Consequently, we need to compile the LaTeX source code ourselves. Once compiled, we obtain a new PDF with bounding box around results and proofs, which are actually false hyperlinks. Using PDFAlto created by *Patrice Lopez* [3] on the new PDF, we were able to extract the position of every hyperlink box, together with the address they are pointing to. Therefore, with a simple regexp, we can easily extract **the full statement** of the mathematical results and proofs of the document.

5.2 Extracting results

Reading the boxes. At this point, we know the position of hyperlink boxes. We can divide this boxes into two groups: *Results* and *Links*. *Result* boxes contain either a mathematical result or a proof, and *Link* boxes are external links or internal hyperlinks to other elements of the article (e.g., *"Section 6"*, *"Figure 12"* or *"[2]"*).

For **each word** of the document, we will detect if it is inside a *Result* box and inside a *Link* box (it can also be in both, none, and even nested *Result* boxes). Checking it naively for each word and each box would take a lot of time (number of words × number of boxes). *KD-Tree* is a really good data structure for this problem.

KD-Trees In a *KD-Tree* [4], the space is partitioned to efficiently store the points and find them afterwards. Thanks to this optimization, the running time of this second step was **divided by a factor 10**.

However, KD-Trees store points, not boxes. My algorithm memorizes one KD-Tree for each page of the PDF and each kind of box (*Link* and *Result*), containing the center of every box of the page. Then, if we want to know if a word is in a box, we search for the **nearest boxes of one word** in the selected KD-Tree and check if the word is inside one of these neighbouring boxes.

Extract results. Now that we have extracted the full text of results and proof, we would like to be able to **enumerate results and references** of the document. First of all, we use the regexp 1 to **detect the name of a mathematical result**, which is any word followed by some kind of

numbering (e.g., *"Theorem 1"*, *"Lemma 2.54.2"*, *"Problem A.1"*, *"Conjecture E.7.1.3.1"*, etc.). However, this script **will fail** if the result is not numbered or is preceded by some character in the text.

```
(\w+)\s+([a-z]\.)?[\d]+(\.\d+)*
```
(1)

We also need to **associate each proof to one result**. For that, there are two possibilities:

1. It is specified at the beginning of the proof which result is being proven, and a simple regexp is sufficient to find it.

2. There is no indication. In that case, we associate the proof to **the most recent** mathematical result seen in the document.

However, I believe that this part can be improved. For instance, there is sometimes a *Remark* between a *Theorem* and its *Proof*. In that case, my algorithm will associate the proof to the *Remark* if the authors did not specify anything in the proof.

Extract reference. Now, we want to find references **to other mathematical results**. For this we are searching for particular words followed by some kind of numbering in the text. The words of interest are the following:

Theorem	*Claim*	*Conjecture*	*Corollary*	*Definition*
Lemma	*Proposition*	*Observation*	*Property*	*Construction*
Example	*Exercise*	*Note*	*Problem*	*Case*
Question	*Solution*	*Remark*	*Fact*	*Hypothesis*

Table 3: List of all the possible mathematical result names used to detect references.

To this list I added some **abbreviations**, like *"thm."*, *"lem."* and *"prop."*. However, a lot of abbreviations are still not detected, as explained in Section 6.3.

Once we have detected a reference to a result, we need to determine whether this is an *internal* or an *external* reference:

- If **a hyperlink is detected** on the result name or numbering, and this hyperlink points to a result inside the paper, then it is obviously **an internal reference** to this result.

- If there is no such hyperlink, we look at **the context** of the reference. By context, I mean the 5 words before and after the reference to the mathematical result. If a link to the bibliography is detected inside the context, we know it is a reference to **an external result**. Otherwise, we say by default that it is a reference to **an internal result**.

For external references, we memorize the tag detected in the context (e.g. *"[5]"* or *"[MoI17]"*) so we can find if the cited paper is in the `arXiv` corpus later (see more details in Section 5.3).

There are some **other optimizations** in this small part of the algorithm:

- We skip the reference if this is **a reference to the result itself** (for instance, *"this concludes the proof of Theorem 3"*).
- If there are several links to the bibliography in the context, we take **the closest one**.
- We detect when **several results** are referenced at the same time, and separate them into several references. For instance, *"Theorems 1, 5 from [2]"* becomes *"Theorem 1 from [2]"* and *"Theorem 5 from [2]"*.
- Sometimes, references to results or sections of the paper (e.g., *"See Section 5.3"*) are falsely detected as links to the bibliography. We automatically remove these links by detecting if they contain particular words, like those in **Table 3**.

5.3 Associate tags and papers

The last step is to associate the tags found in the previous steps to other `arXiv` papers. For that, we want to know **which tag is associated to which paper**, and it requires a bit of work.

Using S2ORC. I downloaded the latest version of the S2ORC dataset [14] from *SemanticScholar*, that I already presented in **Section 2**. The interesting feature of this dataset for us here is that *SemanticScholar* already found **the `arXiv` id** of every reference in the bibliography, when such an id exists (i.e. the paper must exists on `arXiv`).

Moreover, since the S2ORC also uses GROBID to parse the text of the document, **in-line bibliographical references** in the text are already extracted, and so are tags associated with these references. Consequently, using only this dataset, I was able to obtain the tag associated with each `arXiv` reference in the bibliography of every document of my dataset. This gives us a first dataset of links between `arXiv` papers. Let's call it *dataset A*.

However, I was frequently faced with a particular issue: The version of the paper I had in the CS-CC dataset (See **Section 3**) or in the full `arXiv` dataset (see **Section 6**) was not the same as the version used by *SemanticScholar* in their S2ORC dataset. This has unwanted consequences: sometimes tags in the bibliography are simply different, for instance tags are numbers in one version (*[1]*, *[2]*, etc.) and names in another one (*[VK16]*, *[Hi15]*, etc.), but sometimes **the order of the references is changed**. By that I mean that the tag *[1]* might not represent the same reference in our paper and in the paper used in the S2ORC dataset. As one can see on **Figure 12** of **Section 6**, this kind of problem is **frequent enough to require fixing**.

Using GROBID. My solution is pretty simple and can be explained as follows. Using GROBID, we can extract the list of bibliographical references (mainly, the title of each reference) and associate each one to **a tag** in the article. This creates a second dataset, let's call it *dataset B*. Unlike the first dataset, this dataset contains the title of the references, but not their `arXiv` id.

Now, we want to merge the two datasets and **delete rows** of the *dataset A* which referred to an older version of the paper. For that, we can treat each paper separately.

For each paper, we look at bibliographical references that have the same title **in both dataset** and compare their tags in each dataset. If the *dataset A and B* agree on at least 75% of the tags, then we keep references extracted with S2ORC. Otherwise, we remove the references extracted with S2ORC and simply keep the `arXiv` ids and associate them to references extracted with GROBID which have the same title.

5.4 Results on the CS-CC dataset

Now that all the steps of the algorithm presented in **Figure 6** are ready, we can **combine them together** and check the results of the algorithm on our datasets. It is interesting to see that **every step can be parallelized** since we work on every paper independently at each step of the algorithm.

The CS-CC dataset. We first tested our algorithm on the CS-CC dataset presented in **Section 3**. As a reminder, there are 6.000 papers on this dataset, published between 2010 and 2020 under the category *"Computational Complexity"* of `arXiv`. A bit less than 4.500 papers passed the first step (LaTeX compilation and box extraction) without errors. 3.976 papers

contain at least **one result** and 3.338 of them **contain references** (either internal or external).

With the fully optimized algorithm, we obtained 82.637 results and 46.578 references. Among these references, only 3.960 are references to **external results**. Only 829 of the references results are **found**, scattered between 515 papers.

The extended dataset. Before using the algorithm on the whole `arXiv` dataset (see **Section 6**), I wanted to see what happened if we iteratively extend the dataset by **adding referenced papers** to it. I started with the results of the CS-CC 6.000 paper dataset and filtered all papers which were neither at the source nor the target of any external reference. Then, I add every referenced paper which was not already in the dataset, run the algorithm again, and repeat the operation. After only 5 iterations, there were no new papers to add to the dataset (see **Figure 7**).

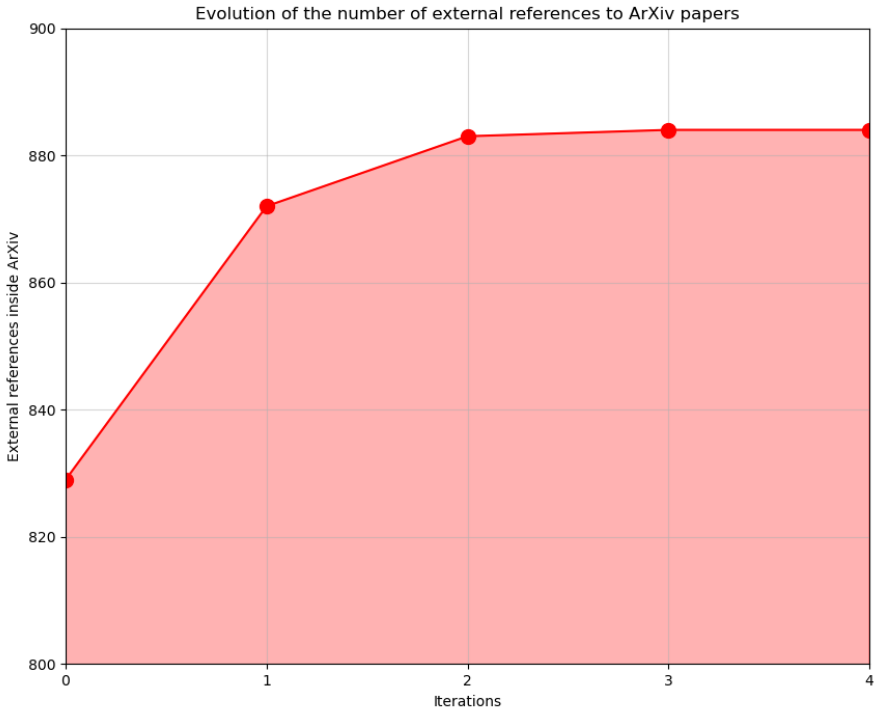

Figure 7: Evolution of the number of external references to other `arXiv` papers **after each iteration**.

From the 6.000 papers of our dataset, I only selected the ones **involved in an external reference** (either as a source or a target). Even after this pruning step, some of these papers are lost due to parsing error when running the script. This gives us less than 700 papers before the first iteration.

The dataset obtained at **the end of the fifth iteration** contains 829 papers, around 21.000 results and 14.600 references, with 2.103 external references and 884 of them with an `arXiv` paper as a target.

I was a bit disappointed that **so few papers** were added to the dataset. The problem probably comes from the fact that we can only add **older and older** papers, because they need to be in the bibliography of some document in the dataset.

Qualitative analysis. Finally, I did **a qualitative analysis** of 20 papers randomly selected in the cs-cc dataset. This helped me find some bugs and update the algorithm. It improved the quality of the results by $\sim 10\%$ (for instance, the number of references found for **the extended dataset** went from 884 to 962).

6 The arXiv Database

Now that we were confident in the theorem and reference **extraction script**, we want to test it on the largest possible database. We chose to run the script on **the whole arXiv database**. This section describes the specific aspects of the dataset used, as well as the quantitative and qualitative analysis of the results of the script.

6.1 The dataset

We downloaded the whole arXiv database (pdf and sources). This dataset contains almost **1.7 million papers** and weighs around 2.8 TB. However, it would have taken too long to run our script on so many papers and a lot of them do not even contain any mathematical result. Consequently, **we removed** every paper which do not contains either a *Theorem*, a *Lemma* or a *Proposition*. I did that using *pdf2text* and a simple regular expression.

After this pruning step, we obtain around 500.000 papers and we are pretty convinced they all contain mathematical results. **Figure** 8 highlights that the number of papers published each year is **increasing exponentially**. We can also see that the proportion of scientific papers containing mathematical results is increasing until 2015. My hypothesis is that it is due to the rise of *deep learning*, with few theoretical papers.

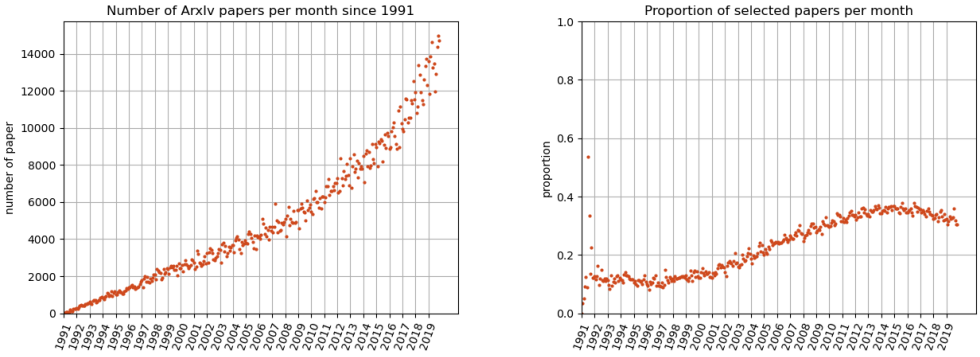

Figure 8: Evolution of the number of papers on arXiv **published** each month since 1991 (*left panel*) and the proportion of them **containing mathematical results** (*right panel*).

I ran the extraction scripts on these 500.000 papers on the RIOC cluster of the *Inria* (which is not easy, given that all the software installed on the cluster is outdated). It took around **one week** to complete. The next

subsections detail the quantitative and qualitative analysis of the results obtained.

6.2 Quantitative Analysis

Results and Reference. We obtained a total of 6.218.044 **mathematical results**, which corresponds to more than 12 results per papers. Moreover, the algorithms counted 4.511.850 **references** to mathematical results, including 848.124 **external references**. The number of results and references for each month logically increases with the number of papers (see **Figure 9**).

An **external reference** is external to the paper in which it appears, not external to the `arXiv` dataset.

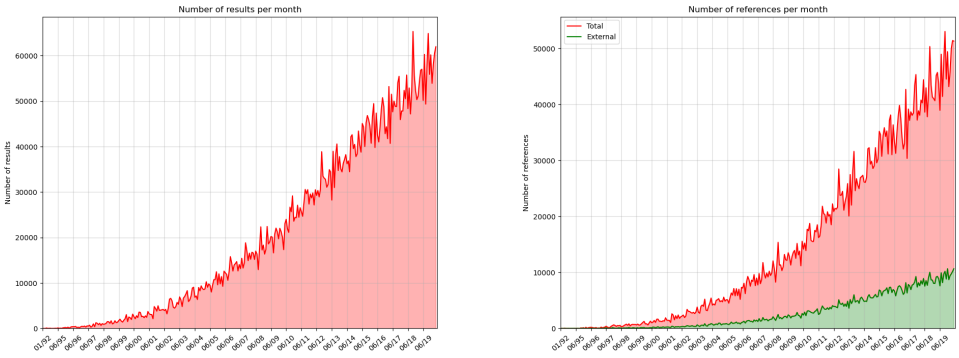

Figure 9: Evolution of the number of results and references to mathematical results in `arXiv` papers for each month since 1991.

Figure 10 shows the proportion of each kind of mathematical results for each month since 1991. Without surprise, *Theorem*, *Lemma* and *Proposition* come first. More surprisingly, *Examples* are less present. There is **no notable change** of the proportion of each kind of result with time.

Citations and Tags. In parallel, the script to detect bibliographical references and to associate tags with them returned 2.215.465 **citations** to other `arXiv` papers (~ 5 per paper), and only 1.808.775 were successfully **associated to some tag** ($\sim 81\%$). Again, we can see on **Figure 11** that these numbers increase with the number of papers.

Moreover, I measured the **confidence indicator** for each month. It corresponds to the proportion of papers for which the *dataset A*, extracted from the S2ORC dataset of SemanticScholar, and the *dataset B*, extracted using GROBID, **agree** on at least 50% of the bibliography. We say that

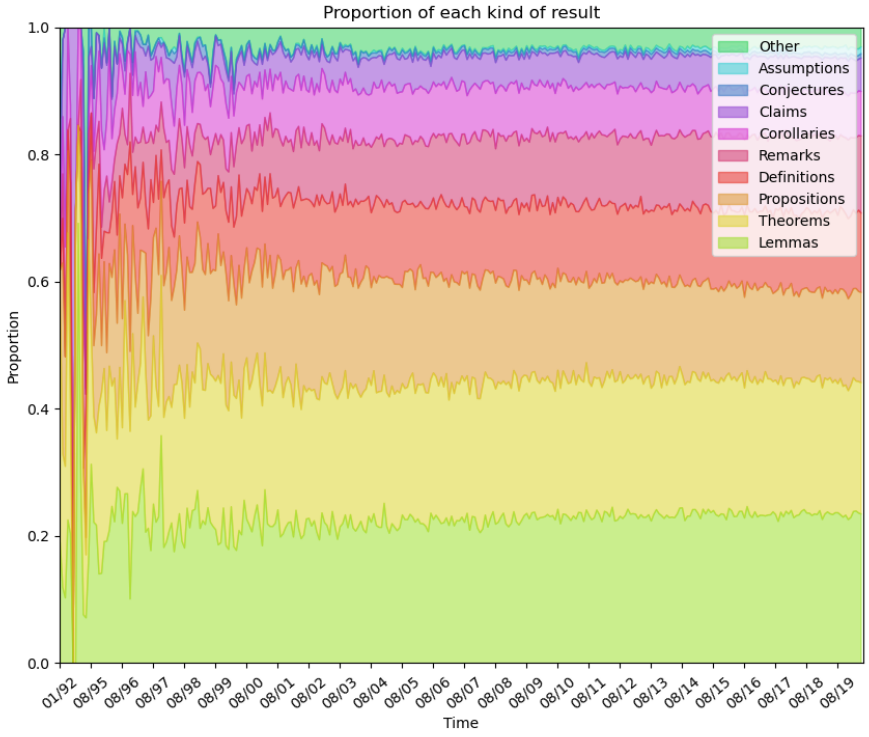

Figure 10: Evolution of the proportion of each kind of results.

they agree on a reference in the bibliography if both datasets associate the same tag to this reference. This indicator swings between 85% and 95% of confidence (see **Figure 12**).

Link results together. If we try to find the papers referenced in the **external references** (the green line in **Figure 8**) using the association between tags and bibliographical references described above (**Figure 11**), only 176.675 of these papers were successfully retrieved.

This corresponds to 21% **of the total**. In 23% of the cases, there are no `arXiv` papers detected in the bibliographical references of the paper. In the remaining 56% of the case, the tag of the reference is not associated to any `arXiv` paper. Among these 176.675 results referenced and for which we found the supposed paper from which they are taken from, only 89.538 of these results were **found inside the paper** (nearly half of them). For the other half which are not found, some results come from papers which were not successfully parsed (see below about errors during extraction) and some

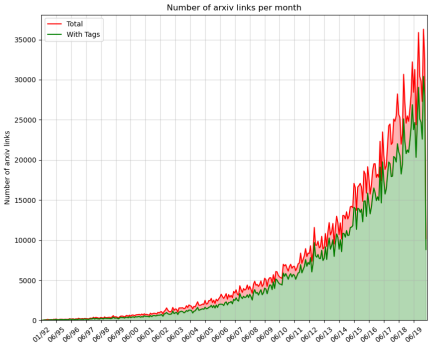

Figure 11: Evolution of the number of `arXiv` papers found in the bibliography of papers published each month.

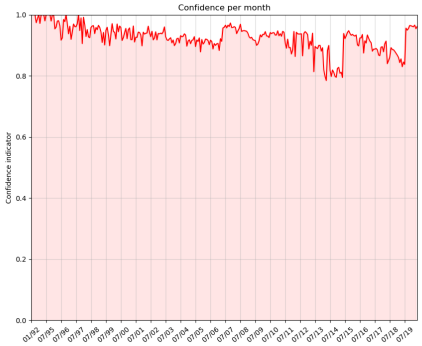

Figure 12: Confidence indicator for bibliographical results since 1991.

others were not found inside the list of mathematical results extracted from the referenced paper. **Table 4** and **Figure 13** sum up all these results.

Internal references	3.663.726	
External references	848.124	100%
No bibliography	202.872	23%
Tag not in bibliography	468.554	56%
Papers referenced not parsed	51.481	6%
Result not found	35.656	4%
Result found	89.538	11%
References (Total)	4.511.850	

Table 4: Summary of the number of references extracted

The graph of results. With the 89.538 references between two mathematical results, I was able to build **a dependency graph of mathematical results**, and a more simple version of the graph by **aggregating results** of the same paper together.

The aggregated version of the graph contains 89.128 paper nodes, and 85.222 reference edges. The graph contains 17.598 connected components and the biggest one has size 35.594, which represents around 40% of the papers nodes. It is interesting to see that there is not a **mega-component** with more than 80% of the nodes, but actually a lot of **micro-components** with less than 10 papers, which is rarely the case in large graphs like this

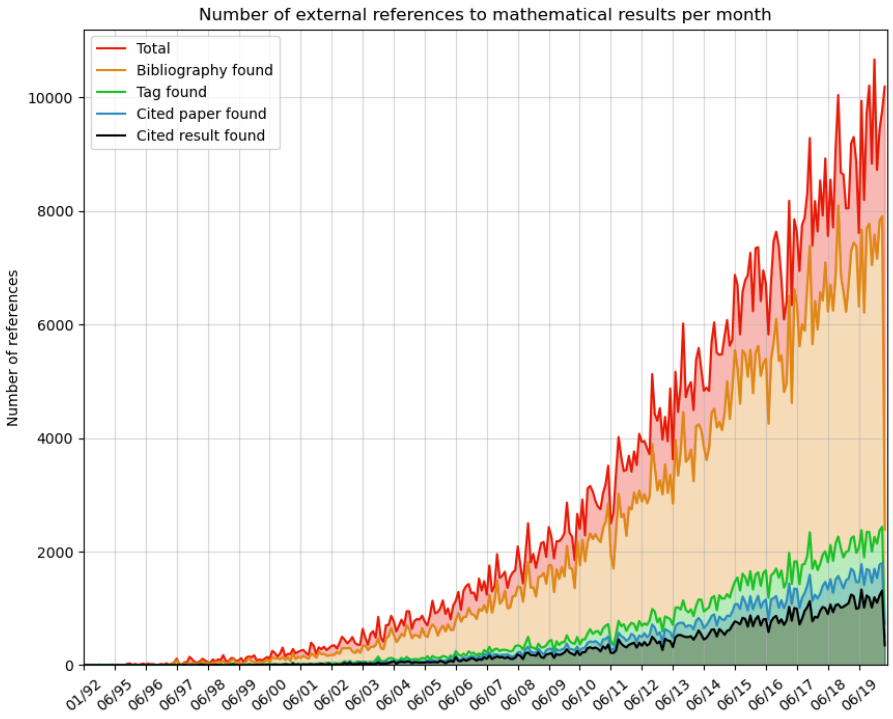

Figure 13: Total number of **external references**, the proportion of them **having a bibliography** in our citation dataset, the proportion having the **reference tag** inside this dataset, the proportion for which the paper containing the **referenced paper** is found and finally, the proportion for which the exact **referenced result** is found. These proportions are shown for each month between 1991 and 2020.

one. An explanation can be that **we are missing** a large amount of papers that are not on `arXiv`.

The **longest path** (when cycles are removed) in this graph is of length 13. However, since it is the aggregated version of the graph, we cannot be sure that the results used as the target of the arriving edge is the same than the results used as the source of the leaving edge. A representation of the biggest connected component with a *Gaussian* noise is shown in **Figure 14**.

I also looked at which results are **cited in the biggest number of papers** and I found that **Proposition 5.3** of *"Stability Conditions On Triangulated Categories"* by *Tom Bridgeland* and **Lemma 5.2** of *"Introduction*

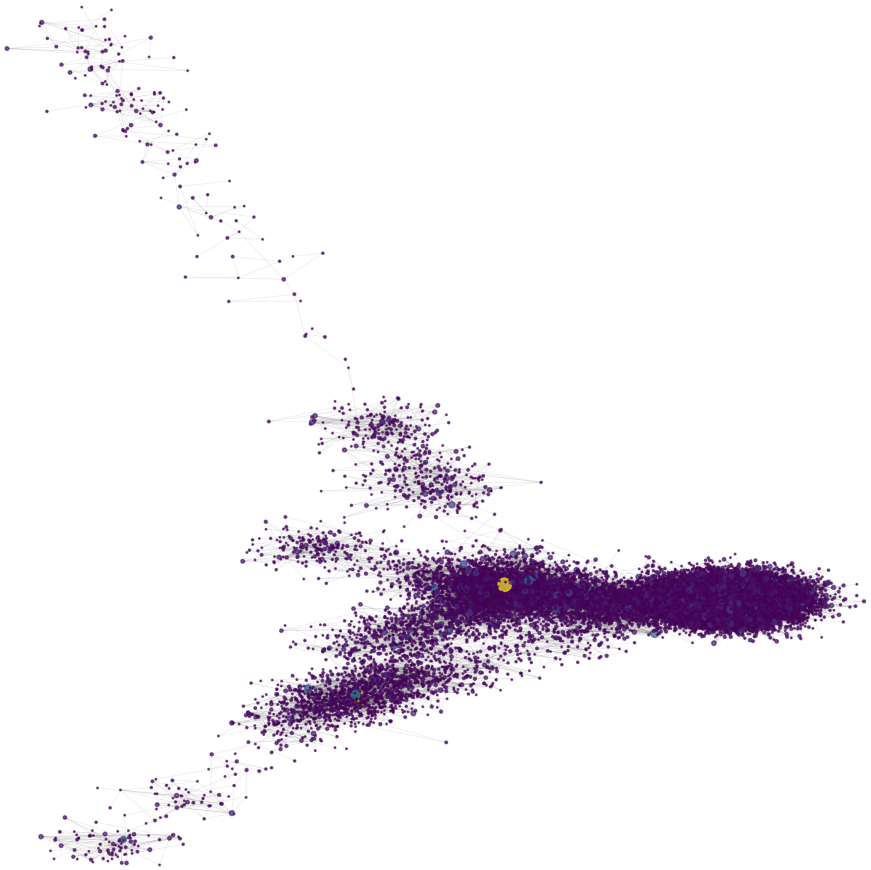

Figure 14: Representation of the biggest connected component of the aggregated graph using *Networkx* **spectral layout**.

to the non-asymptotic analysis of random matrices" by *Roman Vershynin* are ex-aequo with 18 citations each. These superstars of mathematical results are shown in **Figure 15**.

The non-aggregated version of the graph also includes the internal references (i.e., links between the results of the same paper). It contains 2.889.376 nodes and 3.001.424 edges. 97% of these edges are internal references.

However, there is nothing very interesting to say about this graph, because it is very big but mostly **very sparse**. Indeed, most results only cite results **from the same paper**. For instance, the biggest connected component contains less than 2% of all the nodes. Moreover, the longest paths

Proposition 5.3. *To give a stability condition on a triangulated category $\mathcal{D}$ is equivalent to giving a bounded t-structure on $\mathcal{D}$ and a stability function on its heart with the Harder-Narasimhan property.*

Lemma 5.2 (Covering numbers of the sphere). *The unit Euclidean sphere S^{n-1} equipped with the Euclidean metric satisfies for every $\varepsilon > 0$ that*

$$\mathcal{N}(S^{n-1}, \varepsilon) \leq \left(1 + \frac{2}{\varepsilon}\right)^n.$$

Figure 15: Mathematical results cited by the highest number of paper in our database.

contain result from one or two papers, and the most cited results are only cited by results from the same paper (inside very big papers).

Errors and bugs. We also dived into **the different kinds of errors** raised by the first two steps of the script (The LaTeX compilation and the box extraction using PDFAlto). **Figure 16** highlights that the script failed a lot on **old papers** and the **success rate** of the script stabilizes at 80% for paper published after 2010.

As one can see on **Figure 17**, most errors from old papers are due to the script not finding the main file, probably because the version of LaTeX used in the source is not compatible with the version of LaTeX used in our script. Errors on more recent papers are mostly due to PDFAlto (during the box extraction), or because of too many errors during the LaTeX compilation. **Table 5** sums up the sources of errors of these first two parts of the script. Please note that this part of the script **has been improved** since then and the success rate might be higher with the new version.

Success	346.040	75%
LaTeX compilation	82.296	18%
No LaTeX found	4.555	1%
No main file found	41.9983	4%
Memory error	13.384	3%
Other errors	35.656	10%
PDF Alto	31.403	7%
Total	459.739	100%

Table 5: Summary of the errors obtained during the first step of the script.

Finally, among the 346,040 successfully parsed papers, 315,434 (91%) of them contain mathematical results and 258,235 (75%) contain references.

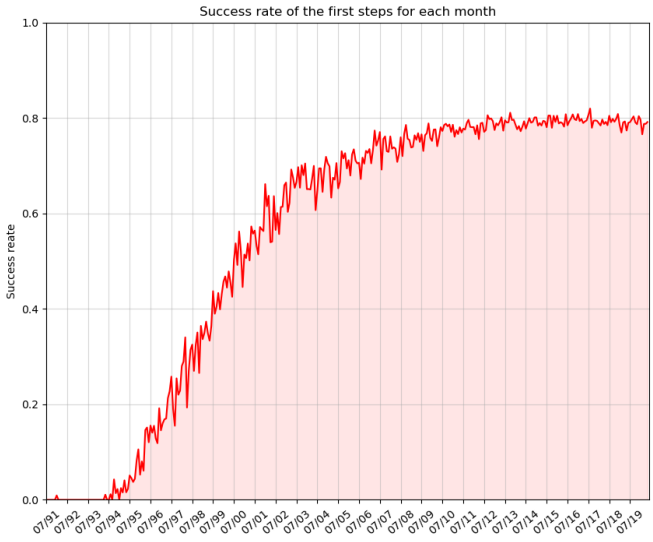

Figure 16: Success rate of the first step of the script from 1991 to 2020.

6.3 Qualitative Analysis

The previous subsection gave us an insight of the number of results and references found by the algorithm described in **Section 5**, and it would be interesting to see **the actual quality** of these results. To get a sense of it, I randomly selected 25 papers and compared the output of the algorithm and the desired output, that I gathered manually. To do so, I read these papers, and I **noted down** every mathematical results and references to mathematical results for each paper. This is the same process that I used in **Section 5** to detect which part of the script could be optimized. **Table 6** sums up the conclusion of this analysis for references.

Mathematical results. There was a total of 604 mathematical results expected (around 24 per paper on average) and 90.6% of them were successfully retrieved. 6.1% were found but their name was not detected. In most cases, we can change the script so that it can detect the name. For instance, if a result has no number (e.g *"Remark"* instead of *"Remark 2.3"*), the name is not detected. Finally, the remaining 3.3% are simply not detected, probably because the authors used a specific command for these results, which is not detected by our LaTeX extraction script. The only false positives for mathematical results are *"Case X"* and *"Step X"* and whether or not these are false positives is up to debate.

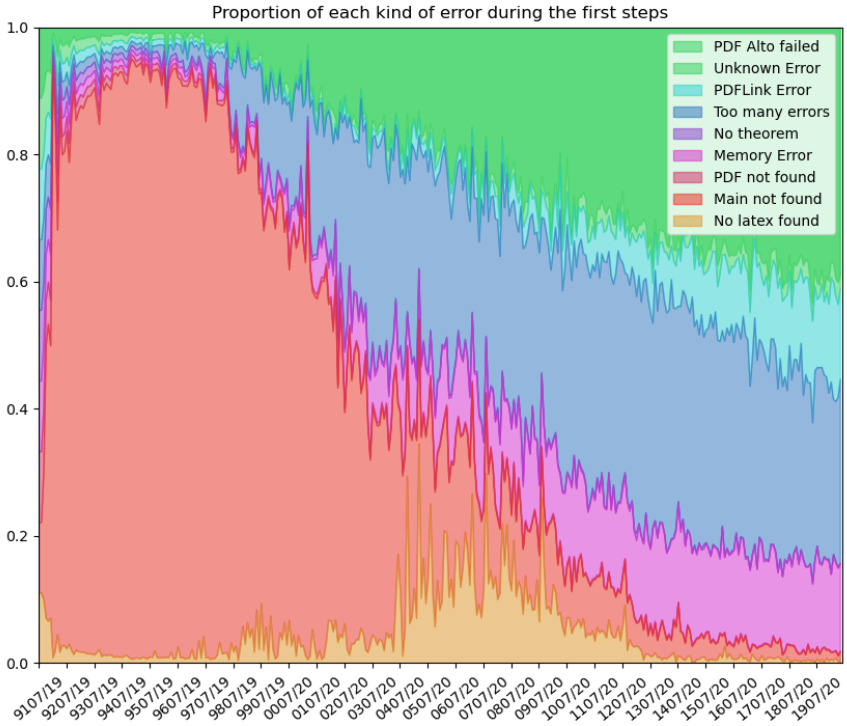

Figure 17: Proportion of the kinds of errors for the first step of the script from 1991 to 2020.

However, the errors for mathematical results are not really interesting to me because they come from parts of the script I did not work on. Let us now discuss the quality of **the extracted references**.

References. The analysis of references does not take account 2 of the 25 papers, because I could not reproduce on my own computer the results obtained on *Inria* clusters. I obtained 345 **internal references** (15 per paper on average) and 49 **external references** (around 2 per paper on average).

The **success rate** for internal (resp. external) references is of 81% (resp. 65%). A lot of errors cannot be solved by my script and are either caused by **the structure of the paper** (e.g., nested proofs) or by errors **during LaTeX compilation**. For instance, *Proofs* are sometimes not detected.

Some errors are direct **consequences of errors on results**. For instance, if the name of a result is not detected, we cannot associate a source name to the reference. Another frequent error is that the authors used an

abbreviation or misspelled a word (for instance, *"Therem 5"* or *"Cor. 2.1"*). I believe that these cases could be solved with some work.

Finally, there are some errors that are **unique** to a paper and can be solved easily. The following errors are gathered as *"Other errors"*:

- Some kinds of results are missing on **Table 3**, namely *Algorithms* and *Assumptions*.
- Sometimes, the hyperlink associated to a reference to an internal result does not point precisely into the box containing the result, causing a bug.
- The algorithm can detect *"Lemmas 2 and 3"* and change it as *"Lemma 2 and Lemma 3"* but cannot deal with *"Lemmas 2, 3 and 4"* (it does not detect *Lemma 4*).

I also obtained **false positives**. Sometimes, internal references are detected as external ones, or parts of the text are wrongfully detected as proof or results. This caused us to have 29 (resp. 9) false positives for internal references (resp. external references). **The precision** is specified in **Table 6**.

	Internal ref.	**External ref.**
Recall	81.4%	65.3%
Not named	2.0%	8.2%
Not detected	16.6%	26.5%
LaTeX failed	6.7%	6.1%
Bad structure	3.2%	4.1%
Abbreviation	0.2%	14.3%
Other errors	6.5%	2.0%
Precision	92.2%	84.5%

Table 6: Qualitative analysis of internal and external references.

7 Future work

This last section presents some of the other works I started during my internship. Some of these ideas are directly related to the extraction algorithm presented in **Section 5**, and others are independent of it.

7.1 Improve the extraction algorithm

The qualitative analysis detailed in **Section 6.3** gave me a lot of ideas to improve the quality of the results. First of all, there is the correction of all the bugs mentioned above, but also the optimization of the references detection algorithm. *Lucas* is also working on improving his part of the script.

Moreover, we will need to find a new way to detect references when we will use *CRF* to extract results directly from the PDF instead of the source files. For instance, a lot of papers do not actually use the `hyperref` LaTeX library. Therefore, since **the detection of external references** is based on hyperlinks to the bibliography, we will not be able to detect them anymore. However, this can be solved by simply detecting tags and brackets instead of links. For instance, we can **train a classifier** to help detecting references to the bibliography (an hybrid between GROBID and our current code).

The algorithm is currently able to detect references which are in the result statement and the text of the proof, but **not in the rest of the document**. This represents a big loss of information, and it might be useful to find a way to extract them too, but also to be able to **associate them to some result** in the paper, using *information extraction* techniques.

7.2 Crowdsourcing

During this internship, I also created some **crowdsourcing tools** in order to label various datasets. In this section, I will quickly present the three tools I built and their goal. Each crowdsourcing platform is divided in two parts: the part in which the user answer the questions, and another part displaying the PDF of the paper, such that the user can directly check inside in case of doubt.

Contextual citations. The goal of the first tool was to gather enough data to train a classifier to **detect the nature of a reference to a mathematical result**, given its context. The idea was to do a work similar to that in [7], but for mathematical results instead of article citations. I divided the possible nature of the reference into 3 categories:

- Statement : When the result is **defined**. *"Theorem 1 : If $a = 6$ and $b = 7$..."*
- Use : When the result is **used**, for instance in a proof. *"...then by Lemma 5 we can conclude..."*
- Discuss : When the result is not used but only **mentioned**, for instance as a comparison. *"...the difference with Proposition 3 is that..."*

However, I stopped working on reference disambiguation, and I did not have time to use the collected data to build a classifier.

External or Internal? Another classifier I wanted to train was a classifier able to detect, given a sentence containing a reference to a mathematical result and a tag, whether the reference **is associated to the tag or not**. Then, I could use it in the algorithm presented in **Section 5** to **disambiguate** internal and external references.

For instance, in the sentence *"We use Theorem 3 from [4] and Lemma 1.5 to..."*, *"Theorem 3"* should be associated to *"[4]"*, but not *"Lemma 1.5"*, since it is an internal reference. The tool I created shows a sentence with the reference and the tag highlighted, and asks if it is indeed an external reference.

After I went through several hundred sentences myself with this tool, I obtained **a recall** of 91%. In the end, I did not use it to train a classifier, but it helped me a lot to understand how to **eliminate some false positives** directly by modifying the code.

Parsing of definitions. One of the goals of TheoremKB is to create a **structured knowledge base**. Consequently, there is some work to be done on the results of the algorithm presented in **Section 6** before we can obtain the right data structure. Ideally, we want a structure close to what already exists for other knowledge bases, like Wikidata or DBpedia, such that it is easy to **navigate inside the graph** of results, but also **user-friendly**, such that information are easily accessible and understandable to anyone.

For instance, it would be great to be able to extract **what is defined** in a *Definition*, so we can detect further references to it. The goal of this last tool is actually to manually gather a dataset with data and labels, and build an NLP classifier able to automatically detect what is defined in a *Definition*.

The user simply has to highlight which part of the definition he thinks correspond to the defined term. I still think it can be useful, but the information on what is defined is often **unclear and insufficient**: there can be

several terms defined in one definition, it can also be a notation and not a definition.

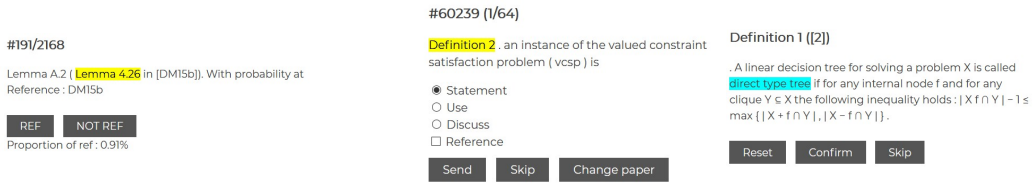

Figure 18: Screenshots of my crowdsourcing projects. Links : Contextual citations, External or Internal ?, Parsing of definitions.

Conclusion

Finally, after these 6 months of this internship, we successfully **extracted a connected graph of mathematical results** from the whole `arXiv` corpus of scientific publications. I optimized the extraction algorithm such that it takes the less possible time (for instance, by using *KD-Trees*) and return results with the best precision and recall (by dealing with the various particular cases). During this project, I also implemented some neural networks and other NLP methods for *Theorem matching*.

Thanks to this project, I also **familiarized myself** with a large family of tools for information extraction and retrieval in scholarly documents: GROBID,PDFALTO, the *dissem.in* API, the *SemanticScholar* API and a lot of python libraries (`TexSoup`, `KDTree`, etc.).

Some of the work done during this internship can be found on our Github repository [1]. It mainly contains the code of the algorithm presented in **Section 5**.

This internship is one of the first steps of the much bigger and promising THEOREMKB project. Hopefully, it will be helpful to have a first idea of the density of the graph of mathematical results, even with simple methods such that those we are using for the moment.

Aknowledgments

I learnt a lot during this internship, and I want to thank my advisor, *Prof. Pierre Senellart*, for guiding me during these five months. I also want to thank *Lucas Pluvinage*, which always had great ideas when I was stuck on the project. Finally, thanks to my friends and my advisor for helping me proofreading this report.-

References

[1] Github of the project. https://github.com/PierreSenellart/theoremkb.

[2] Grobid. https://github.com/kermitt2/grobid, 2008–2020.

[3] Pdfalto. https://github.com/kermitt2/pdfalto, 2008–2020.

[4] Bentley, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM 18*, 9 (Sept. 1975), 509–517.

[5] Bhagavatula, C., Feldman, S., Power, R., and Ammar, W. Content-based citation recommendation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (New Orleans, Louisiana, June 2018), Association for Computational Linguistics, pp. 238–251.

[6] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* (2015).

[7] Cohan, A., Ammar, W., Van Zuylen, M., and Cady, F. Structural scaffolds for citation intent classification in scientific publications. *arXiv preprint arXiv:1904.01608* (2019).

[8] Ganesalingam, M., and Gowers, W. T. A fully automatic problem solver with human-style output. *arXiv preprint arXiv:1309.4501* (2013).

[9] Jeong, C., Jang, S., Park, E., and Choi, S. A context-aware citation recommendation model with bert and graph convolutional networks. *Scientometrics 124*, 3 (2020), 1907–1922.

[10] Kanakia, A., Shen, Z., Eide, D., and Wang, K. A scalable hybrid research paper recommender system for microsoft academic. In *The World Wide Web Conference* (2019), pp. 2893–2899.

[11] Kohlhase, M., and Franke, A. Mbase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation 32*, 4 (2001), 365 – 402.

[12] Liu, H., Kong, X., Bai, X., Wang, W., Bekele, T. M., and Xia, F. Context-based collaborative filtering for citation recommendation. *IEEE Access 3* (2015), 1695–1703.

[13] Liu, Y., Bai, K., Mitra, P., and Giles, C. Tableseer: Automatic table metadata extraction and searching in digital libraries. pp. 91–100.

[14] Lo, K., Wang, L. L., Neumann, M., Kinney, R., and Weld, D. S. S2orc: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 4969–4983.

[15] Lopez, P. Grobid: Combining automatic bibliographic data recognition and term extraction for scholarship publications. In *Research and Advanced Technology for Digital Libraries* (Berlin, Heidelberg, 2009), M. Agosti, J. Borbinha, S. Kapidakis, C. Papatheodorou, and G. Tsakonas, Eds., Springer Berlin Heidelberg, pp. 473–474.

[16] Luu, K., Koncel-Kedziorski, R., Lo, K., Cachola, I., and Smith, N. A. Citation text generation. *arXiv preprint arXiv:2002.00317* (2020).

[17] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[18] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 1532–1543.

[19] Radev, D., Muthukrishnan, P., and Qazvinian, V. The acl anthology network corpus. vol. 47.

[20] Rao, D., and McMahan, B. *Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning.* O'Reilly Media, 2019.

[21] ray choudhury, S., Tuarob, S., Mitra, P., Rokach, L., Kirk, A., Szep, S., Pellegrino, D., Jones, S., and Giles, C. A figure search engine architecture for a chemistry digital library. pp. 369–370.

[22] Saier, T., and Färber, M. Bibliometric-enhanced arxiv: A data set for paper-based and citation-based tasks. In *BIR@ECIR* (2019).

[23] Shen, Z., Ma, H., and Wang, K. A web-scale system for scientific knowledge exploration. In *Proceedings of ACL 2018, System Demonstrations* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 87–92.

[24] Wei, L., and Deng, Z.-H. A variational autoencoding approach for inducing cross-lingual word embeddings. pp. 4165–4171.