Impact of the document class in the automatic extraction of mathematical environments in the scientific literature

Antoine Gauquier

I.M.T. Nord Europe & École normale supérieure

Impact of the document class in the automatic extraction of mathematical environments in the scientific literature

Report of the "Projet de Fin d'Études" (PFE - CI3)

Antoine GAUQUIER – FISE 2023 – University year 2022/2023

Supervisor at ENS :
Pierre Senellart

Supervisor at IMTNE :
Vincent Itier

Acknowledgments

I first want to thank my internship supervisor at ENS, Pierre Senellart, who gave me the opportunity to do this internship. He has always been available when I needed it. I always felt like working with him, and not for him, in the sense that he considered me as a colleague, even though I didn't have an important experience of research. I thus want to thank him again.

Then, I also want to thank my internship supervisor at IMT Nord Europe, Vincent Itier, who were always available when needed and ensured that the internship was going well at anytime.

It is important for me to thank the entire Valda team, which warmly welcomed me and integrated me quickly in the team. I also want to particularly thank Shrey Mishra, for the trust he gave me by getting me involved in his research work. I also want to thank him for the long discussions about good practices and hottest topics of machine learning we had together, that I will not forget and that will definitely help me in my future work.

The administrative assistants of the team and of the IT department also helped me a lot in the organization of my work, especially for the missions out of the office I had the opportunity to do, especially the one abroad. I thus want to thank them for their availability for that.

Finally, I want to thank my family, and also my girlfriend, for helping me making this internship a success. I also want to warmly thank them for all the support they gave me during my studies.

Contents

List of Figures

List of Tables

Abstract and keywords

The purpose of this report is to summarize and present my internship, its context within the Valda team from École normale supérieure, the tasks I have carried out there and the learning and benefits I have gained from it, including my career plan. This internship is part of my last year of engineering at I.M.T. Nord Europe called "Projet de Fin d'Études".

This 24-week internship has led to the demonstration of the relevance of the use of the document class when extracting mathematical environments from the scientific literature. It is part of the TheoremKB project, which aims at creating and exploiting knowledge bases of mathematical results using this same scientific literature. This work resulted in the creation of several deliverables, including scientific articles and repositories to reproduce the experiments conducted.

I have also developed relational skills related to the research context, as well as scientific skills, both theoretical and practical, in the field of data science. Finally, this experience definitely confirmed my wish to continue my engineering studies by a PhD, which I will do in this same team, and with the CEDAR team.

Keywords: I.M.T. Nord Europe – École normale supérieure – Research internship – Artificial intelligence – TheoremKB – Document class – Information extraction – Deep learning – Computer vision – Multimodality

---

Ce rapport a pour but de synthétiser mon stage, son contexte de réalisation au sein de l'équipe Valda de l'École normale supérieure, les tâches que j'y ai effectuées ainsi que les acquis et bénéfices que je tire de ce dernier, incluant mon plan de carrière. Ce projet de fin d'études s'inscrit dans le cadre de ma dernière année de cycle ingénieur à l'I.M.T. Nord Europe.

Ce stage de 24 semaines a permis de démontrer la pertinence de l'utilisation de la classe de document pour extraire les environnements mathématiques dans la littérature scientifique. Il entre dans le cadre du projet TheoremKB, qui vise à créer et à exploiter des bases de connaissances de résultats mathématiques en utilisant cette même littérature scientifique. Ces travaux ont abouti à la création de plusieurs livrables, dont des articles scientifiques et des dépôts permettant la reproduction des expériences réalisées.

J'ai également développé des compétences relationnelles liées au contexte du monde de la recherche, ainsi que des compétences scientifiques, tant théoriques que pratiques, dans le domaine de la science des données. Enfin, cette expérience a définitivement confirmé mon souhait de poursuivre mes études d'ingénieur par un doctorat, que je réaliserai dans cette même équipe, et au sein de l'équipe CEDAR.

Mots clés : I.M.T. Nord Europe – École normale supérieure – Stage de recherche – Intelligence artificielle – TheoremKB – Classe de document – Extraction d'informations – Apprentissage profond – Vision par ordinateur – Multi-modalités

Global scheduling


Total duration of the internship (24 weeks plus one extra week)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Organizing the work by steps
Document Class Inference
Statistical study
CNN-based classifiers
Writing of the paper
Submission of the paper
Sequential Modelling
State-of-the-art study
Preparation of the data
Creation, training and evaluation of models
Writing of my part in the paper
Submission of the paper
Study: Impact of Document Class
Data preparation
Creation, training and evaluation of the models
Achievement of the study
Oral Defense and Written Report
Preparation of oral defense
Oral defense
Writing of written report
Submission of written report

Introduction

As part of my third year of engineering studies at I.M.T. Nord Europe, I had the opportunity to complete a 24-week internship at the *École normale supérieure*, and more particularly in the Valda team. It took place from January 30 to July 21, 2023, and constitutes the last internship of my studies as an engineer, entitled Project of Termination, therefore having to validate my ability to practice the profession of engineer.

This internship must also be an opportunity to complete our competency profile in one or more areas that constitute our interest, and validate our professional project and lead us to our first job. With the ambition of continuing my engineering studies towards research, particularly in the fields of artificial intelligence and data science, I naturally turned to a research team in these fields. Pierre Senellart, head of the Valda team which is working in these fields, has agreed to supervise me as part of the research project included in *Télécom Paris*' A.I. option, which I followed in the context of an academic exchange. This one went well, and he welcomed me back to his team as part of this internship.

The Valda team is working on issues related to the management and exploitation of data on a large scale, in particular concerning data produced by humans. It was in this context that the TheoremKB project was set up. It aims to create and exploit knowledge bases of mathematical results from the scientific literature, which thus constitute human-produced data. In order to build that foundation, one already has to be able to extract that knowledge. This is why it is necessary to focus on the task of automatic extraction of mathematical environments in the scientific literature. This statistical learning task therefore uses the PDF rendering of these scientific contents as input data.

These scientific contents, for most of them, are materialized in the form of scientific articles, accepted or at least submitted to journals or conferences for presentation to the scientific community. However, we know that for many years now, these articles have been, when they are associated with fields such as mathematics, computer science or theoretical physics, mainly written in the LaTeX language, allowing the composition of scientific documents, in the form of a compiled language. In order to standardize the format of different articles accepted in the same conference or journal, they use a *document class*, which is actually LaTeX code defining standards for structuring documents. Thus, authors can directly write their articles with the right class of document, and the conference only has to concatenate the selected articles.

The subject of this internship is then part of a simple observation: as the classes of documents structure the content of scientific articles, we can make the hypothesis that they should also structure the mathematical environments they contain, if they contain any. Following this reasoning, the task of automatic extraction of mathematical environments would be simplified if one knew the document class used. This is why this internship aims at studying the impact of the document class in the automatic extraction of mathematical environments in the scientific literature.

In order to carry out this study, this report is divided into three parts. In the first one, I present in detail the context in which the internship takes place, as well as the environment in which I have evolved. In the second part, I present the concrete realization of this study, in three phases: the task of automatic inference of the document class, since to study its impact, one must already be able to identify it; the task of automatic extraction of mathematical environments, in order to have a baseline used for comparison, and finally the fusion of the first two phases in order to carry out the concrete study of the impact of the document class. Finally, I will conclude this report by presenting the relational and technical skills developed during this internship, as well as the future considered stage for my career.

Chapter 1

Environment and Context of the Internship

In this chapter, I am going to present the environment in which the internship took place, as well as the context that set the boundaries of the work I achieved. In the first section, I will present the École normale supérieure, which is both a school and a research center through a set of different teams, working on different subjects. In the second subsection, I will present one of these teams, which is the Valda team, that focuses on the enhancement of value from data. In the third section, I will present on of the projects on which the Valda team is working, called TheoremKB, which aims at creating and exploiting knowledge bases of mathematical results from the scientific literature.

1.1 The *École normale supérieure*

The *École normale supérieure* (ENS) is the organization that welcomed me for this internship. It is a French public higher education institution founded in 1794. More specifically, it is one of the four *"Écoles Normales Supérieures"*, along with the ones of *Paris-Saclay*, *Lyon* and *Rennes*. It is also one of the component institutions of the *Université Paris Sciences et Lettres* (PSL), which brings together, in addition to the ENS, the *Université Paris-Dauphine*, the *École des Mines de Paris*, the *ESPCI*, *Chimie ParisTech*, the *Conservatoire National d'Art Dramatique*, the *École Nationale des Chartres* and the *École Pratique des Hautes Études*.

The ENS recruits its students through a number of different channels. The historical one recruits student called *"normaliens élèves"* via a competitive entrance examination that students take after two years of preparatory classes (*Classes Préparatoires aux Grandes Écoles* – CPGE). ENS students recruited through this channel have a special status, as they are trainee civil servants (*"fonctionnaire stagiaire"*). They are paid during their education, and in return they sign a ten-year public service contract, in which they must work for the State (their studies being counted as part of the contract). A more recent channel, aimed at diversifying student profiles, is the one that recruits students under the status of *"normalien étudiant"*. These students do not sign a public service contract and are therefore not paid by the State during their studies. However, they follow the same training as their fellow civil servants.

In addition to these two recruitment channels which award a diploma specific to the ENS, there are two other channels which award other diplomas. Some students take courses at the ENS as part of the programs offered by PSL University. They therefore obtain a PSL University diploma associated with their training, and generally take courses in other PSL component institutions. They can be recruited with a university diploma (this is the case for Master's students, recruited with a Bachelor's degree), but also without some (this is notably the case for the *Cycles Préparatoires d'Études Supérieures* - CPES). Finally, a number of doctoral students are recruited in the ENS research laboratories. At the end of their PhD, they obtain a doctorate, awarded by PSL University through a doctoral school, but prepared at the ENS (the preparatory institution is specified).

The ENS offers a wide range of career opportunities. The majority of students recruited through the traditional channels have historically gone into teaching or research. Some examples include teaching in the CPGE, teaching and research in higher education establishments, and research in national research institutions. However, other students are interested in research and development departments in the industry. This is a growing trend in the sciences, particularly in computer science, especially in engineering companies. Last but not least, other students opt for careers in the civil service or the military, the ENS diploma giving in some cases preferred recruitment channels.

Finally, the École normale supérieure is not only a teaching institution, but also a research centre, split into two sections: sciences and humanities. The school hosts 15 departments, divided into these two main sections. The science section is composed of seven sections: the *Département de Mathématiques et Applications* (DMA), the *Département de Physique* (FIP), the *Département de Biologie* (BI), the *Département de Géosciences* (TAO), the *Département d'Études Cognitives* (DEC), the *Département de Chimie* and the *Département d'Informatique* (DI). Each of these departments has a number of research teams with a variety of themes. For example, the DI is divided into ten teams, focusing on security and reliability, machine learning and complex data, and algorithms and their analysis.

I'm now going to introduce the Valda team, which is the one in which I did my internship.

1.2 The Valda Team and its Environment

The Valda team, which is part of the *Département d'Informatique* (UMR 8548), focuses on the research themes of machine learning and complex data. It was created in 2016 by Pierre Senellart. The logo of the team can be found at Figure 1.1.

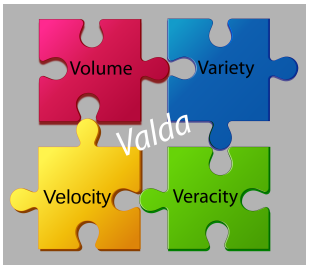

Figure 1.1: Logo of the Valda team (including the "four V's" of big data management)

It is a team jointly supported by the ENS, CNRS, and Inria, which is a french public institute for research in computer science. Therefore, the permanent members of the team are both associate of full professors from ENS and researchers from CNRS and Inria. The team's aim is to extract value from data, hence the name Valda. An importance is specifically given to data produced by humans.

More specifically, the team's research focuses on three areas. The first concerns the foundations of data management. The data processed by the team is often massive, heterogeneous and of diverse origins. The systems used to store and interrogate this data must therefore be optimally efficient.

The second area of research concerns uncertainty and data provenance. As the team works with data that is often human-centric, it is often uncertain, and it is important to be able to quantify its reliability. This can be done directly at the source, but also throughout the data processing process: that's why it's important to be able to know where the data comes from. The aim of the *ProvSQL*[1] project led by Pierre Senellart is to develop a data management system that supports the provenance of data, making it possible to keep a record of its origin, while at the same time making it a point of honor to retain important efficiency properties. Work is also being carried out on probabilistic data management systems, in particular as part of

[1] `https://github.com/PierreSenellart/provsql`

theses supervised by the team.

The last axis of research concerns Personal Information Management Systems. It corresponds to systems that allow users to integrate multiple of their own data. Usual use-cases can be e-mails, contacts, calendar, web searches, content on social media, travel information, and so on. The aim of these systems is to be able to answer complex queries, to various services depending on the data that the user wants to integrate.

From these axes, different research works are carried about by the different team members. The permanent members are also working with students, either under the form of an internship (this is my situation), or under the form of a PhD contract of (usually) three years, where the PhD students are achieving research work related to the expertise field of the concerned permanent members. This is made possible thanks to different funding sources. Fundings can also be obtained through projects. This is the case for the *CQFD* project[2], the *DESCARTES* project[3], the *Dissemin* project [4] or the *PR[AI]RIE* project[5].

Research work or projects then lead to the generation of different results. These may take the form of scientific articles submitted to conferences or journals, with the aim of sharing what has been done with the various associated communities, or in the form of concrete deliverables, such as software, code or models, made publicly available. Even if this work is not to be evaluated, external commissions regularly checks that what is achieved within the team respects certain criteria, and that the team complies with ethical and responsible practices of research. For instance, the team is evaluated every five years by the *"Haut Conseil de l'évaluation de la recherche et de l'enseignement supérieur"* (HCERES), and must also, like all Inria teams, make publicly available an activity report every year.

In addition to research work, the team also has other responsibilities. First, most permanent members, as well as some PhD students, teaches computer science, either directly within the DI, or in the various PSL programs (in particular in the IASD master's program, as well as in the CPES in partnership with the *Henri IV* and *Louis-Le-Grand* high-schools). It is also partly responsible for the selection and recruitment of students for the ENS, via the various competitive examinations. Finally, another important aspect of the team is the sharing of the research work, between the members of the team of course, but also with members of the community from different teams or even different countries. This is why the Valda team regularly holds seminars where researchers present their work and then discuss it. This is the opportunity to discuss and think about various subject, which definitely help the different researchers improve their work.

I am now going to present the work carried out in the scope of the project *TheoremKB*, which is part of the *PR[AI]RIE* project presented above.

1.3 The TheoremKB Project

The subject of my internship is part of the *TheoremKB*[6] project. This one aims at developing and exploiting *knowledge bases* of mathematical results, directly extracted from the literature. Concretely, we want to be able to exploit the content of scientific articles presented in different formats (mainly PDF, but we also exploit the LaTeX source code of the article), in order to extract mathematical content that we want to be able to exploit at a later stage, which we call *knowledge*. One may want to extract the content itself directly (for example, to identify the mathematical environments corresponding to proofs or theorems and thus extract the content of these environments), but one may also want to extract information relating to this content (for example, a description of the mathematical environments extracted: what is the domain associated with the content, what is its type, or what other articles the author(s) drew inspiration from to generate it). A presentation of the various tasks involved in the project is available in Figure 1.2.

[2] `https://www.lirmm.fr/cqfd/`
[3] `https://descartes.cnrsatcreate.cnrs.fr`https://descartes.cnrsatcreate.cnrs.fr
[4] `https://dissem.in/`
[5] `https://prairie-institute.fr/`
[6] `https://github.com/PierreSenellart/theoremkb`

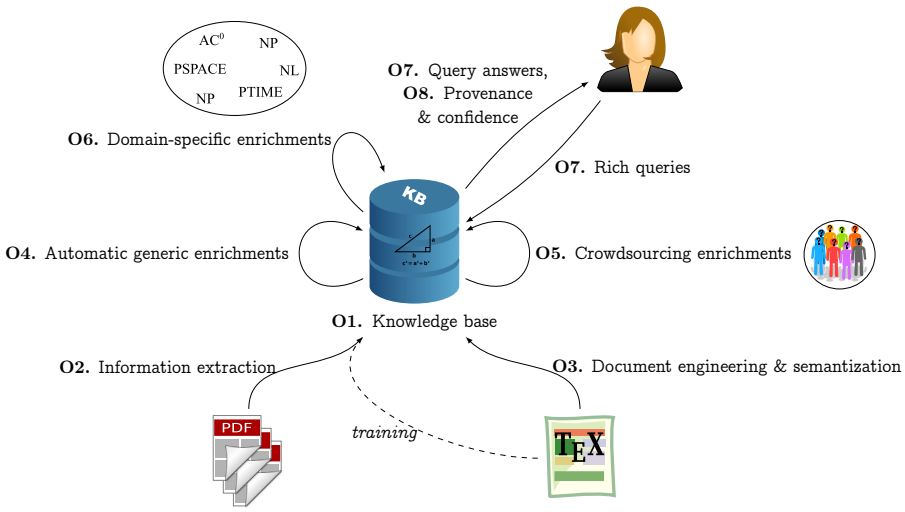

Figure 1.2: Presentation diagram of the TheoremKB project

The central element of this project corresponds to the knowledge base that we are seeking to exploit, represented by item **O1.** To do this, we need to build this base and populate it. This is why we need to extract knowledge from the raw data for the database. This extraction is represented by items **O2.** and **O3.** Respectively, item **O2.** corresponds to the task of extracting information from the PDF rendering of scientific articles, whereas item **O3.** starts from the LaTeX source code of these scientific articles and should enable document engineering (annotation, for example), as well as the semantisation of this content. The LaTeX sources are also exploited to train the machine learning models that we use to achieve item **O2.**, and this will be presented further in this report, in the next chapter.

A series of other tasks can then be carried out, based on this extraction, with the aim of improving or enriching the knowledge base. Item **O4.** should enable the knowledge base to be enriched automatically, through automatic reasoning. An example of application is as follows: if extracted theorems and proofs were to be transformed into logical statements, then it would be possible through automatic inference mechanisms to deduct new mathematical results from several already existing ones in the base. Item **O5.** envisages enrichment by crowd-sourcing. This involves allowing users, or even experts (in this case, the authors of scientific articles), to contribute to and enrich the system. We can imagine, for example, the possibility of annotating or adding information to certain elements, in particular concerning information relating to extracted knowledge, which was presented as an example for item **O3.**. Finally, item **O6.** presents another way of enriching the base, by adding some domain-specific information. This could correspond for instance, as the diagram suggests, to classify specific mathematical results depending on the field they refer to: the theoretical computational complexity family for algorithms is a relevant example.

Once the knowledge base is populated and enriched, it is ready to be exploited. For this, we need to make it searchable, especially by non-expert users. This is represented by item **O7.**, which has two aspects. On the one hand, the ability to build rich queries, and thus support a certain number of elementary and usual operations on databases (and by extension, knowledge bases), while ensuring a certain efficiency in their processing. On the other hand, we need our knowledge base to be able to provide correct answers, in an efficient way. Moreover, given the fact that we want to make the base usable by non-experts, we might want to develop more ready-to-use way to interrogate the base. This could be achieved by making simple queries with just keywords about area, subject or even authors. This requires to have reliable and complete information about the mathematical results that are extracted. Finally, item **O8.** focuses on the provenance and confidence aspect about the results we are extracting. The uncertainty could come from two sources: the data itself, as we might want to feed our base with PDFs that we would directly crowd from the web, and also from the extraction systems, that might do some mistakes. This last item integrates other works

from the team that were presented above, especially the ProvSQL project. An important example of the use of provenance is the ability to generate dependency graphs between extracted mathematical results, in particular by exploiting the bibliographies and references used. This is something that has already been studied in the literature, but whose graph is generated at the scale of an entire article, and not at the scale of the mathematical results present within it.

For the moment, the TheoremKB project is still in its early stages of development. We are focusing on items **O2.** and **O3.**, i.e., everything to do with extracting information that will then be used to populate the knowledge base. However, as mentioned above, other work within the team is already underway, in particular on theoretical considerations concerning items **O7.** and **O8.**. A PhD student (Shrey Mishra) and a post-doc (Shufan Jiang) are actively working on **O2.** and **O3.**, and two PhD students (Baptiste Lafosse and Pratik Karmakar) on **O7.** and **O8.** indirectly.

My internship falls within the scope of the extraction task, and more specifically on improving the extraction methods already developed in the team. This will be developed in the next chapter.

Chapter 2

Realization of the Work

In this chapter, I am going to present the different steps that led me to answer the problem stated in this internship. Since the aim is to study the impact of the *document class* in the automatic extraction of the mathematical environments in the scientific literature, we need to carry some preliminary work before actually do the study. First, since we want to study the impact of the *document class*, we need to actually have this *document class* at our disposal, in an automatic manner. This will be presented in the first section. Then, to know if the *document class* improves or not the extraction of mathematical environments, we need to set a baseline to be compared to. This baseline is presented and improved in the second section of this chapter. After that, we present in section three the study of the impact of the *document class*, through the use of different architectures. Finally, we present in the last section some other tasks that were carried out during the internship, that are not directly related to the work presented in the three first sections.

2.1 Automatically Inferring the Document Class of a Scientific Article

As mentioned above, the first step to consider in order to solve the problem of this internship is to succeed in automatically inferring what is called the *document class*. In the first subsection, we present what is this *document class*, how we think we can infer it, and on which data we will conduct our experiments. In the second subsection, we justify the use of a geometrical approach through the use of five simple features, that we statistically describe thanks to the dataset we presented. Then, we move into a more efficient modelization, that will solve the inference task, by using a deep-learning approach, called *Convolutional Neural Networks*, presented in the last subsection.

2.1.1 Presentation of the Task and of the Data

When authors wish to have their work published in order to make it visible to the scientific community, they write scientific articles in which they summarize their work. These articles are intended to be submitted to proceedings of conferences or journals, which the authors target according to the themes of their research. Examples of associations or societies with which conferences and journals are associated are shown in Figure 2.1. Given that these conferences or journals will publish different articles written by different researchers, it is necessary to harmonize and standardize the form of these articles. This is where the *document class* comes in.

Figure 2.1: Example of three different associations and societies that have their own document class(es)

This *document class* corresponds to the content specified in the `\documentclass` command when writing the code in LaTeX that will be used to generate a PDF of the scientific article. LaTeX is now widely used in fields such as physics, mathematics and computer science. It has the advantage of facilitating the formatting of both textual and multi-media content, as well as allowing the use of scripts that can define new commands, written in the same language, to simplify the formatting of the content. This document class therefore corresponds to the writing of such a script. Our interest in this class of document lies in the fact that it structures the content of the article, and therefore, a priori, also structures the mathematical environments that we are trying to identify.

The Dataset

As we wish to train a supervised learning model to solve this task, we need access to a large corpus of scientific articles whose document class is known. The easiest way to do this is to have access to the LaTeX source code used to generate these articles. This is why we have chosen the *ArXiV*[1] open archive of scientific articles. This is a platform on which a large number of scientific articles in various scientific fields (mathematics, physics, astronomy, theoretical and applied computer science, etc.) are published, along with the LaTeX source code used to generate them where applicable.

We chose to extract all papers published in the platform in the year 2018, through arXiv's bulk access from AWS[2]. We chose this year in order to have articles that represent a quite recent set of document classes. We ended up extracting 119 087 papers. However, only 98 713 of them could be exploited. In fact, for some of them, the LaTeX source is missing, for example in the case where the paper was written using word processing software such as Microsoft Word, making it impossible to infer any document class. For others, document class extraction was ambiguous or impossible (e.g., some authors use Plain TeX instead of LaTeX or redefine the `\documentclass` macro), or the extraction of information used in the modelizations was not possible (because of failures of pdfalto[3] or of the bitmap rendering). A diagram that summarizes these filtering operations is shown in Figure 2.2.

The *Ground Truths*

Within these remaining 98 713 documents, we identify more than 1 200 different document classes, based on their names. But we observe that an overwhelming majority of papers correspond to only a few classes. Indeed, the first 20 classes represent 90% of the papers, and more than half of document classes are only used in one paper. In order to be able to use statistical learning methods, we need enough data for each class so that the models can be properly trained. We therefore impose a criterion of data representativeness, namely that each class kept must contain at least one thousandth of the data (thus appearing in around one hundred articles). 44 document classes satisfy this criterion. In addition, we rename the class `article`, the most commonly used class, to `other` and mapped all documents whose class was not among the 44 to that class.

The reduced list of 44 classes still contain very similar classes (as can directly be observed by name similarities). For example, classes `aastex6`, `aastex6` and `aastex62` (which are document classes of the AAS, presented in Figure 2.1); `ieeeconf` and `ieeetran`; or `amsart` and `amsproc` (document classes of the

[1] `https://arxiv.org/`
[2] `https://info.arxiv.org/help/bulk_data_s3.html`
[3] `https://www.loc.gov/standards/alto/`

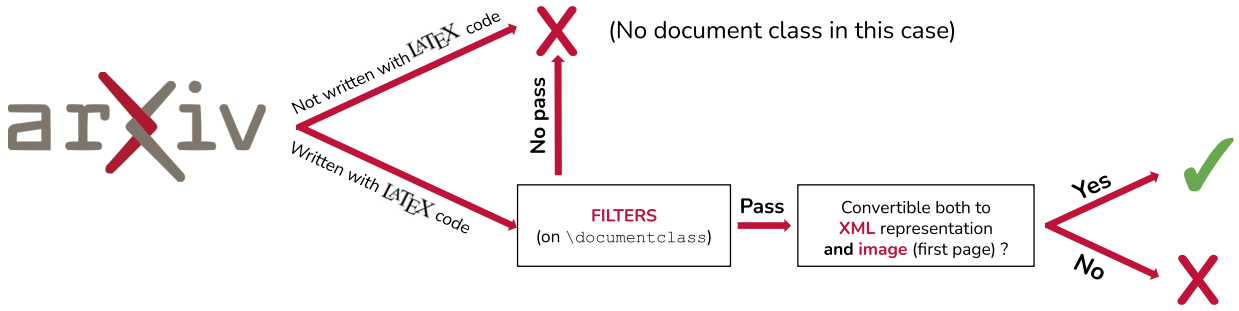

Figure 2.2: Diagram of the selection process of articles

AMS, also see Figure 2.1). It is therefore likely that some classes are in fact very similar, but a similarity in name only is not enough to justify merging these classes. We could also miss similar classes whose names are too different to predict that they are similar. To properly deal with similar document classes with different names, we compute a similarity metric between the source codes of the `.cls` document classes. We chose term frequency-inverse document frequency (TF-IDF) [20], which evaluates, for a given document class, the importance of each term in the source code of that class. We therefore compute a vector whose size is the number of distinct terms and associates to each term an importance score. Once those scores are computed for all document classes, we can compute a pairwise similarity, by computing, for all pairs of document classes (i, j) with $i \neq j$ the dot product of their TF-IDF vectors. This results in a score between 0 and 1, representing how similar the pair of document classes is. We set a threshold of 0.8. By applying this rule, we have a new short-list of 33 classes, which is our final label set for our experiments.

2.1.2 Statistical Analysis on Five Features

We explained in the previous sub-section that the document class structures the PDF rendering of the scientific article, generated from the LaTeX code. We therefore assume that it is possible to discriminate between document classes, i.e. to distinguish them from one another, by means of structural and therefore geometric characteristics. It is important to verify this hypothesis, for two reasons: firstly, to avoid getting into potentially costly experiments with no guarantee of success, and secondly, to justify that this approach makes sense and is interpretable (in particular, that the decisions taken by the model which will be presented in next subsection is theoretically explainable, and is not just a *"black box"*).

The Five Hand-designed Features

This is the reason why we chose to construct a set of five simple, human-understandable, geometric features. The idea of each feature took birth through human observation of scientific papers, and through the prior knowledge that we have about how scientific papers are usually organized. We thus want to exploit: left margins of text blocks (1), top margins on each page (2), the width of text columns (3), as well as font families and font sizes used. A graphical illustration of the first three of these elements is shown in Figure 2.3. However, each of these elements is presented in multiple copies in the document. Indeed, a scientific article almost always include several blocks of text, several pages and several fonts. Therefore, it is necessary to build aggregated features composed of the different instances of the elements presented above. Here are these features.

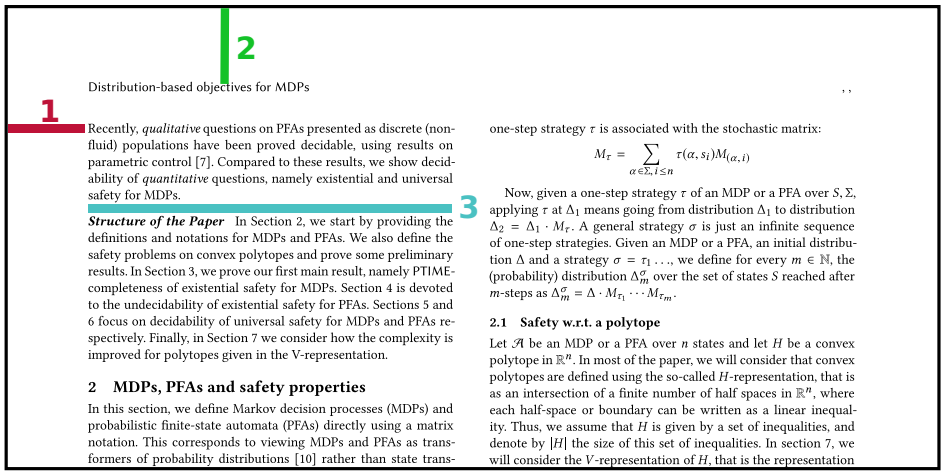

Figure 2.3: Illustration of geometrical information of interest for feature construction

Weighted average left margin (lm). The average is weighted by text-block length, in order to obtain the most representative value for the left margin. Let us denote m_i^{h} the left margin of the i-th text-block of the document and l_i its vertical height. Then, the weighted average left margin $\overline{m^{\mathrm{h}}}$ is defined as:

$$\overline{m^{\mathrm{h}}} = \frac{\sum_{i=1}^{N_b} m_i^{\mathrm{h}} \times l_i}{\sum_{i=1}^{N_b} l_i} \tag{2.1}$$

where N_b denotes the number of text-blocks in the document.

Average first top margin (tm). We are here interested in the gap between the top of the page and the very first block of content (and not between the top of the page and all blocks of content). By denoting $m_{i,j}^{\mathrm{v}}$ the distance between the top of the page and the j-th block of the i-th page of the document, the average first top margin $\overline{m^{\mathrm{v}}}$ is defined as:

$$\overline{m^{\mathrm{v}}} = \frac{\sum_{i=1}^{N_p} \min_j m_{i,j}^{\mathrm{v}}}{N_p} \tag{2.2}$$

where N_p denotes the number of pages in the document.

Weighted average column width (cw). Technically, it corresponds to the average of the width of every text block in the document, weighted by the height of the block (so as not to put importance on very short blocks, such as equations or sections headings, which do not have a typical width). We here denote w_i the width of the i-th text-block in the document, and l_i its vertical height. The weighted average column-width $\overline{w}$ is defined as:

$$\overline{w} = \frac{\sum_{i=1}^{N_b} w_i \times l_i}{\sum_{i=1}^{N_b} l_i} \tag{2.3}$$

where N_b is still the number of text-blocks in the document. Note this is very similar to the definition of **lm**.

Most common font family (ff). Choosing the most common font family requires defining a criterion. We choose to quantify the importance of a font family by the space it occupies in the document. Assume we have N_f different fonts used in the document. Let us denote S_i the set of all tokens in the document styled

in the i-th font of the document. We define the font importance f_i of the i-th font as:

$$f_i = \frac{\sum_{s \in S_i} l_s \times h_s}{\sum_{j=1}^{N_f} \sum_{s \in S_j} l_s \times h_s} \tag{2.4}$$

where l_s is the length of token s, and h_s the height of token s. The product $l_s \times h_s$ thus gives an area, and we compute the space ratio that each font family occupies. To finally get the most common font-family f^{family}, we take the font family of the font with the highest f_i:

$$f^{\text{family}} = \text{family}\left(\arg \max_{i \in \{1,\dots,N_f\}} f_i\right). \tag{2.5}$$

Despite the previous ones, this feature is a categorical one.

Most common font size (fs). The computation is based on the computation f_i defined above for each font i, and then we obtain the feature

$$f^{\text{size}} = \text{size}\left(\arg \max_{i \in \{1,\dots,N_f\}} f_i\right). \tag{2.6}$$

This is a numerical value, though in practice it mostly acts as a categorical feature as the number of different font sizes is limited for a particular document class.

Statistical Analysis

Now that the features are settled, we can have a look at their distribution once evaluated on the dataset. There are two levels of analysis that can be considered: the overall distribution of each of the features, all document class considered, and the per document class distribution of each of them. The first one will allow us to see if we can extract some global trends for the different features, and the second one will make us able to show that the distribution of the features are significantly different considering different document classes.

Regarding the overall distribution, Figure 2.4 presents the histograms of the numerical features, i.e., **lm**, **tm**, **cw**, and **fs**. Table 2.1 presents some usual statistical metrics about each of them, including: their mean value, their median value, and their standard-deviation value. There is an extra column in the table, called "Unit". In fact, the features **lm**, **tm**, and **cw** represent aggregated values of distances. These distances are computed on PDF files, where the usual distance metric is the *point*. A *point* is defined as a $\frac{1}{72}$-th of an inch.

Table 2.1: Key figures of the distribution of features **lm**, **tm**, **cw**, and **fs**

Feature	Unit	Mean	Median	SD
lm	points	159.99	166.42	38.20
tm	points	89.58	87.39	44.33
cw	points	424.25	228.51	65.93
fs	size	10.44	10.00	0.89

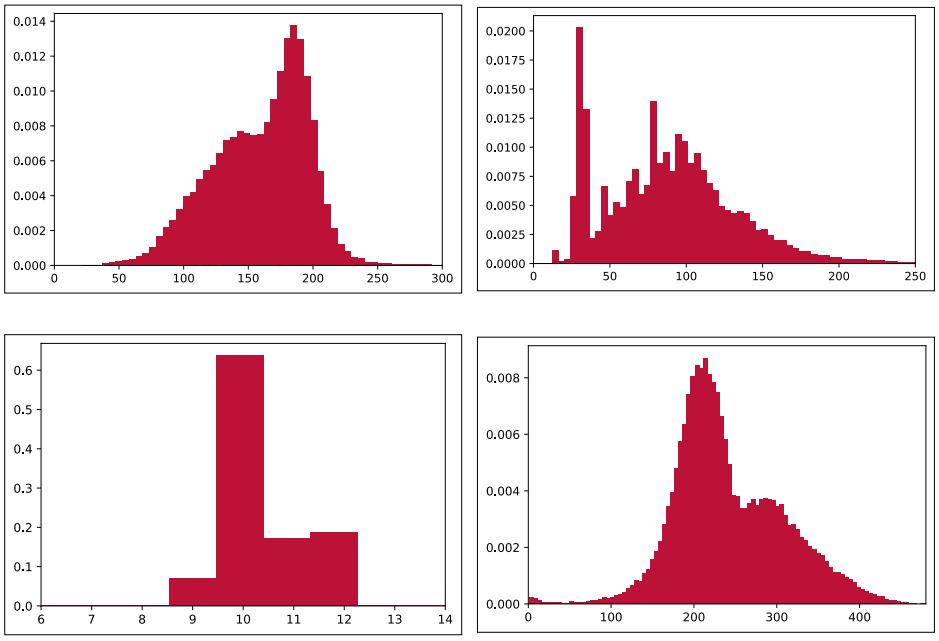

Figure 2.4: Histograms of the distribution of numerical features. From left to right, then top to bottom: **lm**, **tm**, **fs**, and **cw**

The histograms for features **lm**, **tm**, and **cw** on Figure 2.4, represent in the x-axis the number of points, and the corresponding occurrence frequency in the y-axis (between 0 and 1). For the distribution of feature **lm**, we observe what seems to be the superposition of two Gaussian-like distributions: the first with a mean close to 140 and the other one with a mean close to 190. In fact, some common document classes such as `llncs` or `article` when unmodified indeed have a high left margins, whereas other don't. This is a first, potential, source of discrimination among document classes.

Then, the distribution of feature **tm** also shows what could a Gaussian-like distribution, distributed around the value 90. However, we observe an important peek around the value 40, indicating that some articles, and probably articles associated with the same document classes, are all distributed around the same value, which is also a potential source of discrimination as well.

Regarding distribution of feature **cw**, we once again observe Gaussian-like distributions. Actually, there seem to be two of them: one with mean around 200, and the other one with mean around 300. Given the fact that the width of a page is usually around 600 (612 points for US letter and 595 for ISO 216 A4), and that space is kept for margins, these two distributions most likely correspond to papers with one column (larger value) or two columns (smaller value). This is another potential discrimination criterion.

Finally for the numerical features, distribution of **fs** shows that the font sizes are distributed among a few values, the x-axis representing a size. We see that most articles are usually using font sizes of 9 and 10 (TEX's default size), and 11 and 12, with a few exception. Once again, we can expect that different document classes uses different font sizes, and therefore that it is easier to separate document classes knowing this value.

Since we so far studied numerical features, we can compute their correlation matrix. The correlation is a score that takes value between -1 and 1, which represents how much each pair of features are linearly related (either negatively or positively). This correlation matrix is presented in Table 2.2, where each correlation score is represented in percents. We can observe that there is one important negative correlation between **lm** and **cw**. We can easily interpret this result: a document with small column width will have high left-margin (for a fixed number of columns). We also observe that there is no other significant correlation: this gives the intuition that a combination of several features is necessary to infer the document class, and justifies the idea of using a more complex modelization.

Table 2.2: Correlation matrix of features **lm**, **tm**, **cw**, and **fs**

	lm	**tm**	**cw**	**fs**
lm	-	-20.46%	-65.28%	-1.86%
tm		-	9.62%	1.17%
wc			-	3.22%

Figure 2.5 now shows the distribution of the categorical feature **ff**. We only displayed the eight most frequent font families, plus a last one called *"All others"*, containing all remaining font families (there are 109 of them in total, so this one contains 101). The three most frequent are `cmr` (default *Computer Modern* font family of TEX), `nimbusromnol` (Nimbus Roman No.9 L) and `sfrm` (a redesign of TEX 's classic Computer Modern font). We expect different document classes to use different font families, helping differentiate them.

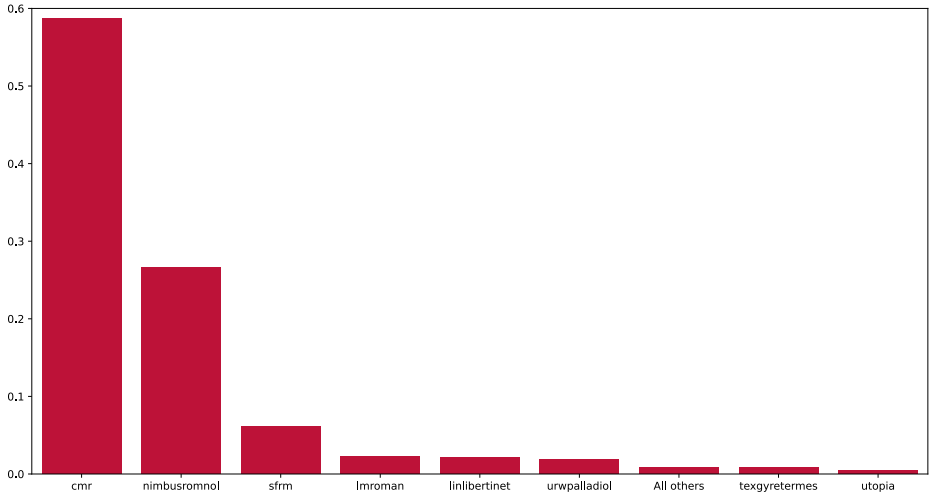

Figure 2.5: Histogram of the distribution of feature **ff**

This analysis at global scale enhances the fact that geometrical features give efficient criterion to discriminate document classes. We can verify it by comparing distribution of features for two document classes. We decided, for the purpose of illustration, to take two document classes that we know to structure articles differently. Figure 2.6 presents the distribution of features **lm**, **tm**, and **cw**, for document classes `bmvc2k` (document class of the *British Machine Vision Conference*), and `aa` (document class of the *Astronomy and Astrophysics* journal). We can see that the distributions of the two document classes are significantly different, distributed around different mean values. `bmvc2k` generates a single column, and thus smaller left-margins and larger column widths then `aa`, which generates two columns. We also observe that `bmvc2k` has small top-margins, whereas `aa` has larger ones (twice its value, in mean).

This statistical study shows the relevance of the use of a geometrical approach to infer the document class of a scientific article. In the next subsection, we will now present the modelization we use to achieve this classification task.

2.1.3 Deep-Learning Approach: Convolutional Neural Networks

In order to achieve the task of inferring the document class by using geometrical features, we chose to use a computer-vision based method, called *Convolutional Neural Networks* (CNN) [13, 8]. CNNs are a deep-learning method that is based on the convolutional operation. This is why this method is particularly suited for our task, since they perform well in image processing and geometric pattern learning through the use of convolution, while limiting the number of parameters to be trained (in comparison with fully

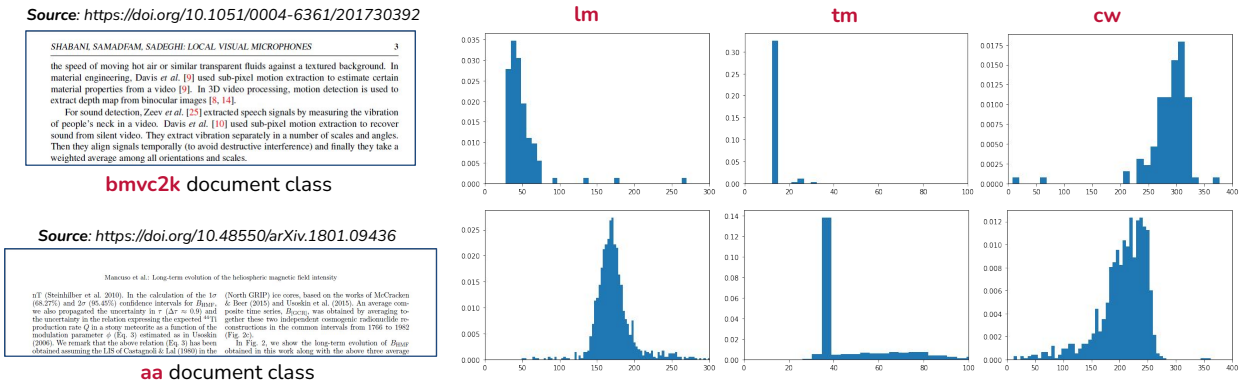

Figure 2.6: Distribution of two document classes on the features **lm**, **tm**, and **cw**.

connected, feed-forward, neural networks), by successively applying several convolutional operations over an image presented as input. Since we are dealing with PDF representation of articles, we can easily extract images of pages from this representation, and thus use these as input for our CNN.

Choice of the Input

However, a PDF of an article is (usually) composed of several pages, and thus would lead to several images. Since we want to keep focused on a simple CNN architecture, in comparison with other architectures that enable the processing of a *sequence* of images, we needed to select one image to present to the model. This image had to be representative enough of the discriminating characteristics that an article contain regarding the document class.

We chose to use the image of the first page of each article. This is explained by the fact that this particular page contains elements that are structured differently depending on the document class that is used. Examples of these usual elements are the following: the title of the article, its authors and their affiliations, or the abstract of the paper. We are not interested in the concrete content of these elements, but how they are displayed. It can include their position, their size, if the text is in capital letters or not, if the abstract is written with single or double column, and so one. You can observe here that these observations are indirectly related to the hand-designed features we presented before. Another interesting characteristic of the first image of the paper is that, for some document classes, it contains other elements that can help the model directly identify which document class is used. For instance, some of the documents classes from the AAS contain a line on the top-left corner of the first image, indicating that this article was written using a particular document class. This is also something that can be observed for document classes of the ACM, which most of the time contains a line describing how should be formatted the reference of ACM's paper, and a quite common paragraph describing the rights associated to the paper. Other document classes contain in their first page a table of contents, which make them easily separable from the ones that don't.

Thus, we will present to the model a bitmap rendering of the image. We generate it by using a standard Python library called `pdf2image`[4]. However, the size of the obtained images will differ from one to another (as we described it in Section 2.1.2, size of pages varies depending of the paper that is used), and our CNN need a fixed input size. We thus chose to rescale each obtained image, to make it fit into a $256\dot{2}56$ square of pixels. This particular size was chosen by experimentation, especially to have an interesting trade-off between performances and size of the model (this trade-off will be discussed later on). We also chose to only keep a grayscale version of the image, because we consider the color useless for the task we consider. This also helps significantly reducing the number of parameters and operations to be processed by the model. An example of such a generated image is presented in Figure 2.7, from a paper taken randomly in the dataset. You will note that the text becomes illegible with this resolution. However, as we mentioned it above, we are not interested

[4] `https://pdf2image.readthedocs.io/`

Figure 2.7: Example of an image used as input for the CNN, for a paper taken randomly in the dataset

in the content, but in the information about the structure, geometry, and style of the document, which are independent from this content.

Choice of the Modelization

As already mentioned several times, we want to implement a CNN. But this is just a generic method, which we need to specify. As the statistical study shows, this classification task seems pretty simple if we consider a modelization based on geometrical characteristics. Moreover, a simple model (a random forest [5]) was already trained on the five characteristics presented in Section 2.1.2. The results of this experiment can be found in [6], and shows that even with only five features, we end up having encouraging results. This is why we made the choice to build a really light CNN model. But before presenting it, one needs to understand how is structured a CNN.

As opposed to fully connected layers of other forms of neural networks, CNN uses *convolutional layers*, based on the *convolution* operation. Denoted $*$, it is defined, in its discrete form in two dimensions, which is the one we are interested in, as:

$$(f * g)_{m,n} = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} f_{i,j} \times g_{m-i,n-j} \tag{2.7}$$

where f is the kernel or filter, and g is the 2D input. For numerical applications, f and g are finite. The parameters of the model consists in the values of the f matrices. We can have several filters per hidden layer, as well as several hidden layers. In addition to these convolution operations, we also apply a set of other operations, such as: activation functions, to know which convolutional neurons are activated by the parameters; pooling operations, to progressively reduce the size of the images through the hidden layers and to get a small set of embeddings at the end of the model; and dropout operations, which aim at randomly dropping some of the neurons to avoid overfitting.

We then concatenate several of these layers, and finally add a flattening operation and one (or several) last fully-connected layer, to transform our set of small matrices of embeddings into a set of *logits*, corresponding to the number of classes in our problem. It is also possible to alternatively add hidden convolutional layers and fully-connected layers, but since we want to keep things simple, we do not consider this.

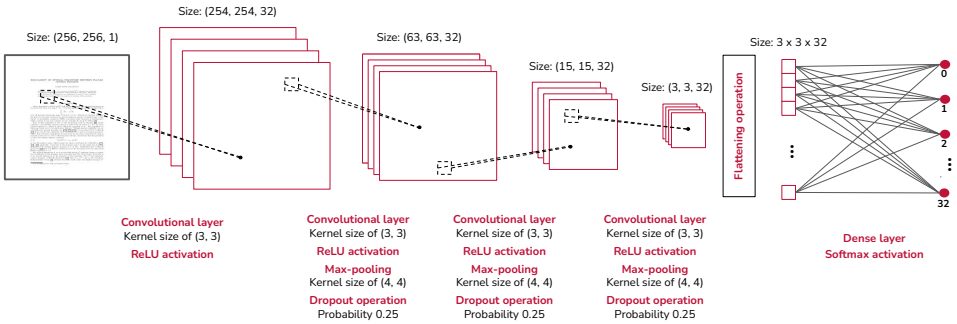

Figure 2.8: Diagram of our CNN architecture

We chose to design, in a fairly typical way for simple CNNs, a simple model with four hidden layers, with increasing numbers of filters and decreasing size of the embeddings throughout the layers. Its architecture is presented in Figure 2.8.

Our assumption is that our simple architecture is sufficient to discriminate document classes since, as already mentioned above, first images of papers contain an important number of possible patterns indicative of document classes. To verify this assumption, we also train some more complex architectures from the state of the art. Thus, we also consider the following ones: a 50-layer *ResNet* from 2016 [10], a Mobile version of *NASNet* from 2018 [25], and a B0 version of *EfficientNetV2* from 2021 [21]. These are chosen because they are standard CNN architectures and have a reasonably limited number of parameters. We pay special attention to *EfficentNetV2-B0*, first because it is the most recent of the considered architectures, and second because it was designed to propose better parameter efficiency than previous models. We set aside even more modern architectures with a large number of parameters, such as the *ConvNeXt* architecture of 2022 [15], whose basic model nearly reaches 90 million parameters.

Results and Analysis

To be able to compare the complexity of these different models, we present in Table 2.3 the total number of parameters of each architecture, as well as the number of floating-point operations (*FLOPs*) required for one pass of the image over the entire model at inference (as defined in [22]), for the same (fixed) size of input images, which is a machine-independent way to measure the time for inference. It is important to consider these two criteria in order to check that our model is respectively simple, light (thus, that its number of parameters is limited and that it takes little space in memory), as well as fast, in training phase, but especially in inference phase: it is an essential criterion so that the model can be used later on for mathematical environments extraction (we must keep here in mind that the goal of this inference is to use the document class as input for another model).

Table 2.3: Complexity comparison between different architectures of CNN, for input size 256×256

Architecture	**Total # of parameters**	**FLOPs**
Our simple architecture	0.04M	1.36B
ResNet50V2	23.63M	9.13B
NASNetMobile	4.30M	1.50B
EfficientNetV2B0	4.09M	0.80B

Now regarding the evaluation of the concrete performance of the model, we decided to use the F_1-score at macroscopic level. First, the F_1-score of a given class i is defined as follows:

$$\text{precision}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i} \tag{2.8}$$

, with the recall score of a given class i defined as follows:

$$\text{recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \tag{2.9}$$

and the F_1-score, which is the harmonic mean on of both precision and recall, defined as follows:

$$\text{F}_1\text{-score}_i = 2\frac{\text{precision}_i \times \text{recall}_i}{\text{precision}_i + \text{recall}_i} \tag{2.10}$$

Here, TP defines true positive, FN, false negative, and FP, false positive. Using this metric instead of the simple global accuracy helps identifying the missclassification errors by being less sensitive to data imbalance. In this same direction, we use the macroscopic level that gives same weight to the F_1-score of each class, even though these classes have a difference natural frequency.

In order to generate these results, we first trained each model with the same data, by keeping 80% of the initial dataset for training purpose, and the rest for training. We then used *oversampling* techniques [18], so that the model gives the same importance to each document class, no matter their natural frequency. In order to do so, we just randomly duplicate samples for all document classes except the most frequent one, so that they all reach a same number of samples than this most frequent one. Doing this also has the (nice) side effect to show more data to the model, helping it converging towards its optimal loss. The obtained results for each of the architectures presented above are reported in Table 2.4. An extended version of these results on the performance metrics we use can be found in [6]. The training was made possible thanks to the use of the supercomputer *Jean Zay* from IDRIS[5], using 32-GB Nvidia Tesla V100 GPUs.

Table 2.4: Mean precision, recall and F_1-Score for different architectures of CNN

Architecture	**Precision**	**Recall**	**F_1-Score**
Our simple architecture	92.67%	92.70%	92.31%
ResNet50V2	93.59%	92.34%	92.28%
NASNetMobile	93.23%	91.17%	91.31%
EfficientNetV2B0	**93.63%**	**93.52%**	**93.43%**

We observe that the best performance is obtained for the *EfficientNetV2B0* model, with 93.63%. However, our architecture is just above 1% less performing than this one. This is a really good result, since it contains more than 100 times less parameters, making it way lighter. Regarding the number of floating operations at inference times, our architecture is around 1.7 times longer than *EfficientNetV2B0* model. However, in practice, the difference is very negligible, since the inference time for our architecture is only about a couple of milliseconds. Moreover, *EfficientNetV2B0* was specifically designed to reduce this number of operations, while our model does not specifically focuses on this task.

We consider these results to be good enough to be used as input data for other models. However, we must keep in mind that it will still add uncertainty to the performance metrics of the models that will take these inferred document classes as input, since these ones will be somehow uncertain as their value results from a prediction which is not certain. Still, by having a look at the detailed results (presented in [6]), you will see that only a couple of document classes is pulling global results down. These classes are in fact ones that we could call *heterogeneous*, meaning that they are really customizable, and that their structure highly varies from one article to another. If we now take a step back and consider this task as part of the extraction of mathematical results, it is most likely that the papers of interest will be papers published in journals or proceedings of conferences, and thus will not be associated with these kind of *heterogeneous* document classes. The impact of these specific document classes might be less important when considering this specific task. This is also why we retrained a CNN model with the same architecture, by removing these heterogeneous classes from training. We obtain better result, with macroscopic F_1-score above

[5] `http://www.idris.fr/eng/jean-zay/`

96%. Here again, the detailed results can be found in [6]. Finally, there exists a bias here, which is due to the data we considered to train our model. We only considered document classes of year 2018, and since document classes evolves, we might need to train this model on a wider range of time. We are also restrained by the fact that we need to extract the ground-truths on ArXiV, which might not be representative of all papers published in the scientific fields, even though its use F now widely spread in the scientific community.

We presented the overall task of automatic document class inference in a more detailed manner in a scientific article [6]. This one was submitted as a long research paper in the *DocEng 2023* conference, a conference about document engineering in general. It was accepted, and will thus be presented at the conference that will be held in August 2023 in Limerick, Ireland. The models trained in this work, as well as information about the dataset we used, are available online here[6], and are part of the deliverables of this internship.

2.2 Multimodal and Sequential Approach for Extraction of Theorems and Proofs in the Scientific Literature

The next step of this work consists in setting a strong baseline regarding extraction of theorems and proofs in the scientific literature, which will be used as a basis to study if the document class can bring any improvement to it. Extraction of theorems and proofs in the scientific literature is something that has been carried out by the Valda team since 2021. It has conducted to some publications, in particular [17], and most of the work is still in progress.

In this section, I will first present what we are exactly trying to do when extracting theorems and proofs, as well as a short description of which data we are going to use. Then, I am going to present the most recent modelization the team, and especially one PhD student of the team who is Shrey Mishra, up to date, which we call *multimodal*. Then, I will present my contribution on this work, which consisted in improving the *multimodal approach* by adding a *sequential* layer over it. I will then comment on the obtained results, first to show the relevance of my work, and then take a step back to analyze the results in the scope of the task we try to achieve.

2.2.1 Presentation of the Task and of the Data

As described above, we hereby want to extract the mathematical environments from the scientific literature, and especially theorems and proofs. It is important to specify here that we are talking about the LaTeX definition of theorems environments and proofs environments, i.e., the one defined with commands like `\begin{theorem}` and `\begin{proof}`, or other different keywords in the brackets if it was redefined. This extraction task is a problem that has already been studied in the literature. In particular, [7] achieve the task of classifying paragraphs from scientific articles into thirteen different categories. For that, they use a bi-directional LSTM encoder-decoder model [9], which is a particular implementation of recurrent neural networks. While still giving good performances, their approach is based on the use of the HTML rendering of the papers, through LaTeXML. However, this is only possible if one has the LaTeX source of the article at its disposal, which is a big limitation, since we want our system to only use the PDF representation as input, so that any article can be presented (we do use LaTeX source code at training time, as it will be presented later on, but not at inference time).

Moreover, [7] only focuses on the textual content of each of the paragraph. But we know, and the work on document class inference presented in Section 2.1 shows it, that there exists other important information that can be useful to detect specific layouts or structures in a scientific article. This is why we make the hypothesis that considering other (and multiple) *views* of each paragraph, which we call *modalities*, would make our classification task easier. Each *modality* would bring its own representation of the paragraph, which will embeds a limited amount of the total information that a paragraph really embeds. Using multiple *modalities* should help reconstruct a more important quantity of information, and thus better detect mathematical

[6] `https://github.com/AntoineGauquier/inferring_document_class_of_scientific_article/`

environments. Information extraction from PDF documents with multiple *modalities* is something that has already studied. The most famous works about it are certainly the *LayoutLM* systems [23, 24, 11]. They give good performances by using two modalities, which are textual information as well as layout information (2D positional embedding), but they are not specifically trained for our context of scientific literature, and only use two different modalities.

This justifies the consideration of a *multimodal approach*, built from modalities we think relevant given the task we are achieving, each of them specifically trained in the context of scientific literature. In order to do so, we are going to use the same kind of data that we used in Section 2.1.1: the PDF representation of each article from which we will extract the information used to train the modalities, and the LaTeX source code to know which paragraphs are actually theorems, proofs, or none of them. The ArXiV platform is also used here, for the same reasons as before.

2.2.2 Multimodal Approach

The multimodal approach that has been developed by the Valda team uses three different modalities. In fact, when having a look at mathematical environments, we can identify several elements that appears to be frequent. The first element is the *textual content*: most of proofs and theorems environments will begin with words such as *"Proof"*, *"Theorem"*, *"Definition"*, *"Lemma"*, and so on. These words will also sometimes be followed by a number that helps referring to this particular mathematical content. The textual content might also catch a certain vocabulary that is specifically used in these environments. The words mentioned earlier most of the times come with a certain font, i.e., a certain formatting. For instance, the word *"Proof"* in proofs environments are usually written in italic, whereas *"Theorem"* word from theorems environments come in bold. This gives us a second modality, which we could qualify as *font modality*. Finally, we think that we could benefit from catching some of these already presented elements, as well as positioning ones, such as indents for instance, through the use of a visual representation of paragraphs. This representation could also catch some graphical elements, such as the *QED* symbol (□ or ■), that textual content might not, especially when it was "badly" written in LaTeX (as a drawing for instance). This last representation builds a last third modality, called *vision modality*. An example of several of these elements is presented in Figure 2.9, which represents an example of a theorem environment followed by a proof environment taken from a paper randomly chosen in our database.

Theorem 5. *Let B be a positive operator on the Hilbert space $\mathcal{H}$. The mapping*

$$\Psi_B(P) := J_B P J_B^* \tag{2}$$

establishes an order preserving bijection between $\mathbf{P}(\mathcal{H}_B)$ of $\mathrm{Q}(B)$.

Proof. According to Theorem 3, Ψ_B is a surjection and since J_B^* has dense range and J_B is one-to-one, Ψ_B is also an injection. It is immediate that Ψ_B is order preserving. That Ψ_B^{-1} is order preserving too follows again from the fact that the range of J_B^* is dense. □

Figure 2.9: Example of (LaTeX) theorem and proof environments, with characteristics elements

Gated Multimodal Units

As mentioned above, this approach aims at using multiple *modalities* to achieve the classification task. This implies two subtasks: training models, associated with the modalities, that can process the associated data, and finally merge these modalities to have one global model that merges them. A representation of the global model is presented in Figure 2.10. We are going to present it from input to output (from left to right in the figure), and go over these two subtasks.

On the extreme left of the diagram, we can see observe the inputs of the model, already presented before. Then, these inputs are cut into blocks. Each of these blocks in fact correspond to one paragraph of the article, which we want to classify as theorem, proof or none. In order to do this cutting, we use two different

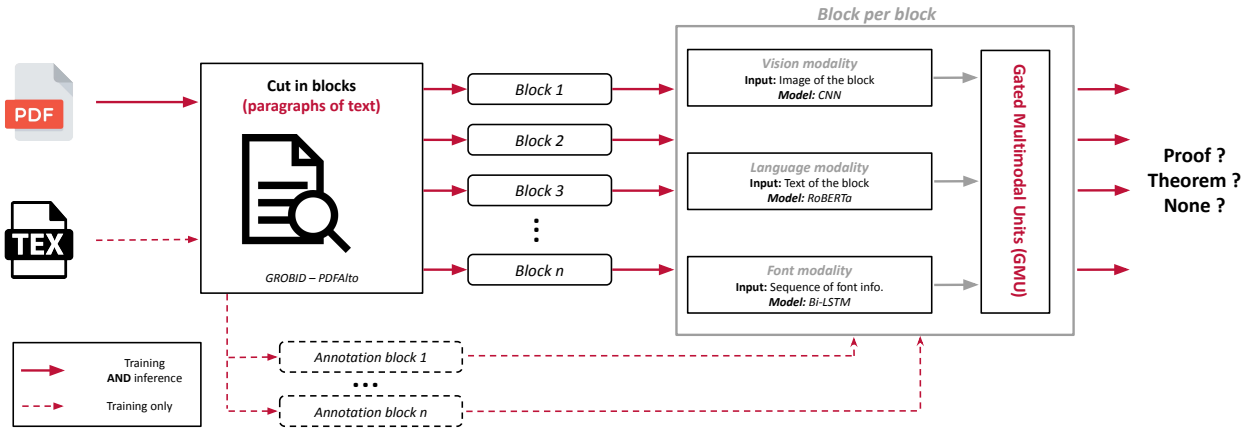

Figure 2.10: Diagram presenting the the multimodal approach

software. The first one, *GROBID*[7], is used to extract paragraphs from the PDF representation. The second one, *PDFAlto*[8], is only used for training, because it aims at extracting where are theorems and proofs in the article, from the LaTeX source code, if there are so. Before applying the software, we automatically add some annotations to the LaTeX source code to help it find where are the mathematical environments. Once both inputs are processed, we then merge the blocks of text from *GROBID* and the positions of mathematical environments from *PDFAlto* through their respective page number and coordinates.

The rest of the process is then done block per block, so paragraph per paragraph. The next step, going a bit further on the right, is the training of the different modalities. As a reminder, there are three of them. The first presented in the diagram is the *vision* modality. This modality takes as input an image of the block. This is done by using the same Python library as in Section 2.1.3, this time by cutting on the image of the page, the block through its coordinates. Then, from the image of the block, a CNN [13, 8], and especially an averaged version of *EfficientNetV2* [21], is trained on the task of classifying this block as theorem, proof or none of them, i.e., the exact task that we want our multimodal approach to achieve, by adding a *classification head* ad the end of the network. Then we have the *language* modality, that takes as input the textual content of the current paragaph. *GROBID* directly provides this content, since we indeed used it to cut the article in paragraphs. This text is tokenized (in a vocabulary of size 50 000) and presented to a *RoBERTa* like model [14], which is pre-trained on a specific corpus of scientific articles (around 200 000 of them). Then, a *classification head* is added to this pre-trained language model, to transform the embedding generated by the BERT-like (Bidirectional Encoder Representations from Transformers) model into the task of extracting mathematical environments. The last modality, called *font*, takes as input a sequence of styling information (font-size, font-family, etc.). A vocabulary of 4031 tokens is built from the most observed styling information in the dataset. Then, a sequence of these token is built accordingly to the current paragraph, and then presented to a 128-cell bi-directional LSTM [9], which is trained to capture the sequencing of these styling information. A *classification head* is here again added.

Each of these modalities are trained independently from one to another (since they all have their own input data). Once they are trained, we want to merge them through the use of a *Gated Multimodal Units* (GMU) [1]. This modelization was designed to make information fusion possible, especially when having at disposal different modalities representing the same information. The use case described in this article is movie genre classification, where the authors try to classify, from both textual description of the film and image of the film cover, the genre of the associated film. In order to do so, they train two models that can handle these information, on the task of genre classification; so as we did with our modalities on the classification of types

[7] `https://github.com/kermitt2/grobid`
[8] `https://github.com/kermitt2/pdfalto`

of paragraphs. However, this modelization merges modalities through the use of several gates in the network, which are conceptually similar to LSTMs. An illustration of a GMU gate, taken from [1], is presented in Figure 2.11. Thus, the GMU model requires a numerical representation (represented in the illustration by the $\{x_1, \ldots, x_k\}$) of the modalities to be able to merge them. This is why the team decided to present *embeddings* as input for the multimodal gates. In machine learning, we call *embedding* the information carried by a specific representation of some data, in a latent space. Concretely, it will usually correspond to the numerical vector that is obtained before applying *classification heads* from each of our modalities. To generate them, we just need, at inference time, to extract the numerical representation before the *classification head*. These *embeddings* will thus represent different views of the same paragraph through the prism of our classification task.

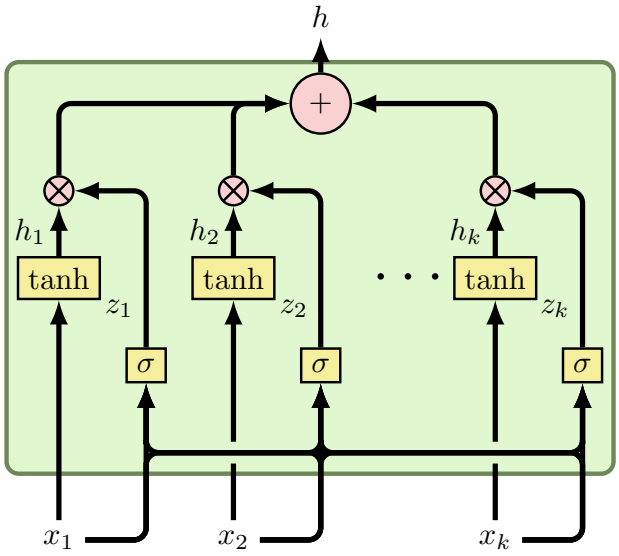

Figure 2.11: Graphical representation of a GMU gate

Finally, the GMU modelization is then applied to the concatenation of the *embeddings* of the different modalities, producing, once trained, a final *embedding*, which we will call *merged embedding*, because it is carrying all the information of the different views of the same paragraph, still through the prism of the identification of mathematical environments. From this final *merged embedding*, we can achieve the task of classifying our paragraphs of text in proof, theorem, or none of them, by applying a final *classification head*, which is trained at the same time as GMU model is.

Results and Analysis

Each of the modalities, as well as the GMU model, were trained using the supercomputer *Jean Zay* from IDRIS (the same as in Section 2.1.3). Multiple kind of GPU resources were used, including 32-GB Nvidia Tesla V100, 40-GB Nvidia Tesla A100 ans 80-GB Nvidia Tesla A100. The evaluation of the performances is carried out using the macroscopic F_1-score already presented in Section 2.1.3, as well as the microscopic accuracy, which is basically the accuracy computed considering the natural frequency distribution of the classes (proof, theorem or none of them). Table 2.5 presents the results for the three modalities (*vision*, *language* and *font*) and for the multimodal approach.

Table 2.5: Microscopic accuracy and macroscopic F_1-score for each modality and for the multimodal approach

Modality	**Modelization**	**Microscopic Accuracy**	**Macroscopic $\mathbf{F_1}$-Score**
Font	Bi-LSTM	65.00%	45.00%
Vision	Averaged *EfficientNetV2*	69.43%	60.33%
Language	Pretrained *RoBERTa*	76.45%	72.33%
Multimodal	**GMU**	**76.86%**	**73.87%**

The first thing we notice is that the multimodal approach is giving the best results (above 76% in accuracy and 73% in F_1-score), confirming the hypothesis and intuition we described earlier. However, we also notice that the performance of *language* modality is really close from multimodal approach, hinting that the language modality is the one that seems to bring the most information. The work presented in the next subsection will give explanations to this. Then *vision* modality comes, with around 70% of accuracy and 60% of F_1-score, and finally *font* modality has the "worst" results, with 65% of accuracy and 45% of F_1-score.

If we take a step back, these are actually promising results, given the complexity of the task. Even though some blocks of text will have discriminating elements that will make them easily identifiable as theorems or proofs environments, other will not. This is for instance particularly true for long proofs, where we will find some of these elements at the beginning and at the end of the proof, but all paragraphs in the middle will likely contain similar information than a classic paragraph of text which is neither a theorem, nor a proof environment. This is what brought us to consider and modelize the *sequencing* of the paragraphs of text we are trying to classify.

2.2.3 Sequential Approach: Conditional Random Fields

Even though we splitted the PDF in blocks, we must keep in mind that the article was written to be read in a specific order. Thus, the consideration of the *sequencing* of the blocks is something natural. Moreover, if we further think about the example of long proofs taking multiple successive blocks, the consideration of this *sequencing* could help classify this set of blocks as proof. In fact, if we take any block of this set, as long as it is not the first or last block of the set, then knowing take there are one or several blocks identified as proofs before and after the considered block, then it is really likely that that block is itself a proof.

This led me to make the assumption that using a *sequential* approach on top of what has already been done so far should improve the results, and especially the good classification of classes "of interest" (the proofs and theorems). Given the observation we made about the multiple successive blocks of a same class, a quite common approach to this problem is the use of probabilistic modelizations that handle sequences of random variables. There are several usual modelizations of this kind, such as *Hidden Markov Models* (HMM) [4] for instance. We decided to use *Conditional Random Fields* (CRF) [12], which are less known, but have good theoretical properties. They are commonly used for labeling tasks over sequential data, which is exactly what we are looking for, the labeling being in our case the classification of blocks as proof, theorem or non of them. CRF enable the modelization of *graphical models*, i.e., models describing the dependencies that can exist between the random variables. Some of its interest is related to the fact that CRF can model any kind of dependency between random variables. This is not the case for [4] for instance. This is the reason why CRF are sometimes called *probabilistic neural networks*, because they can describe any kind of connections between random variables.

A CRF handles to sequences of mathematical objects: $\mathbf{X} = (X_1, \dots, X_n)$, a sequence of observations about $\mathbf{Y} = (Y_1, \dots, Y_n)$, the set of random variables we are trying to estimate through the use of CRF. We say that $(\mathbf{X}, \mathbf{Y})$ is a conditional random field if, by letting $G = (V, E)$ be a graph such that $\mathbf{Y} = \{Y_v\}_{v \in V}$ (and thus $|V| = n$, concretely $\mathbf{Y}$ constitutes the vertices of the graph), the probability of Y_v only depends on its *neighbors* in the graph G. Formally, we must have $P\left(Y_v | \mathbf{X}, \{Y_w : v \neq w\}\right) = P\left(Y_v | \mathbf{X}, \{Y_w : v \sim w\}\right)$. The freedom in the modelization of the dependency mentioned above is embodied by the $\sim$ symbol between v and w in the right hand side of the equality. A graphical representation describing this neighborhood dependency

is represented in the general case in Figure 2.12. The graphical representation of an HMM is also represented in this figure. This figures also shows some of the differences between an HMM and a CRF : the freedom of variable dependency already described above, but also the fact that CRF are *unoriented graphs*, whereas HMM are *oriented*. In practice, this difference is important, especially for our case, because, in theory, we would want our modelization to catch dependencies in both directions.

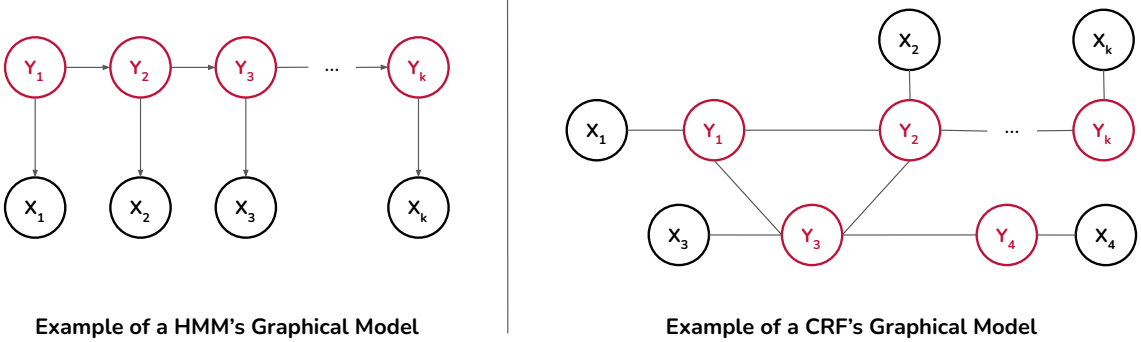

Figure 2.12: Examples of graphical models for HMM and CRF models

Modelization

If we now go back to our problem, we need to define what are $\mathbf{X}$ and $\mathbf{Y}$ to modelize our implementation of a CRF. $\mathbf{Y}$ being the sequence of random variables we are trying to estimate, it will correspond to the type of the blocks we want to know. Therefore, here $n = |V|$ will be equal to the number of blocks in our sequence. The set $\mathbf{X}$ of observations can be seen as the observations that we have about the set $\mathbf{Y}$. This set of observation is known, and will constitute the input of our CRF, which will be described just after. Finally, there is the question of the $\sim$ symbol, i.e., how we want our random variables to depend on each other. Ideally, we would have wanted each of the variables to rely on some of the previous random variables, and to some of the upcoming random variables: in other words, we wanted the type of the current block to rely on the types of the few blocks preceding and following it.

However, we faced a technical implementation issue there. After looking for some general implementation of conditional random fields on the web, it turned out that they were no implementation of CRFs in Python that were not dependent of a certain dependency modelization. There are some in C++, but they are not converted in Python yet, despite a lot of other probabilistic modelizations. Since it was not reasonable to consider implementing this CRF general modelization myself, mostly due to a lack of time, we made the choice to use one of the already implemented CRFs with a fixed dependency. The one we chose is `python-crfsuite`[9], a Python implementation of CRF with Markov property assumption for dependency. The Markov property assumes that the conditional probability of a random variable at state $k+1$ only depends on the last considered random variable k, and not all the already considered random variables $\{1, \dots, k-1\}$ (the future only depends on the present, and not on the past). Considering our problem, it consists in only looking at the type of the previous block when trying to estimate the type of the current block. In that case, we talk about "linear-chained CRF", because the random variables are linearly chained.

We still need to define what will be our sequence of observations $\mathbf{X}$. Since we wanted to show that our hypothesis about the relevance of considering *sequencing* is true, we decided to train four different CRF models, on four different sequences of observations. The corresponding features are presented in Table 2.6.

[9] `https://python-crfsuite.readthedocs.io/`

We first train three different models which will take as input the output of the three modalities presented in Section 2.2.2. In order to have a sufficient amount of variables to correctly describe the data of each modality, we will use the *embeddings* we extracted to train the GMU approach. We thus respectively have 1 280 features for vision modality, 768 features for language modality and 128 features for font modality. A fourth CRF model is also trained, taking as observations the output of the multimodal approach, i.e., the *merge embedding* that we obtain after going through the GMU model, representing 768 features. Finally, we had, for all four models, some extra features that represent the sequencing itself. These features are the normalized number of the page (number of current page of the block divided by the total number of pages in the PDF), the normalized vertical and horizontal distance between the current block and the previous block (normalized in the sense that the computed distances are divided by the maximum width and height of the PDF) and finally a Boolean feature describing whether the current block is on the same page as the previous block. This last feature is necessary since the distances mentioned earlier are computed with the coordinates of each block, but it does make sense only if these two blocks are in the same page. You will notice that these extra features are adapted to the Markov assumption.

Table 2.6: Different sequences of observations $\mathbf{X}$ for the different CRF models we want to train

Information	**Nature**	**Number of features**
Vision modality	CNN *embeddings*	1 280 features
Language modality	RoBERTa-like *embeddings*	768 features
Font modality	Bi-LSTM *embeddings*	128 features
Multimodal model	GMU *embeddings*	768 features

Results and Analysis

Each of these models are trained independently, with their own features, but on the same data. The training is made directly through CPU (since it is not deep learning, it does not require GPU), on *Violette*, a computer shared in the team mostly used to store data and run jobs that require CPU. Of course, we must keep in mind that this is an "extra" cost, meaning that the features we are using to train these CRF models are extracted from models that required GPU to be trained (the different modalities and the multimodal approach).

In order to show the relevance of this approach, we are going to compare the results we obtained for the task of classifying the blocks, before applying the sequential approach, with the ones with obtain when training CRF model and therefore taking into account the sequencing. We are using the same performance metrics as in last subsection, which are macroscopic F_1-score and microscopic accuracy. The obtained results are presented in Table 2.7.

Table 2.7: Microscopic accuracy and macroscopic F_1-score for each CRF model, compared to baselines

Modality	***Accuracy (Baseline)***	**Accuracy**	***F_1-Score (Baseline)***	F_1-Score
Font modality	**65.00%**	52.20%	45.00%	**50.49%**
Vision modality	69.43%	**74.13%**	60.33%	**69.82%**
Language modality	76.45%	**82.70%**	72.33%	**80.52%**
Multimodal model	76.86%	**84.38%**	73.87%	**83.01%**

The results that are obtained are really positive. First of all, we observe that considering the *sequencing* of the blocks definitely helps solving our task, since it improves both accuracy and F_1-score in almost each case, exception made for the font modality, which is the modality that have the most difficulties to solve the task anyway. This improvement is even more interesting since it shows that considering *sequencing* improves robustness of the results. By robustness, we mean that the model specifically improves in detecting the classes that are of interest. This is one of the assumptions we made earlier, given the intuition we had about examples as long proofs. We can observe it by noticing that, in global, we have a bigger improvement on the F_1-score than on the accuracy. But we know that accuracy is taken at microscopic level, so by following natural distribution of the data, whereas F_1-score gives equal importance to all classes. Since there are more

blocks that are neither proofs, nor theorems naturally, then this discrepancy of improvements show that the model is better in well classifying proofs and theorems.

If we now take a step back, we successfully showed that considering *sequencing* is significantly improving the performance on this task. This is something really promising keeping in mind that we only used a probabilistic approach, without deep learning. Moreover, we made a really restrictive assumption with the Markov property. We can reasonably affirm that improving this modelling should improve results even more. By doing so, we also showed that GMU approach really catches more information that other modalities does. In fact, we observed earlier that multimodal approach was not improving that much the results on this task compared to language modality for instance. But here the results with CRF modelization show that there is actually a big difference between these two modelizations, and that mutlimodal approach is catching more information since it is improving when considering what happened for the last block.

The overall work presented in this whole section was described in a long research paper [16], which was sent for publication in the proceedings of a conference and is still being reviewed by the peers. A demonstration of the overall system, including GMU modelization and CRF approach, is being set up at the moment, and will be sent for presentation in a conference. A repository (still under construction) that will be part of the material of both paper and demonstration can be found here[10]. Since my work is part of this repository, and that I also took part in its development, this work is also one of the deliverables of this internship.

We will now describe the study of the impact of the document class on the task of identifying mathematical environments.

2.3 Study of the Impact of the Document Class on the Extraction of Mathematical Environments

In this section, we describe the study of the impact of the document class on the extraction of mathematical environments, which was the initial aim of this internship. This study is now possible thanks to the work described in the two previous sections. Indeed, we are now able to automatically infer the document class given the PDF rendering of a scientific article, with a great confidence, in an efficient manner. Moreover, we built a strong baseline on the task of classifying blocks of text according to their mathematical nature.

We will now concretely present what we want to achieve in this study, as well as the data and features we are going to use to do it. Then, we will present the different architectures we consider to study this impact, in the order they were considered. Finally, we will make a summary of the few improvements that still need to be considered regarding the overall task of extracting mathematical environments from the scientific literature.

2.3.1 Presentation of the Task and of the Data

Since we want to exploit what we have already done to conduct the study, the task we hereby want to solve must start from the two main modelizations we made so far. The first one is the CNN architecture we trained to automatically infer the document class, and the second one is the GMU architecture we trained to classify the blocks of text. What we want to do is to try to connect in some way these modelizations, or at least, their outputs, to see if we can benefit from including information about the document class in a modelization that is identifying mathematical environments in the scientific literature.

The fact that we used, for both previous sections, data with the same structure (PDF representations, plus LaTeX source code in training phase), extracted from the same platform, is a big advantage here, because we can practically directly begin the training phase once the modelization is done, by using the dataset of last section for instance.

[10] `https://github.com/mv96/mm_extraction`

The modelizations we will present in the next subsection will thus have to solve a classification problem, which is the one of identifying the mathematical environments on a scientific article. We will still work block per block, so that the multimodal approach can be used as input (and compared to). We don't use the CRF approach as input, because we wouldn't be able to extract an input which is an *embedding*. Besides, we ideally want to study the impact of the document class independently from the consideration of the *sequencing*. This will be further discussed when presenting the different architectures.

As we just mentioned, we want to use *embeddings* from the two architectures we try to "merge". On the one hand, for multimodal approach, we will use the *merged embedding* which is the output of the GMU model, without the *classification head*. For the CNN which automatically infers the document class, we can also extract an *embedding*, again by removing the *classification head* which can be observed on the extreme right of Figure 2.8. In this case, it corresponds to a 288-dimensional *embedding* (just after the flattening operation, we flatten the set of 32, 3×3-sized filters into a vector of 288 features). However, since we are interested in the document class, we can also try to use the prediction which is made by the CNN architecture plus the *classification head*. This can be done using the one-hot-encoding representation of the document class which is predicted. A one-hot-encoding is basically a vector of 0 of size k, k being the number of output classes that we have. In our case, as a reminder, this number is 33. Then, a 1 is placed in this vector, at the position of the document class that is predicted. We can also see it as the vector which is obtained after applying an $\arg\max$ operation on the vector of logits which is obtained after applying the softmax function. We will use both of these representations of the document classes in the different architectures we will now present.

2.3.2 Different Architectures and Their Results

The three architecture that are presented below were considered one after the other. For each of them, we present the architecture, including its inputs, as well as what kind of modelization is being trained; the advantages and drawbacks of each architecture, on different criteria, and finally the performance of each architecture on the task of correctly classifying the blocks presented to the model.

CRF Model with Document Class Information

The first architecture we considered is based on the CRF modelization we already used when considering *sequencing* over multimodal approach. We already saw in Section 2.2.3 that the input of this CRF consisted in the concatenation of the 768-dimensional *merged embedding* which is the output of the GMU model, and of the 4 sequential features that were common to all four CRF models.

For this architecture, we will add a third concatenated input to a CRF model, which is the document class information. We will in fact train two different CRF models. The first one will use the 288-dimensional *embedding* as document class information. This first CRF model will thus have $768 + 4 + 288 = 1\,060$ input features. The second one will use the one-hot-encoding of the document class as document class information, resulting in $768 + 4 + 33 = 805$ input features for the second CRF model. A graphical representation summarizing the CRF models is presented in Figure 2.13.

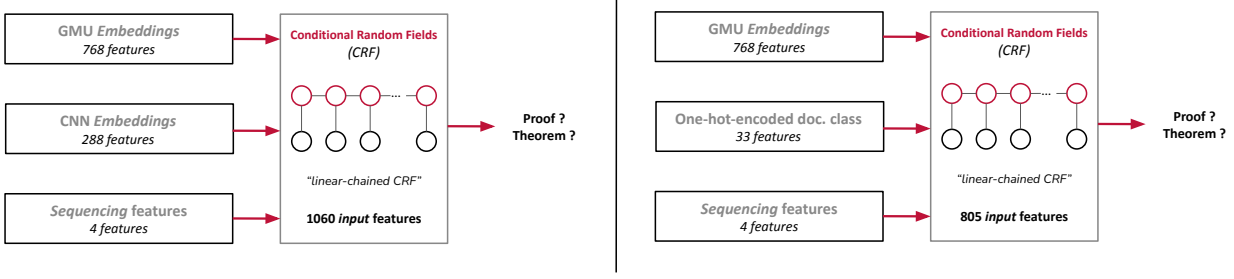

Figure 2.13: Diagram of the two CRF models using document class information

We can now analyze this modelization. Its main advantage consist in the fact that it does not require any GPU to be trained. Indeed, the weights of the CNN and GMU architectures are "frozen" (meaning that training a CRF does not requires to retrain the GMU and CNN architectures), and, as already explained in Section 2.2.3, the training of a CRF is itself costless in GPU. However, there are also major drawbacks to this architecture. First, it depends on a particular modelization of the *sequencing* of information, and we mentioned earlier that this is something we don't want, and this for two main reasons. The first one is that this modelization adds a "bias" to the study, because we cannot quantify, once we will present the results, the real impact that have the *sequencing* and therefore we cannot strictly quantify the impact of the document class. The second one is that it prevents us from using a different modelization of the sequencing: with this modelization we must stick to CRF. This point is really important, and we will be further explaining it in Section 2.3.3. The second major drawback is that it does not capture any connection between the GMU and the CNN modelizations. In fact, these ones are presented to the CRF as inputs, independently from one to another. This is a major issue because it should not connect the document class information with the information of the block.

This can be observed by training and evaluating the two architectures. The training is done using the *Violette* team's computer already presented before. We present the results with already presented macroscopic F_1-score and microscopic accuracy. We use as baseline, the CRF model trained with the multimodal approach *embeddings* (and *sequential* features) as input, to be fair in the comparison. The obtained results for the two architectures are presented in Table 2.8. We can see that the last identified drawback is observed in the results, because both performance metrics do not show an improvement in the results, and thus for both models. This leads to the idea that we need to choose a modelization that make connections between GMU and CNN possible. This is the core idea of the next architecture.

Table 2.8: Microscopic accuracy and macroscopic F_1-score for the CRF model with document class, compared to its baseline

Model	**Accuracy**	**F_1-Score**
CRF with multimodal (Baseline)	*84.38%*	*83.01%*
CRF with 1 060 features (CNN *embedding*)	84.49%	83.13%
CRF with 805 features (document class one-hot-encoding)	84.52%	83.11%

Dense Layers Connecting GMU And Document Class Information

This second architecture aims at catching at least some of the connection that might exist between document class information and blocks information, through the different modalities merged with the GMU architecture. There are several possible ways to try to catch such connections. We chose to use deep learning, and especially *Multi-Layers Perceptrion* (MLP) [19]. A MLP is composed of an *input* layer, an *output* layer and at least one *hidden* layer. Each component of each layer, which we call *neuron*, is connected to all *neurons* of the previous layer. We then talk about a "fully-connected" neural network, since each of these connections is associated with its own weight, all learned thanks to statistical learning.

We must thus define each of these components. The output layer is chosen regarding the nature of the problem. We want at the end to classify the blocks into either theorem environment, proof environment or none of them, so we want to have as many output *neurons*. We will just have to apply a softmax function on the outputted logits and an argmax to have a prediction. Now regarding the *input* layer, it has to be composed of the features representing the blocks and document class information. As it was mentioned above, we are taking the *merged embeddings* of the GMU model. For document class, we will again consider two different inputs and therefore two different models. The first one will be composed of the *embeddings* of the CNN model, resulting in a concatenated input layer of size $768 + 288 = 1\,056$ features (or *neurons*). The second model will use the one-hot-encoding of the document class for the document class information, and the *input* layer will thus be composed of $768 + 33 = 801$ *neurons*. Finally, we have to define the *hidden* layer(s). We made the choice to include two of them in the model, the first one being composed of 64 neurons, and the next one composed of 32 neurons. These number are a bit arbitrary, however it turned out by experimentation that adding more neurons does not really improve the performance. We still decided to add a dropout operation between

these two layers, to avoid overfitting, in particular if it turned out that there were not that many connections between the two modelizations, and that the model could be simplified (because, as already described in Section 2.1.3 with CNN, this dropout operation is randomly deactivating some of the neurons of the network to force the model not to overfit). By doing so, we end up having 69 860 parameters for the first model, considering all connections between the different layers, plus the *biases*. For the second one, we end up having 53 540 parameters, using the same reasoning. A graphical representation of such models is shown in Figure 2.14.

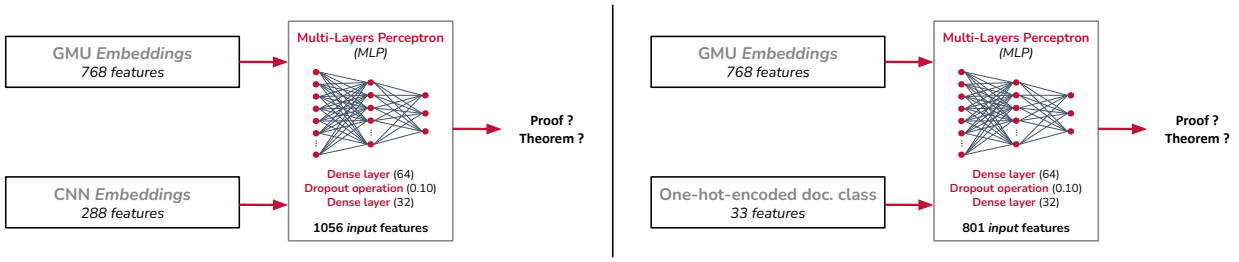

Figure 2.14: Diagram of the two MLP models with document class information

We can describe the advantages and drawbacks of the architecture of these models. First of all, it still does not require to retrain the GMU and CNN models, the weights still being "frozen". Then, this architecture is actually catching some connections between the blocks information and the document class information, if there are so. It might however not capture all potentially existing connections. This should improve the results. Finally, we can here study the impact of the document class in the task of identifying mathematical environments without being dependent on a certain modelization of *sequencing*, which is something we consider being important. As a consequence, this modelization seems to be better than the previous one, and this even before having a look at the results. However, there are a few drawbacks of this method. First, it is consuming GPU resources, in the training phase of the dense layers, because we are now dealing with deep learning techniques. The obtained results are presented in Table 2.9. We once again use the macroscopic F_1-score and the microscopic accuracy to evaluate the performance of the model. We compare these results to the ones of the GMU model without the CRF approach, to obtain a fair comparison once again. The training was done on *Jean Zay* supercomputer, using 16-GB Nvidia Tesla V100 GPU nodes.

Table 2.9: Microscopic accuracy and macroscopic F_1-score for the MLP modelization compared to its baseline

Model	**Accuracy**	**F_1-Score**
GMU model (Baseline)	*76.86%*	*73.87%*
MLP with 1 056 input features	78.85%	76.11%
MLP with 801 input features	78.92%	76.12%

We can see that this architecture does improve the results, in opposition with the previous one. The improvement is not outstanding, however it is still considerable giving the complexity of the task, and shows itself the relevance of the use of document class information in the task of detecting mathematical environments in the scientific literature. We also see that both models gives similar results. The one with one-hot-encoded document class information is giving slightly better results, while saving around 25% ($100 \times \frac{69860-53540}{69860} \approx 23.36\%$) of number of parameters: this is the one that we should keep. We will see in the discussion of the future work why this improvement is important.

Retraining of a GMU Including Document Class Information as New Modality

This last architecture also came out as an idea when facing the issues of the first one. It consists in observing that we trained a GMU model which was taking as input embeddings representing the same blocks with

different views. Since the GMU approach is a deep learning technique which tries to enhance the fact that merging different embeddings can improve performance on classification task, and given that we have at our disposal an *embedding* representing the document class information, we can imagine training a new GMU model considering an extra modality, which would be the one of document class information. This is the core idea of this third architecture. As a consequence, we will not consider the one-hot-encoding representation for the document class information, but only the *embeddings*.

A graphical representation of such an architecture is presented in Figure 2.15. We can see that it is just an updated version of Figure 2.10 which was already presenting the GMU architecture (unless we here removed the par about the input and cutting in blocks). This architecture has several advantages. First, this architecture will be able to catch deeper connections and interactions between block information and document class information. This is the reason why we thought about it actually. It will catch even more information than the last architecture, because it will be able to connect each original modality (vision, language and font), individually with the document class information, whereas last architecture was just trying to catch connections with the global, *merged* embedding. This approach also looses its dependency to a certain *sequencing* modelization, unlike the first one. However, it has important drawbacks that must be considered. The major one, which led us not to train it, and therefore not to be able to evaluate it, is its huge cost in both time and GPU resources. In fact, even if we would not retrain the weights of the CNN architecture, we would still need to train from scratch an entire GMU architecture, with even more features than the version presented in Section 2.2.2, taking thus even more resources. Given the fact that we had already trained the approach using MLP, we didn't need this one to show the impact of the document class. This choice is even more relevant since we must try as scientists, and especially scientists that are using deep learning techniques, to limit the use of GPU resources that can quickly have a bad environmental impact due to the important amount of greenhouse gases emissions generated by the use of supercomputers.

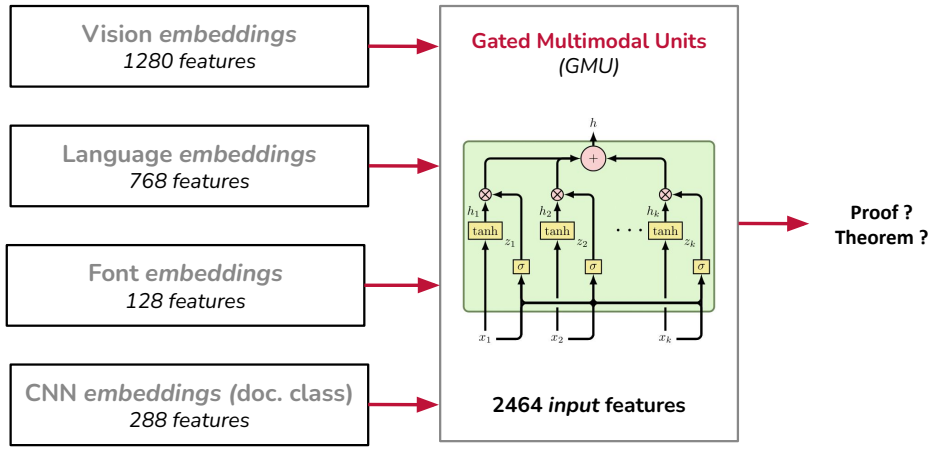

Figure 2.15: Diagram of GMU model with document class information

A last drawback that should be mentioned is that GMU model was initially developed to merge multimodal information, i.e., different views of a same information. This worked very well for the GMU modelling with three modalities, as they represented the same information. However, this time, by adding a document class modality, the *embeddings* presented to the GMU model no longer all represent the same information. The model could still perform satisfactorily, but there will remain a fundamental question about the meaning of what we are doing here, in particular regarding the explicability of the results obtained (even if this is a deep learning method anyway, which is therefore inherently difficult to explain).

2.3.3 Future of This Work

We thus conducted the study of the impact of the document class in the task of extracting the mathematical environments from scientific articles. In particular, the second architecture using MLP model shows that

document class information helps the classifier to correctly classify the blocks as either proof, theorem or none of them.

However, we saw earlier that the increase in performance metrics was not that outstanding. However, this work is still important, because it shows that document class information can be used in this classification task, no matter how we modelize it. Actually, we already consider some further improvements in this modelization. In particular, we want to change the way that we modelize the consideration of the *sequencing* of the blocks. This is the reason why we heavily insisted on the fact that the study must be conducted independently from a certain modelization of this *sequencing*, which was CRF in that case. We want now to consider some deep-learning techniques to consider the *sequencing*. One way to do so could be to use a Bi-LSTM modelization [9] over the GMU approach, using its *embeddings* as input. Beyond the fact that it should capture *sequencing* on a larger scope (considering blocks in both directions, and several of them), it also uses a *context*, which is here the information about the *sequencing*. This *context* must be initialized in the network, and will be updated while some new elements of the sequence is presented to the network. One thing we didn't mentioned before is that the document class information we present to the different architectures are redundant, no matter which representation of them we use, in the sense that the document class is the same for the entire article, and does not differ from one block to the other. Therefore, using the document class information as initial *context* would be beneficial from two point of views: first, it would avoid having this redundancy and therefore potentially save some features, while still being included in an (expected) more efficient architecture.

If we now take a step back from this classification task, we consider that the results that we have, and even more with the new results we expect to have with these improvements, will be stable enough so that we begin to extract the mathematical content that is contained in the blocks identified as theorems or proofs (we already reach nearly 85% on the classification task alone, i.e., without *sequencing*). Once this will be done, the natural next step of this work is to work on the next items of the TheoremKB project, presented in the first chapter of this report, on Figure 1.2, especially on the population of a knowledge base of mathematical results.

Chapter 3

Analysis of Developed Skills and Career Plan

Beyond the work that has been achieved during this internship, it is also important to think about what it brought me at different levels. In the first section of this chapter, I will present the different skills that I developed, both technical and relational. Then, in the second section, I will present the contribution of this internship to my career plan, by presenting what I plan on doing in the next few months as well as some additional tasks I worked on during this internship to prepare this next professional step.

3.1 Technical and Relational Skills

I had the opportunity to develop a lot of skills through this internship. First, the environment in which I evolved has allowed me to develop and strengthen a number of relational skills.

3.1.1 Relational skills

Indeed, research work requires acquiring a number of qualities that are essential to the conduct of quality scientific work. Among them, we can mention the rigor, especially in the experiments that are conducted. We must ensure at all times that we produce a thorough and scientifically rigorous work, so that it makes sense, and that is is relevant. Rigor therefore comes in tandem with ethics and especially scientific ethics. It is the responsibility of the scientist to present his results exactly as they are, without omitting details, including results that would not be positive. As I have had been told several times, exploring a possibility that proves to be fruitless is still a step forward in our work since we are eliminating a possibility. It is also important to be aware of the critical ethical aspects of the field in which we are working. This is a particularly important point in data science. Especially, concerning the data that is processed: we must respect the confidentiality of the data, in particular when it comes to personal data. We have not been confronted with this kind of treatment since the data considered in this work are available in open access online. However, for copyright reasons, we could not directly publish the data used for the work of Section 2.1. We therefore indicated the unique identifiers of the scientific articles concerned, as well as version information so that it is possible to find the exact same article used. This leads to another aspect of research ethics, particularly in computer science research, which is the repeatability of the experiments that are conducted. This is why we have endeavored to produce open access repositories presenting the data (or information relating to its data), trained models, as well as pieces of code to reproduce these experiments.

Other qualities, a little more practical, are also essential in order to conduct research properly. Organization first: even if from the outside research may seem a profession in which we try to lead randomly, until a success takes place (or not), in reality it is even more necessary to organize our work to avoid scattering. One of the main limiting factors in this work is time, which does not allow us to try all the existing possibilities.

This is something we experience particularly in artificial intelligence, where we cannot try every existing modelling that could potentially solve a problem. The presentation of this internship is a good example: I tried, for each task, only some of the candidate models. This is where reflection comes in, in order to make judicious and especially justifiable choices, in accordance with the assumptions we make, the constraints and the expectations we have. That is why another quality, which I think is mostly acquired through experience, is the ability to step back. Take a step back to stop exploring a path when it is inoperative, but also take a step back in the analysis of the results of our experiments. This step back not only makes it possible to produce a relevant critical analysis which therefore falls within the framework of the ethical aspects I mentioned earlier, but it also often makes it possible to become aware of elements which had not previously appeared to us, often leading to good progress in our work. This is what I have endeavored to do throughout this internship, and to make it apparent in this report.

As for the working methodologies, I had the opportunity to experience several. At first, I had to work in autonomy. This is something that I had already developed as part of my past experiences, but that I had the opportunity to reinforce during this internship. This was especially the case thanks to my internship tutor, who trusted me and allowed me to conduct my research work quite independently, while being available when I needed it. This therefore introduces a second work methodology, which is team work. I had several opportunities to work as a team during this internship. First of all, with my internship manager, of course, but also more broadly with the Valda team, and even with other teams in the "DI" department (in particular as part of the establishment of a new course in the *Louis-Le-Grand* CPES, jointly between the Valda and Talgo teams). In the Valda team as part of the work presented in Section 2.2, I had the opportunity to work with a doctoral student (Shrey Mishra), but also indirectly with the other trainees of the team (Ilyas Lebleu, Belkis Djaffal), and a post-doctoral student (Shufan Jiang). This work was done within the framework of an international team, and therefore also multi-cultural. These exchange times were certainly the ones in which I learned the most. I particularly appreciated the fact that everyone, regardless of their status and experience, could contribute their ideas and be listened to.

3.1.2 Technical skills

In order to carry out the work carried out during the internship, I had to mobilize knowledge already acquired, but also learn or strengthen certain technical skills, the main ones being as follows.

First, the development and training of statistical learning models allowed me to develop and strengthen my Python development skills. On the one hand through libraries of statistical learning, such as `keras`[1] and `tensorflow`[2] for deep learning, or `python-crfsuite` for probabilistic modeling, already presented earlier in this report. On the other hand, much of the work has also involved extracting information from raw data, and pre-processing it to make it usable by these learning models. This required improving my skills in libraries such as `pandas` [3], `scikit-learn`[4], `pdf2image`, or `grobid_client`[5]. Beyond knowing how to use libraries, I learned how to write effective code, and adapt the data to the different libraries used.

Another skill, linked also to the preparation of the data but also to their acquisition, is the strengthening of my skills on Linux systems. In particular, I learned to efficiently transfer data between servers and remote machines through SSH tunnels, but also to write Bash scripts to perform certain processing (especially to use the *GROBID* and *PDFAlto* software I presented in Section 2.2.2). I also had the opportunity, as already mentioned, to use the Jean-Zay supercomputer, and thus to learn how to use it. This includes the efficient management of the allocated storage space (divided into three partitions, for different uses), the management of a physical resource manager called SLURM, especially through the writing of scripts allowing the submission of work in their waiting list, in particular to target computational resources in line with the needs of experiments. To do this, we had to use Bash in order for SLURM to be able to execute the Python code I was talking about earlier.

[1] `https://keras.io/`
[2] `https://www.tensorflow.org/`
[3] `https://pandas.pydata.org/`
[4] `https://scikit-learn.org/`
[5] `https://github.com/kermitt2/grobid_client_python`

In addition, I also had the opportunity to acquire theoretical knowledge in statistical learning and mathematics associated with it. This was done through the study of the state of the art in the few areas I was confronted with during my internship. Examples include probabilistic modeling of random variable sequencing, deep learning in computer vision, or extracting information from PDF representations of documents for instance.

Finally, a last skill that we would call technical is the use of technical English. First orally, in order to successfully communicate on scientific subjects with other members of the team, but especially in writing. Indeed, writing reports and even scientific articles requires a certain methodology, both on the structuring of the content and on the content itself, which requires a specific vocabulary that I had the opportunity to learn and practice throughout the internship.

3.2 Contribution of this Internship to my Career Plan

This internship, which also ended my training as an engineer, contributed to the confirmation of my professional project. First, he definitely confirmed my interest in computer science, and more specifically artificial intelligence and data science. Indeed, I have been developing this attraction for several years, having chosen elective courses in this field in CI1 (CI means "cycle ingénieur") and specialty courses in CI2. I continued to develop my skills in artificial intelligence as well as in general computing, opening my field of study through a semester of study at UQAM (*Université du Québec à Montréal*) in Canada. Finally, I specialized in artificial intelligence and data management during my last year of study through an exchange at *Télécom Paris*. I was able to apply my learnings during this last internship, which definitely confirmed my attraction for these fields.

I also had the objective of continuing to discover research and confirming my wish to continue my professional career there. I already had the opportunity to do a research internship in computer science last year, within the Inria CEDAR team, a project team between Inria Saclay and the LIX (*Laboratoire d'Informatique de l'École Polytechnique*). This internship had already made me want to continue my engineering studies with a doctorate in the field of computer science. This last internship at the ENS definitely confirmed this wish on the one hand, but also gave me the opportunity to realize it on the other hand. That is why I will be pursuing a PhD within the ENS from the beginning of 2023. More specifically, this PhD will be co-supervised by Pierre Senellart, my current internship supervisor, and Ioana Manolescu, my internship supervisor from last year (from the CEDAR team). The thesis will have the following title: "Intelligent construction of a multimodal and heterogeneous data warehouse, with data traceability". The TheoremKB project is also partly in the scope of the tasks that will be carried out in this PhD, in particular as an application of knowledge extraction by multimodal approaches, and the construction of a knowledge warehouse.

In addition, I carried out several research-related tasks during this internship. First of all, I had to set up, together with my (future) thesis co-supervisors, application files for funding from different organizations, since no thesis funding was already held by them. Then, I continued to work indirectly and punctually on the themes of my internship last year, so with the CEDAR team. In particular, I participated with them in writing two scientific articles, submitted to conferences. The first is a demonstration [2], and was accepted in the *20th European Semantic Web Conference* (ESWC 2023). The second is a long research article [3], and was accepted in the *27th European Conference on Advances in Databases and Information Systems* (ADBIS 2023). Finally, I have more recently spent time preparing for future teaching activities in which I will take part from the beginning of my PhD, starting in September 2023. I will take part into three courses: in an introduction course to algorithmics in the first year of the *Louis-Le-Grand* CPES entitled "Sciences des données, arts et cultures"; in a practical introductory research course in the second year of *Henri IV*'s CPES in sciences and in a differential calculus course as part of the second year of this same last CPES.

Conclusion and perspectives

So, after 24 weeks spent with the Valda team, I managed to conduct the study I was given, and to show the relevance of the use of the document class in the automatic extraction of mathematical environments from the scientific literature. This study led to the production of deliverables, in the form of scientific articles, accepted or submitted for publication in conferences, and on the other hand in the form of repositories containing models and information on the data used in order to reproduce the experiments which have been conducted.

Indeed, I first showed that it was possible to automatically infer the document class of a scientific article from the image of the only first page of the PDF rendering of the article. This inference is made with a computer vision approach, which shows excellent results. I then improved a mathematical environment extraction model already developed by the team, from a simple observation that consists in considering a scientific article as a sequence of text blocks, and not as a simple set of such blocks. Using probabilistic modelling, I showed that this approach significantly improved the model's performance in this extraction task, despite a relatively simple modelling. Finally, I was able to exploit the inferred document class as input data, which, once combined with the data produced by the mathematical environment extractor in the form of various architectures, shows that the information carried by the document class simplifies the task of extracting mathematical environments, thus answering the problem set by this internship.

This overall work will certainly benefit the Valda team that welcomed me. Indeed, the study conducted allows, and will even more through the few improvements considered, to have sufficient confidence in the task of extracting mathematical environments in order to be able to move to the higher stage of the TheoremKB project. This will result in the establishment of knowledge bases, as mentioned above, but in the longer term the possibility of really exploiting these knowledge bases and thus unleashing the full potential of these bases. Moreover, since the work on the automatic inference of the document class was carried out independently of the application to the extraction of mathematical environments, it could be used for other works that would be related to extracting content from scientific articles. Since this work has been published at an international conference, it will certainly be used more widely in the scientific community working on these issues.

More personally now, this internship was also beneficial to me personally. First of all because it allowed me to develop relational skills, but also scientific skills, both practical and theoretical, which will be very useful for me in the rest of my career. But also because it confirmed my deep desire to continue working in scientific research in data science. I will have the opportunity to continue my studies through a PhD, within this same Valda team, but also within the CEDAR team. This PhD, which will cover various aspects of heterogeneous data science and data management, will also include some of the work to be done in the scope of the TheoremKB project, as an application. The study that was conducted will therefore also directly benefit myself in the context of my own, future, work.

I will therefore have the chance to see and take part in the future developments of this work, and in the progress of the TheoremKB project in the upcoming years.

Bibliography

[1] Arevalo, J., Solorio, T., Montes-y Gómez, M., and González, F. A. Gated multimodal networks. *Neural Computing and Applications 32*, 14 (Jul 2020), 10209–10228.

[2] Barret, N., Gauquier, A., Law, J.-J., and Manolescu, I. PathWays: entity-focused exploration of heterogeneous data graphs. In *20th European Semantic Web Conference (ESWC 2023)* (Hersonissos (Crete), Greece, May 2023). Demonstration paper.

[3] Barret, N., Gauquier, A., Law, J.-J., and Manolescu, I. Exploring heterogeneous data graphs through their entity paths. In *27th European Conference on Advances in Databases and Information Systems (ADBIS 2023)* (Barcelona, Spain, Sept. 2023).

[4] Baum, L. E., and Petrie, T. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics 37*, 6 (1966), 1554–1563.

[5] Breiman, L. Random forests. *Machine Learning 45*, 1 (Oct 2001), 5–32.

[6] Gauquier, A., and Senellart, P. Automatically Inferring the Document Class of a Scientific Article. In *23rd ACM Symposium on Document Engineering (DocEng 2023)* (Limerick, Ireland, Aug. 2023).

[7] Ginev, D., and Miller, B. R. Scientific statement classification over arXiv.org. In *LREC* (2020).

[8] Goodfellow, I. J., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016, ch. 9. `http://www.deeplearningbook.org`.

[9] Graves, A., and Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks 18*, 5 (2005), 602–610. IJCNN 2005.

[10] He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *Computer Vision – ECCV 2016* (Cham, 2016), B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Springer International Publishing, pp. 630–645.

[11] Huang, Y., Lv, T., Cui, L., Lu, Y., and Wei, F. LayoutLMv3: Pre-training for document ai with unified text and image masking. In *ACM MM* (2022).

[12] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML* (2001).

[13] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Comput. 1*, 4 (dec 1989), 541–551.

[14] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692* (2019).

[15] Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 11966–11976.

[16] Mishra, S., Gauquier, A., and Senellart, P. Multimodal machine learning for extraction of theorems and proofs in the scientific literature. Under review.

[17] Mishra, S., Pluvinage, L., and Senellart, P. Towards Extraction of Theorems and Proofs in Scholarly Articles. In *Proc. DocEng* (Limerick, Ireland, Aug. 2021).

[18] Mohammed, R., Rawashdeh, J., and Abdullah, M. Machine learning with oversampling and undersampling techniques: Overview study and experimental results. pp. 243–248.

[19] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review 65*, 6 (1958), 386–408.

[20] Sammut, C., and Webb, G. I., Eds. *TF–IDF*. Springer US, Boston, MA, 2010, pp. 986–987.

[21] Tan, M., and Le, Q. Efficientnetv2: Smaller models and faster training. In *Proceedings of the 38th International Conference on Machine Learning* (18–24 Jul 2021), M. Meila and T. Zhang, Eds., vol. 139 of *Proceedings of Machine Learning Research*, PMLR, pp. 10096–10106.

[22] Tang, R., Adhikari, A., and Lin, J. Flops as a direct optimization objective for learning sparse neural networks. In *CDNNRIA* (2018).

[23] Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., and Zhou, M. LayoutLM: Pre-training of text and layout for document image understanding. In *SIGKDD* (2020).

[24] Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., Lu, Y., Florencio, D., Zhang, C., Che, W., et al. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In *ACL/IJCNLP* (2021).

[25] Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Los Alamitos, CA, USA, jun 2018), IEEE Computer Society, pp. 8697–8710.