Projet d'Initiation à la Recherche de 4ème année IR

Natural Language Processing (NLP) applications in scientific context

Pierre Laur

José Daniel Organista Calderón

Paul Gervais

Djihadi Ahamdy

Keziah Sorlin

Amine Arfa

Sana Rafik

4th Year - Department of Informatique et Réseaux

Institut National des Sciences Appliqués de Toulouse

Tutor: M. Hassan Hassan, M. Drira Khalil

May 2022

Acknowledgments

We would like to express our special thanks to our tutors Hassan Hassan and Khalil Drira, as well as Abdel Kader Chabi Sika Boni, who gave us the opportunity to discover this subject and helped us when needed.

We also thankful to the direction of the GEI department of INSA for giving us access to their servers.

Projet d'Initiation à la Recherche de 4ème année IR

Natural Language Processing (NLP) applications in scientific context

Pierre Laur

José Daniel Organista Calderón

Paul Gervais

Djihadi Ahamdy

Keziah Sorlin

Amine Arfa

Sana Rafik

4th Year - Department of Informatique et Réseaux

Institut National des Sciences Appliqués de Toulouse

Tutor: M. Hassan Hassan, M. Drira Khalil

May 2022

Table des matières

1. Introduction 2
2. State of Art 3
   A. Introduction 3
   B. Applications 4
      B.1. Text classification (TC) 4
      B.2. Information Retrieval (IR) & Text Ranking (TR) 4
      B.3. Other Applications 5
   C. Overview of NLP models 5
   D. Learning constraints 8
      D.1. Overfitting 8
      D.2. Underfitting 8
      D.3. Underfitting vs overfitting 9
      D.4. Lack of data 9
   E. Evaluation 9
   F. Scientific NLP 12
   G. Conclusion 13
3. Implementation 15
   A. Web Application 16
      a. Front-End 16
      b. API 17
   B. Search Engine 17
      a. A neural text re-ranking architecture 17
      b. Preliminary BM25 implementation 18
      c. Adapting CEDR's architecture for our application 19
      d. Testing the model 20
      f. Running the system on a server 21
      g. Fine-tuning the model with an appropriate dataset 21
4. Project management 22
5. Conclusion 24

1. Introduction

Browsing scientific literature is a considerable part of the work in any research project, and this task is made even longer by the likelihood of trawling trough irrelevant papers to make sure not to miss something important. While we should expect this process to be partly unavoidable, it could be made shorter with better-performing natural language processing (NLP) algorithms.

The aim of this project was to respond to that need by making an improved, dedicated search engine. In this project, we studied recent advancements in NLP and their potential application in scientific context. We then applied our findings to make a web search engine for scientific literature, using state-of-the-art technologies in NLP to enhance arXiv's basic search results. The report is organised as follows: in chapter 2, we present the state of the art of NLP algorithms with a special focus on models used in scientific context. In chapter 3, we present our implementation of the web interface as well as the search engine, and in chapter 4 we present the project management method we used to achieve this work. Finally, we conclude with some remarks and directions for future work.

2. State of Art

Abstract

Deep Learning (DL) technology is responsible for significant advances in many different areas: cancer diagnosis and self-driving cars, for example, have been impacted by the rise of neural networks. Natural Language Processing (NLP), the branch of artificial intelligence that helps machines understand natural languages, has also made remarkable progress thanks to DL in recent years. Pre-trained models (PTMs) particularly stand out as the most used technological solution in state-of-the-art systems for NLP applications.

We investigate these advances to summarize NLP uses and their potential application in scientific context. As our goal is to build a web portal for bibliographical research, we give particular attention to NLP applications that may be used for this project. We discuss scientific NLP and its specific features, along with state-of-the-art systems for NLP applications in a scientific context.

Keywords: *natural language processing (NLP), deep learning, neural networks, scientific NLP, pre-trained model*

A. Introduction

Natural Language Processing (NLP) refers to a branch of computer science, specially to a branch of artificial intelligence (AI) that provides a bridge between natural languages and computers [1]. As NLP systems can process and understand human language, they can be used to rank documents by relevance to a user's query, cluster documents into categories or translate texts into different languages.

To process human language, NLP uses various models and methods. These models build structured representations for the input data, which can then be analyzed. NLP methods, meanwhile, are the different approaches to exploiting models to solve an application.

Advances in deep learning have allowed new NLP methods to emerge; these methods include transfer learning, the use of pre-trained models to solve specific tasks such as document ranking. Pre-trained models (PTMs) are neural networks trained on general language understanding tasks that can be adapted (fine-tuned) to a specific task with a limited amount of labeled data. In recent years, state-of-the-art performance on NLP applications has mainly been achieved by deep learning systems, as evidenced by their prevalence on NLP benchmarks such as GLUE [2]. These modern approaches have been supported by increases in computational power, enabling users to train neural networks that can contain billions of parameters [3].

Due to the differences between natural languages, NLP systems are not the same for English and other languages. However, methods can be re-employed between languages that are relatively close: for example, the RoBERTa model [4], originally conceived for the English language, has been adapted to process French [5]. Likewise with scientific English here we investigate how whether NLP models can be successfully used in a scientific context, despite scientific English's differences with general domain English.

This state of the art has six sections. Section 2 presents the different NLP applications. Section 3 discusses several models that are used to perform these applications and how they work. Section 4 lays out the constraints associated with deep learning technology and how they affect NLP systems. Section 5 summarizes the most common evaluation methods for NLP

systems. Finally, section 6 discusses the specific features of scientific NLP.

B. Applications

NLP has a significant impact in our everyday life; it is used in many modern systems such as translators, web search engines and automatic speech recognition systems. Its applications can be classified in many ways. We use a classification proposed by Otter et al. [6] in their survey: if we exclude speech recognition systems that can be associated to a different field, NLP systems can be said to perform seven applications: Text Classification (TC), Information Retrieval (IR), Information Extraction (IE), Question Answering (QA), Machine Translation (MT), Text Summarization (TS) and Text Generation (TG).

This section describes these NLP applications. We will focus on TC and IR, since they have the most potential to be useful for bibliographical research.

B.1. Text classification (TC)

Text classification models *preprocess* data, then use statistical or machine learning models to assign text into classes. Classes may be, in the case of bibliographical research, documents that discuss a specific topic, and documents that do not.

Traditional approaches for TC are either rule-based or rely on machine learning [7]. Rule-based systems classify text based on handwritten rules: for example, a classifier could assign newspaper articles to the "Sports" category if they include several sports-related words that are superior to a certain threshold. On the other hand, traditional ML approaches such as Naïve Bayes are given relevant features that characterize texts and output a class as a function of these features. They are trained on labeled datasets to adjust the function, until they can accurately predict the right class when given the value of these features in a text.

These methods are relatively simple and can be useful, especially when only a limited amount of data is available. However, to achieve good results, they require "feature engineering", which takes a significant amount of time. Modern DL methods require more labeled data but are significantly more effective [7]. They rely on neural networks, which obviate the need for feature engineering, and considerably expand the number of trainable parameters.

While TC systems are used for tasks such as news categorization, they can only be used on predefined classes, which is very limiting for bibliographical research. They may be used for simple domain classification (biomedical / computer science / psychology…).

B.2. Information Retrieval (IR) & Text Ranking (TR)

The goal of an Information Retrieval (IR) application is to help users find relevant information in a collection of documents. Many IR systems such as web search engines play a considerable role in helping us deal with 21st century information overload.

A particular type of IR application known as Text Ranking consists of generating an ordered list of texts retrieved from a corpus in response to a query [8]: as opposed to TC, text ranking makes a *graded* relevance judgment. This application may be very useful for our project, which consists of retrieving scientific documents that are most relevant to a user's query.

Simple statistical methods for IR and TR rely on calculating the frequency of specific words in documents. In the case of a search engine, these words may be extracted from the user's query, resulting in a list of the documents that use them most frequently that approximates the list of the

most relevant documents to answer the query. BM25 is an example of a statistical model for information retrieval that is still widely used today. Many high performing TR systems such as Sledge-Z [9] use it as a first-stage ranker, retrieving the 100 or 1000 most relevant documents before re-ranking them with a more precise model.

State-of-the-art performance of IR models has been significantly improved by neural networks, especially since the release of Google's pre-trained model BERT [10]. Likewise, TR systems have been steadily improving, as evidenced by the evolution of top scores on text ranking leaderboards such as Microsoft's MSMARCO [11],[12].

Table 1: Mean Reciprocal Ranking (MRR) comparison of three text ranking models on MSMARCO's document ranking leaderboard (https://microsoft.github.io/msmarco/)

	BM25	**BERT-base**	**UniRetriever**
MRR@100	0.201	0.317	0.440

Neural IR systems build and train representations of the interactions between the words in both the query and the documents [6]. This allows for better language understanding, with the context of the words and their interactions giving crucial information to estimate the document's relevance to the user's query or perform any other IR-related task.

B.3. Other Applications

B.3.1. Information Extraction

The purpose of Information Extraction is to extract information from a document. To do that, the application goes through segmentation and labeling, then treats the information extracted from the document. Information Extraction can be seen as a part of the Information Retrieval process.

B.3.2. Question Answering

Question Answering's goal is to find documents that might contain an answer which can be handled by traditional information retrieval/web search. QA systems extract pieces of text from the documents and attempt to provide an answer to the question by using information from these texts.

B.3.3. Text Summarization

Text Summarization is a process of reducing the size of the original document while preserving its information content and its summary is less than half of the main text [13].

B.3.4. Text Generation

The principle of Text Generation is to generate text that resembles human writing. An example of this application is JarvisAI [14].

B.3.5. Machine translation

The goal of this method is to automatically translate a text from a certain language into another. It uses similar texts to find the best possible translation for the required sentences. Deep learning makes it approximately 10% more effective than statistical methods [15]. An example of this application is DeepL [16].

C. Overview of NLP models

With the development of deep learning, various models have been extensively used to solve natural language processing (NLP) tasks, such as convolutional neural networks (CNNs),

recurrent neural networks (RNNs), Long Short-Term Memory Networks (LSTM) and models based on attention mechanisms [50]. This section discusses several models that are used to perform these applications and how they work.

C.1. Classical Models Used

Before Deep Learning models became common thanks to their better performances, machine learning models such as Naïve Bayes, k-nearest neighbors, and random forests, were widely used to solve NLP tasks. During the past few years new approaches such as neural models and attention mechanisms, have been developed, replacing or enhancing these models [6].

Neural networks are inspired by the human brain. When they are given training data, it is processed, verified, and learned by improving its accuracy over time. Different applications of Neural networks (NN) have been used to solve natural language processing tasks [17]. There are different classes of Neural Networks, like Convolutional Neural Networks (CNN) and Recursive Neural Networks (RNN). CNN and RNN impacts many areas such as sentence classification [18], modeling sentences [19] and the analysis of natural language sentences [20].

C.2. Pre-trained Models and Transformer architecture

In recent years, Deep Learning technology has been taking over NLP, improving over the previous models results on most NLP applications. These advances have been driven using pre-trained models (PTMs) like OpenAI's GPT and Google's BERT [10].

One of the advantages of PTMs is that they extract knowledge from massive amounts of pre-trained data and can be reused for specific applications with very little training time. For instance, OpenAI's GPT-3 has 175 billion trainable parameters and can produce human-like text. BERT is pre-trained on a large corpus of unlabeled text, including the entire Wikipedia. These two PTMs owe part of their success to their use of the Transformer architecture (Fig. 1).

This architecture consists of an Encoder and a Decoder structure. The task of the encoder is to map an input sequence to a sequence of continuous representation. The decoder receives the output of the encoder to generate an output sequence. At each step the model is auto regressive, consuming the previously generated symbols as additional input when generating the next [51]. By encoding entire sentences, this architecture allows PTMs to take into account word dependencies, and achieve better language understanding.

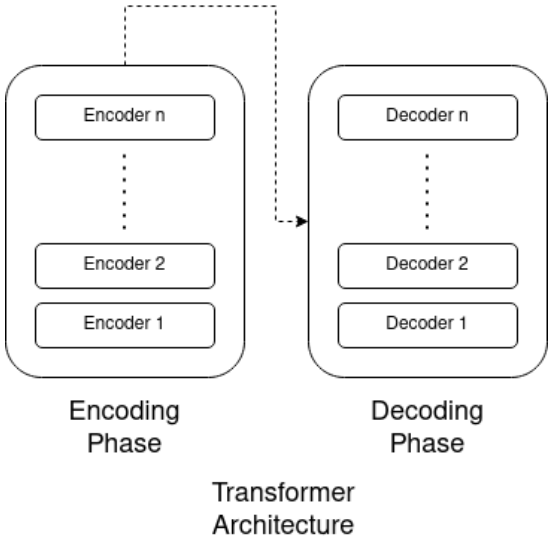

Fig. 1: Transformer architecture.

C.3. BERT

BERT (Bidirectional Encoder Representations from Transformers) is a PTM released by Google that has had considerable influence on NLP. BERT uses the encoder block to learn information from

both the left and right side of a token's context during the training phase (Fig. 2), while OpenAI GPT uses unidirectional (left-to-right) architecture to learn general language representations, every token can only attend to previous tokens [10].

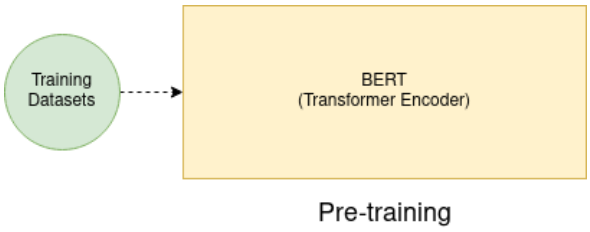

Fig. 2: BERT Base Model pre-training phase.

BERT is pre-trained on general language understanding tasks with large training datasets. This pre-training phase makes the model capable of adapting to a NLP task without requiring major changes to its architecture. This can be done by simply adding an additional layer to the pre-trained model and making it go through a tuning phase (Fig. 3). This characteristic allows the creation of state-of-the-art models for various NLP tasks.

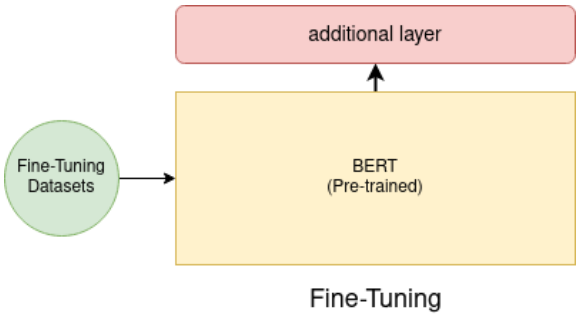

Fig. 3: BERT Base Model tuning phase.

C.4. Models based on BERT

BERT's considerable influence on NLP can be observed by looking at any NLP benchmark such as GLUE [2], where a significant part of the high-performing models is based on BERT's specific architecture and often contain BERT in their name. Many adjustments have been made to BERT, to produce other models that are better suited for specific needs.

PTMs, including those based on BERT, perform differently depending on their usage. For instance, we are mainly interested in the versions of BERT that are used in the scientific domain and can tackle the issues related. We also shed light on the models that may be useful for scientific document ranking which is highly mandatory for our research.

C.4.1. DistillBert model

The Distill version of Bert (DistilBert) has in fact fewer parameters than the original ones (40%) [22]. It also runs 60% faster and is able to ensure 95% of the performance according to GLUE which is a widely used Natural Language Understanding benchmark. In fact, it included Q/A data, in particular the Stanford Question Answering Dataset. All these characteristics make this model convenient and time saving.

C.4.2. Albert model

The Albert model which stands for "A Lite Bert', by changing a few things in Bert's architecture, this model achieves a great performance. The main difference between this model and the classical Bert is that the parameters of the word embedding are factorized resulting smaller matrices [23] and reducing the computational cost.

C.4.3. RoBERTa model

The main goal of this model is to optimize the training of Bert so that it takes less time [4]. We notice that it is mainly trained with dynamic masking, full sentences without NSP loss and a large byte level.

C.4.4. monoBERT and duoBERT model

The monoBERT and duoBERT are models proposed in the context of document ranking [24]. MonoBert is a classification model for document relevant and duoBert is a novel extension of monoBert that adopts an approach of pairwise to document relevance.

C.4.5. BioBert

BioBert is a pre-trained model used for biomedical text mining [25] in order to extract valuable information regarding this field. Initially, Bert was trained on English wikipedia and texts, but BioBERT improved it with additional pre-training on biomedical papers. Indeed, the biomedical field contains a considerable number of domain specific terms which are better dealt with specialized systems.

C.4.6. SciBERT

The sciBERT model is pre-trained on a large multi-domain corpus of scientific publications. This model shows better performance compared to BERT when applied to NLP tasks in a scientific context [26].

To summarize, we have seen different models, each one with diverse characteristics. For instance, Bert and GPT are both based on the transformer architecture, but they are different in a sense that Bert only takes the encoder blocks from the initial architecture while GPT takes the decoder blocks. Furthermore, BERT aims for bidirectional conditioning which makes it more efficient.

D. Learning constraints

Deep learning models have demonstrated great performances on a wide variety of NLP tasks, using many trainable parameters to extract detailed knowledge from large amounts of data [27]. However, underfitting, overfitting and the lack of data are common problems on these models [28],[29]. In this section, we explain why these problems occur and explore different approaches that are used to solve them.

D.1. Overfitting

Overfitting occurs when models are trained to achieve zero or near-zero training error. This produces a model that is precisely tailored to its training data. As a result, the algorithm cannot perform as expected against unseen data, which defeats its purpose. If trained for too long, models can reach a point where they are not able to perform as well as they could [30].

To avoid this issue, there are two main strategies: reducing the number of dimensions of the parameter space or reducing the effective size of each dimension [31]. In particular, pruning [32], [33], weight sharing [34], regularization, dropout [28] and early stopping [35] help to handle the overfitting problem.

D.2. Underfitting

Underfitting occurs when the data model is incapable of grasping a clear relationship between the input and output variables. As a result, this generates an error rate on the training data and the generated data. [36]

When the underfitting is detected, we can reduce it and make more convenient predictions, by using decrease regularization which can be used mainly to lessen the variance with a model by applying a penalty to the input parameters with the larger coefficients. There are several different methods, such as L1 regularization, Lasso regularization, dropout, etc... Otherwise, underfit can be avoided by increasing the duration of training if there is enough training data available.

D.3. Underfitting vs overfitting

Underfitting and overfitting are two opposite events, this mainly happens when the model has been trained too much and contains issues, therefore a high error rate is noticed on the data. In essence, overfitting a model is actually more common than underfitting one, technically underfitting is used to avoid the upcoming of an overfitting issue that is manifested through a process called "early stopping" [36].

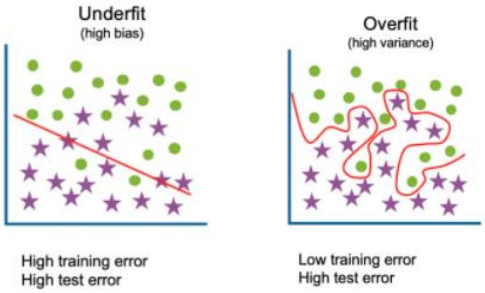

*Fig. 4: Underfit vs Overfit (***IBM Cloud Education [36])**

D.4. Lack of data

	Predicted Positive	Predicted Negative
Actual positive	True Positive (TP)	False Negative (FN)
Actual negative	False Positive (FP)	True Negative (TN)

Missing data is a real-world problem often encountered in scientific settings. Data that is missing is problematic as many statistical analyses require complete data. When this problem occurs, it is common to make assumptions about the missing values [37]. In fact, there are algorithms based on this approach.

For instance, Random Forest (RF) missing data algorithms are considered efficient to input missing data. In fact, they could deal with various types of missing data, and cope with nonlinearity, interactions, and scale huge data settings. Despite the important number of (RF) algorithms, their efficacy and performance hasn't been approved yet [38].

Large datasets are also needed to train Deep Learning models. While many exist for several NLP applications, the lack of large-scale labeled data can be an issue in specific contexts, such as scientific papers. We will discuss this particular issue in section 6.

E. Evaluation

Evaluating NLP models is crucial to assess which models are better. For a given application, a model should be judged in a **systematic** and **stable** way so we can objectively compare it to other models. This section will discuss metrics and datasets used for NLP tasks.

E.1. Common metrics for NLP tasks

Various metrics have been developed to measure the ability of NLP models to perform tasks.

Used for the evaluation of the performance of a classification, the confusion matrix is a table layout with the prediction frequencies in a real problem.

Table 2: Confusion Matrix

The accuracy metric is used in classification to measure how often an algorithm will classify data correctly. However, it is not a reliable indicator of its effectiveness in case of unbalanced datasets.

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP}$$

Precision is a model which measures the impact of the false positives. Higher the precision is, the less our algorithm confuses a group class with others.

$$Precision = \frac{TP}{TP + FP}$$

Recall is the frequency of documents correctly assigned to a class relative to the total number of documents belonging to the same class. The higher the recall is, the more our algorithm identifies one group class. it measures the number of positive individuals/documents that we missed.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score is a model widely used in the context of unbalanced data because it does not consider false negatives. It is therefore to be preferred over accuracy in the case of an imbalanced class situation.

$$F1\text{-}Score = 2 . \frac{Precision * recall}{Precision + recall}$$

There are variants of F-Score using the F1-Score for each label which is called "Weighted" by computing the weighted average. "Macro" calculates the F1-Score of each label then makes an average. "Micro" makes an average with the total of the true positives, negatives, and false positives/negatives and computes the F1-Score.

Specificity, on the other hand, measures the proportion of negative elements correctly identified.

$$Specificity = \frac{TN}{TN + FP}$$

Using this specificity and the recall, we can draw the ROC Curve which shows recall (or sensitivity) as a function of specificity. It is useful to compare some models and choose the optimal threshold. The larger the area under the curve is, the better our model is.

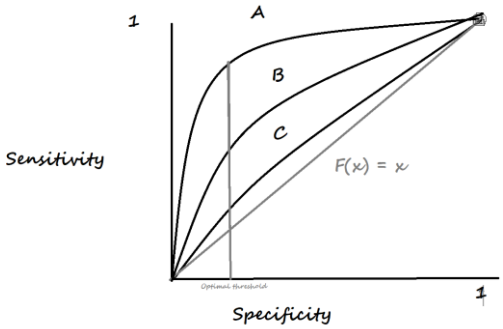

Fig. 5: F(specificity)=Sensitivity

In this graph, the best model is represented by model A with its largest area under the curve. The optimal threshold corresponds to the point closest to the ideal (1.1) and at the same time furthest from the diagonal.

Perplexity is an estimation of how well a likelihood model predicts a test information. With regards to Natural Language Processing, perplexity is one method for assessing language models. Lower is our perplexity, better are our results because perplexity is the exponential of our entropy.

Normalized discounted cumulative gain also known as nDCG [52], is a measure of ranking quality specifically designed for graded relevance judgments. It is most frequently used to measure the quality of web search results.

The Mean Reciprocal Ranking, also known as MRR, is a measure for the evaluation of the document search ranking process. When people click on an article, the position of the article in the list is registered, and a weight is

calculated so that the search displays our selection as the first result.

$$\text{MRR} = \frac{1}{|N|}\sum_{i=1}^{|N|}\frac{1}{rank\ i}$$

E.2. Datasets & Leaderboards

Good evaluation of NLP models can only be made with quality labeled datasets: their performance has to be compared with what is considered to be an *ideal* performance and efficiency. For instance, some metrics for document ranking such as nDCG [52] use the output document's **relevance**, manually assigned to each document in the dataset for a given test query.

A model's performance, as measured by any metric, depends on the dataset used. Therefore, a solid comparison between two NLP models has to be done with the same dataset, using the same metric. Since many datasets exist, models are not always compared on the same datasets. Leaderboards such as GLUE [2] are a good way to compare multiple models' side by side: they offer labeled datasets to test NLP models on many different tasks. The top-scoring models are easily identified, and anyone can create a model and compare it to the other models' results on these leaderboards.

E.3. Issues when evaluating

Table 2: Comparison between two document ranking NLP models: Nogueira et al.'s multi-stage document ranking system [24] and CEDR-KRNM [39], original and as implemented by Yang et al. [35].

	BM25+ monoBERT+duo BERT [24]	**Original CEDR-KNRM [39]**	**Yang et al.'s CEDR-KNRM [35]**
Robust 04 (nDCG @20)	•	0.5381	0.4637
MSM ARCO passag e (MRR @10)	0.370	•	0.344

Table 2 illustrates the problem where available evaluations are not enough to estimate which model is better between CEDR-KRNM [39] and BM25+monoBERT+duoBERT [24]. Different datasets and different metrics have been used, which makes it difficult to compare these models. Yang et al. [35] have made their own implementation of CEDR-KNRM, which ranked lower than BM25+monoBERT+duoBERT on the MSMARCO [11] passage retrieval leaderboard. However, their score is also lower than the original implementation on the Robust04 dataset: as a result, we cannot conclude on either model's superiority for document ranking.

This situation where we do not have a way to accurately compare NLP models has been made less common by initiatives like GLUE and TREC. However, difficulties when comparing models can still occur most high-performing NLP systems are very recent, and we are often limited to a comparison based on a single dataset, and/or a single implementation of the models. The most precise solution consists of implementing, testing, and training the models, but it takes time.

F. Scientific NLP

Scientific documents differ from general domain documents in several ways. As a result, NLP systems must consider this contrast to achieve good results on most applications. This section will discuss differences in **vocabulary**, **sentence splitting**, and the **training and testing** process between general domain and scientific NLP. We will also mention out-of-text criteria for scientific document ranking.

F.1. Differences in vocabulary

Frequently used words differ substantially between scientific texts and general domain texts.

Beltagy et al. [26] built a vocabulary for their SciBERT model from a large multi-domain scientific corpus, and found a 42% overlap when comparing it with BERT's base vocabulary. Gururangan et al. [40] compared RoBERTa's general domain vocabulary with scientific vocabularies and found only 19.2% overlap in computer science and 27.3% in biomedical papers.

When using a PTM trained on general domain data like BERT for scientific NLP tasks, this difference in vocabulary can lead to a high out-of-vocabulary (OOV) ratio, and thus affect the model's performance. As an example, Garneau et al. [41] measured the accuracy difference between all words and OOV words for several OOV handling algorithms and 2 NLP tasks and found between 11% and 21,5% for part-of-speech tagging. This partly explains SciBERT's [26] results, which are significantly superior to BERT's on core scientific NLP tasks.

We have identified three possible ways to address this issue.

- The first option consists of **using a PTM trained on scientific data**, such as SciBERT [26] - SciBERT is based on BERT but is not pre-trained on the same data. This option has been used by Macavaney et al. [42] to create Sledge-Z, a high-performing ranking algorithm based on SciBERT and fine-tuned to document ranking in Covid-19 literature.
- The second option consists of adapting an existing PTM by training it on scientific data. The result is **a PTM trained on general domain *and* scientific data**: an example would be Lee et al.'s BioBERT [25].
- The third option, called exBERT [43], consists of **incorporating scientific vocabulary in the original BERT-type model**. It is done by learning what the authors call a "small extension module" and combining the original word embeddings with the module.

F.2. Differences in sentence splitting

Scientific texts use abbreviated names, noun compounds and a wide range of citation styles which may contain punctuation. Standard sentence segmentation algorithms may encounter difficulties when dealing with these. Custom rule-based segments can be used to avoid this problem, but the process is slower.

Neumann et al.'s ScispaCy [42] achieved a better performance than the original spaCy [44] model in sentence segmentation without rule-based sentence segmentation by training on in-domain (biomedical) texts: sentence splitting was done correctly more often, with a faster algorithm. SciBERT [26] used it to build vocabulary.

While this specificity seems to impact sentence splitting performance, it is less relevant for systems that do not construct a vocabulary from scratch. These include systems based on PTMs, whose vocabulary has been built during their pre-training.

F.3. Training & testing scientific NLP models

Like any model, scientific models are ideally trained and tested on datasets that match their intended usage: these datasets should be made from scientific literature and evaluate the right application(s).

For instance, SciBERT [26] being a general model for understanding scientific text, it has (at publication time) been tested on scientific datasets that evaluate performance on core NLP tasks. BioBERT [25] was specifically designed to understand biomedical text - it was also evaluated on core NLP tasks, but on biomedical datasets.

Therefore, dataset availability can be a limit: large-scale scientific datasets do not exist for every imaginable application, and although transfer learning can be used to reduce the amount of specialized data needed [45], a substantial amount is always needed for fine-tuning and testing models.

When this problem occurs, it is sometimes possible to deal with domain mismatch by creating a subset of a dataset that evaluates the same application - the result is a smaller but appropriate dataset for the required scientific domain. This has been done to train COVID-19 document ranker "Sledge-Z" [9]: a subset of the MSMARCO document ranking dataset was created to select medical documents. This approach gave significantly better results than the version trained on MSMARCO, even though the training data was 10 times smaller.

F.4. Out-of-text criteria for scientific document ranking

To match human precision in the process of selecting useful scientific documents, algorithmic ranking systems must also consider some criteria that do not appear in the text itself.

Publication date is an important one: even if a scientific paper is judged relevant based on language understanding, it may be too old to be useful. This is particularly important in rapidly advancing fields, where a significant breakthrough can make previous works much less relevant. COVID-19 document ranker Sledge-Z [9] addresses this with a simple date filter: papers published before 2020 are filtered out of the corpus. It is not always as simple as that, but a scientific document ranker should give higher relevance scores to more recent papers, as humans do.

Famous scientific web search engine Google Scholar has been shown to use **citation count** as its most important criteria. Highly cited papers have been judged by the field to be particularly important, so a document ranking system should judge them as more relevant. However, as mentioned by Beel & Gipp [46], Google Scholar's high attention to citation count makes the search engine less appropriate to investigate new ideas and views that differ from the main interests of the field. **Other research metrics** such as **h-index** or **journal impact factor** may also help NLP systems in giving appropriate relevance judgments.

Finally, **publication status** (i.e., pre-print, published) should also be considered: while pre-prints should be given an appropriate amount of trust, they can still give useful information about very recent advances. This has been shown to be very relevant by the COVID-19 pandemic: preprints have been shared and read more [47], in a "data sharing" intent particularly advocated in times of medical crisis [48].

However, the importance of these criteria depends on what the user wants to do. A good document ranking system should ideally allow the user to control their weight in the algorithm's relevance judgment.

G. Conclusion

To conclude, we have seen different NLP applications that are used in countless

fields. In our case, with scientific bibliographic research, we shed light on the most useful applications for our study which is to stay Text Classification and Information Retrieval and try to tackle the issues related. We are also interested in ranking documents based on the topic query to reach a high level of precision in selecting suitable articles. We have looked at a panoply of models as defined in the sections above each one with its unique characteristics and uses. BERT stands out as a particularly versatile model, but it has been outperformed in many domains, including scientific NLP, by other models. SciBERT is a particularly good option in this field.

Multiple options seem to be viable for highly accurate scientific document ranking: fine-tuning SciBERT, using another PTM with additional pre-training in scientific context, and adding vocabulary embeddings to a PTM with the exBERT method are the three methods that seem to be a particularly good fit.

3. Implementation

The implementation of the web site is decomposed into two communicating parts, the front-end, and back-end. The front-end renders two main web pages to the final user: the search page, where a text input box is shown, and the result page, where a list of the top-ranked articles is displayed. The back end is composed by an API server and the search engine.

Fig. 6: Web interface of the search engine

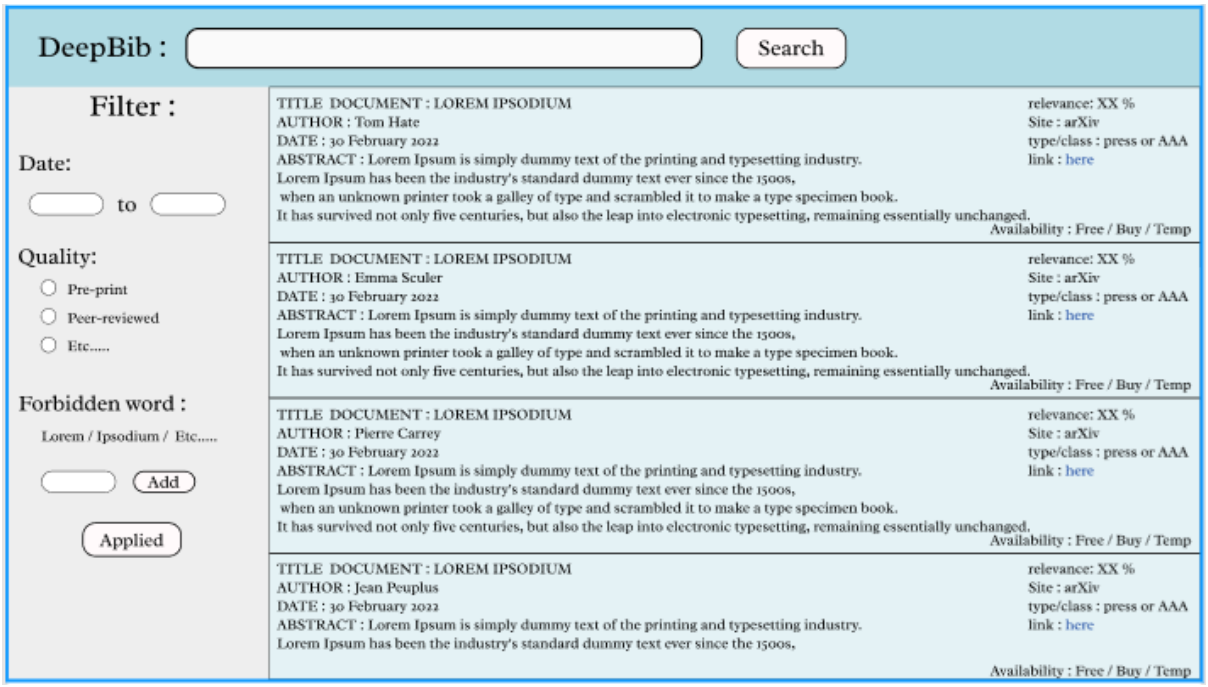

Fig. 7: Results of the search

The API component oversees getting the user's request and the top 100 documents from libraries such as arXiv (APIs responses) to send it to the search engine. Then, the search engine re-ranks the articles using a neural network to advise the user for his bibliographic research.

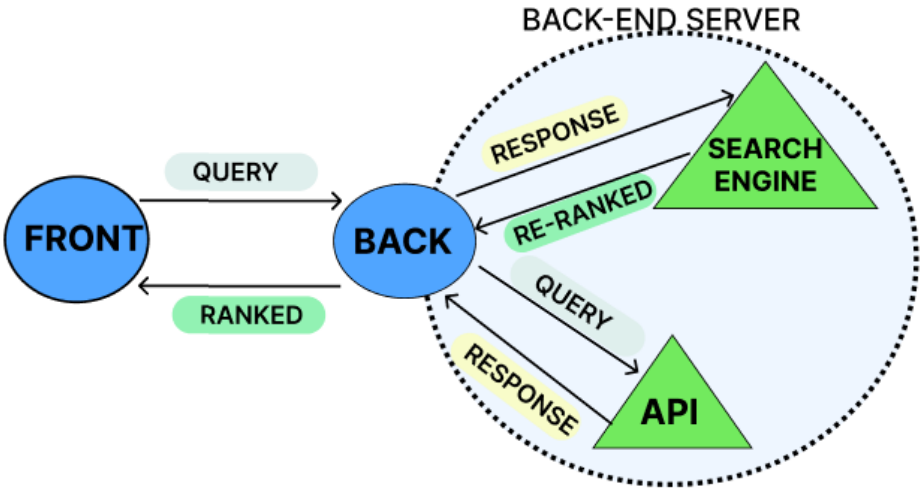

Fig. 8. DeepBib architecture

A. **Web Application**

The web application is the interface presented to users. An user can fill a form with his request and select a source of scientific articles. The request is then processed to the selected API (for now, only the arXiv API is available). The query results in a list of documents that are then re-ranked by our engine and displayed on the user interface.

a. Front-End

Due to the interaction with other autonomous entities, API, search engine and user, our web interface should respect interactive and synchronous behaviour. To provide a good user experience, we decided to develop our interface with ReactJs coupled with CSS bootstrap. ReactJs provides a real-time view for the user for each step of our application and CSS bootstrap will provide a good user experience. For the development of the user interface, we start with a sketch of the different perspectives.

Firstly, we focus on the graphical part with CSS. Each visual component of our application is defined and attached to a CSS class. This same component will be used for the ReactJs development to provide a synchronous behaviour. Then to ensure the synchronisation of shared variables, we have to decide precisely where they will be defined, and which component will interact with them. The next diagram represents each component used by our web application and its data.

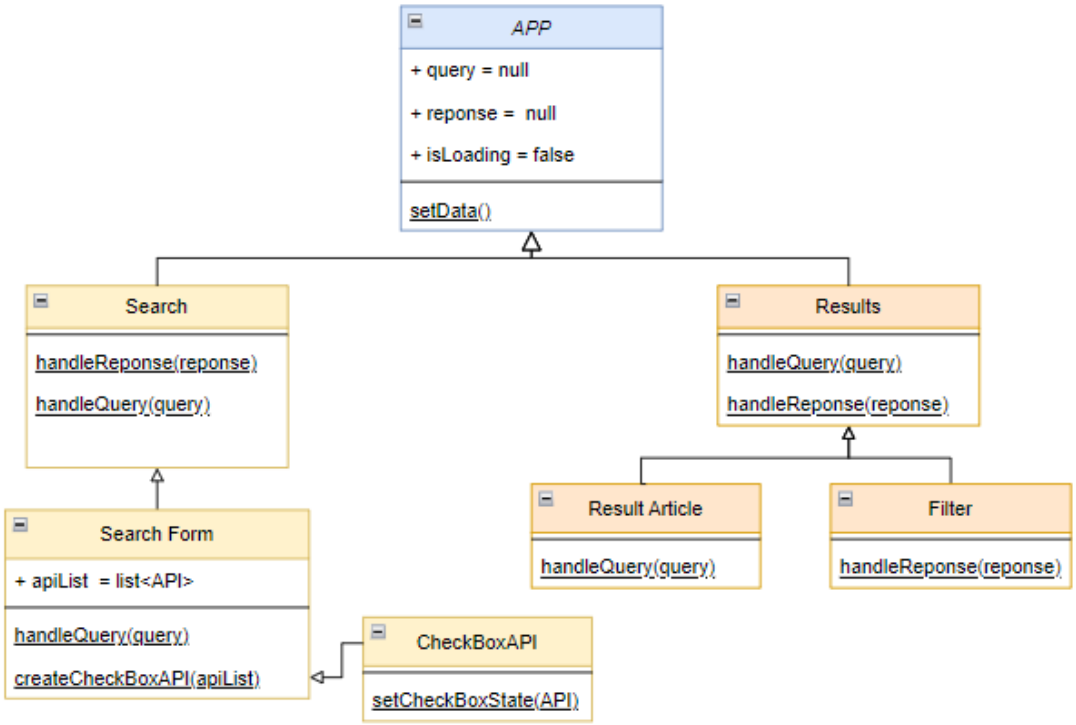

Fig. 9: Interface class diagram

b. API

To collect documents for our search engine to rerank, we used the arXiv's API. An API allows two applications to connect through request and response. Each request is sent to a location named API endpoint. The API is composed of two parts: the client in charge of making and manipulating the request and the server in charge of sending the information. In order to make one request processable by the endpoint, the client must provide a uniform resource locator (URL), a method, a list of headers and a body. For example, for the API of arXiv the base url is :

http://export.arxiv.org/api/{method_name}?{parameters}

After that the API provides a Json file which stores all the documents needed for the search engine. This json file is composed of all documents retrieved by the API, along with metadata such as date, author, source, etc. We format the abstract and number the documents for the search engine, which re-ranks the documents.

B. Search Engine

a. A neural text re-ranking architecture

To obtain the best possible results, we used one of the top-performing technological solutions for scientific document ranking. We found that pre-trained models based on BERT variants perform the best, as evidenced by their scores on leaderboards such as MSMARCO [11].

These general language understanding PTMs are generally adapted to the document ranking task by adding additional layers in a specific layout: we used Macavaney et al.'s CEDR architecture [39] for this purpose.

Since neural ranking models can be slow, they are often used in combination with a simpler model such as BM25. Despite being less precise, this faster model is able to quickly rank the entire corpus of documents, and output an approximation of the top N (i.e. in MSMARCO's re-ranking task, top 200). The neural ranking model can then re-rank this smaller corpus to give the most relevant ones.

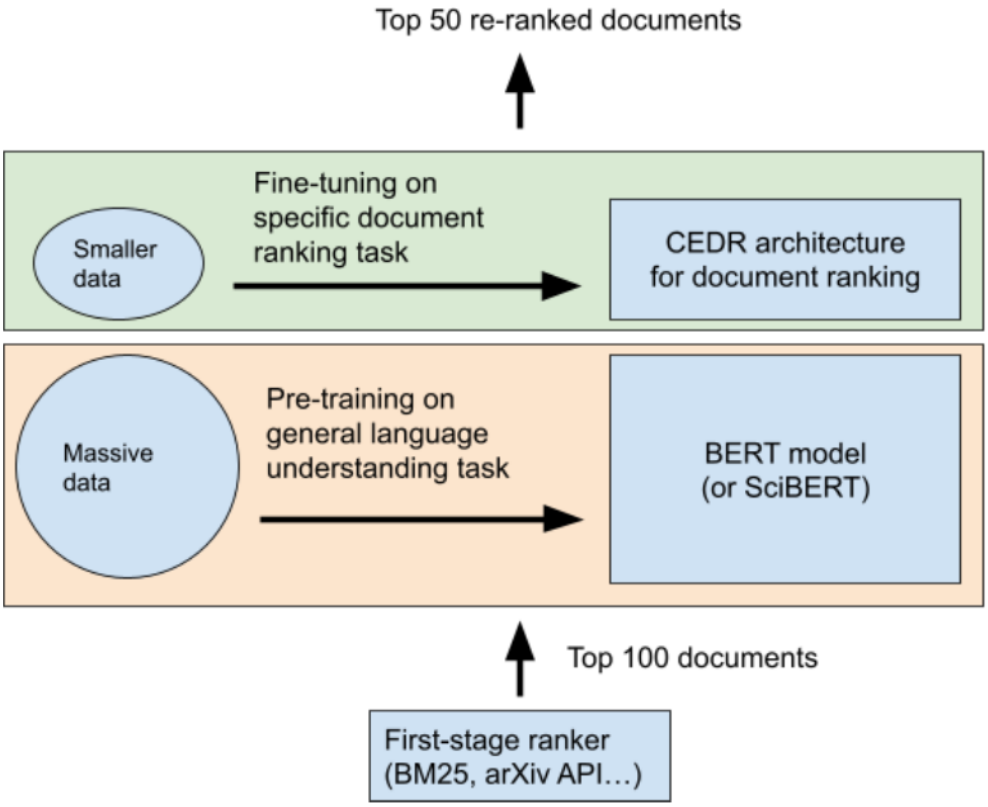

Fig. 10: Neural re-ranking architecture

b. Preliminary BM25 implementation

We first implemented a BM25 ranker without knowing that it was already used in the libraries we were working with. BM25 is a "bag of words" function, it does not take into account the order of the words: it ranks the occurrence frequency of a word in a query.

$$\mathrm{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$$

We implemented in python a pdf to csv converter then a document word tokenizer and then used a BM25 function library to score our document.

$$\mathrm{score}(D,Q) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\mathrm{avgdl}}\right)}$$

Where q_i is a term's frequency and

Fig. 11: BM25 Function

However, we estimated that this preliminary ranking of the corpus of documents was not useful, since we could directly retrieve a smaller corpus from the arXiv API already using BM25.

c. Adapting CEDR's architecture for our application

To perform document ranking, we adapted Macavaney et al.'s code [39], which was available on GitHub, to be able to automatically perform document ranking from a given query and a list of documents. The queries and documents have to be written in a specific format for automatic processing:

```
doc     A7      Research in applying natural language processing (NLP) techniques
doc     A8      This report describes an NLP assistant for the collaborative develo
doc     A9      There are a lot of tools and resources available for processing Fi
doc     A10     With recent developments in new architectures like Transformer and
doc     A11     Numbers are essential components of text, like any other word toke
doc     A12     Recent years have seen many breakthroughs in natural language proc
doc     A13     The rise of big data analytics on top of NLP increases the computa
doc     A14     Driven by the visions of Data Science, recent years have seen a pa
doc     A15     NLP's sphere of influence went much beyond computer science resear
doc     A16     Deep learning has become the dominant approach in coping with vari
```

[type] [number] [text]

For re-ranking, we also must produce a file to indicate the scores obtained by every query-document pair. We generate this file automatically with a python script.

```
f  = open("data/arxivRun","w")
for docnumber in dataset[1] :
   f.write("0       Q0      "+docnumber+"   "+docnumber+"   100.0   run\n")
f.close()
```

The neural ranking model can then be used to re-rank the documents. The output indicates, for each document (identified by a number), the associated relevance score given by the model. These results can then be used by the front-end.

```
5 4.1635541915893555
12 4.098103046417236
4 3.9031996726989746
7 2.593170166015625
1 2.391307830810547
6 2.389174222946167
10 2.265540599822998
9 2.116832971572876
13 2.0749588012695312
8 1.3113917112350464
2 1.1798205375671387
3 0.8788994550704956
11 0.6990375518798828
```

d. Testing the model

For a given list of queries and documents, our system generates the documents in order of relevance, along with their relevance scores. By defining simple queries, we can test our system on a list of documents. With a few simple tries, we verify that the search engine gives appropriate relevance judgments.

We retrieve the top 100 documents from arXiv with queries “natural language processing” (identified with letter A), “natural selection” (letter B) and “backpropagation” (letter C) and we ask our system to rank the 300 documents for these three queries. If the system works, most of the top results for the query “natural selection” should be identified with B, and so on.

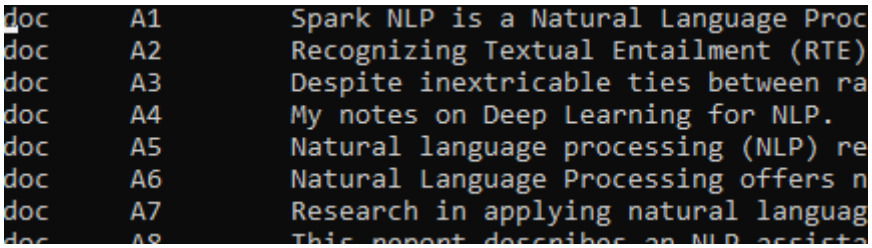

Table 3: Number of false positive documents in the top 50 retrieved by the re-ranker

Architecture used	Natural language processing	Natural selection	Backpropagation
Vanilla BERT	0-1	1-2	0-1
CEDR-PACRR	1	0	0

Despite the vocabulary overlap between queries A and B, and the fact that both A and C are related to Deep Learning, we obtain very few false positives. While this does not allow us to do a precise evaluation of our system, these results are very promising, given that our system has not been fine-tuned.

0. Using another pre-trained model: SciBERT

As explained in our state of the art, we found that the SciBERT [26] model was the best available option for scientific NLP: its pre-training on scientific texts allows it to obtain better scores on scientific NLP leader boards than other pre-trained models.
However, we were not able to implement the model: we encountered several problems when trying to adapt the backend architecture to implement this model. We left the possibility of using SciBERT for a future second version of our system.

1. Running the system on a server

To train the neural ranking model and use it to re-rank documents, it is necessary to use a machine with GPUs. Due to the high computational cost of deep learning, the machine should preferably be very powerful. We also needed a server to run our system, to always make the website available. For these two purposes, we investigated two options: using Amazon Web Services, or using one of INSA's available machines with powerful GPUs.

Different problems were encountered due to the lack of permissions on the INSA computers. For instance, it was at first impossible to install tools for deployment to modify GPU metrics. On the other hand, with the different amazon services, the problem of the lack of permission was solved, but other problems appeared such as the limit of use due to the use of free tier accounts. Besides, the deployment considered the learning curve of amazon services as no member of the team had ever used these services before.

In the end, the solution used was the INSA computers because despite the lack of permissions to modify the project it was found to be more functional than the amazon services.

2. Fine-tuning the model with an appropriate dataset

For optimal document ranking performances on scientific articles, the pre-trained model should ideally be fine-tuned with a dataset. This training dataset should be labeled for document ranking and composed of scientific texts, to match our system's intended task.
Sledge-Z [9] is an example of a neural ranking system that has been fine-tuned for document ranking in a medical context, using a subset of a very large document ranking dataset, MSMARCO [11]. The authors removed all the questions that did not include medical terms, using a lexicon made by Yates and Goharian [53]. This solution could be applied to our case with a similar lexicon designed for scientific text.
Another way to build a dataset for the scientific document ranking task would be to gather many scientific systematic reviews and use their title and list of citations as query-document pairs. The papers cited in these systematic reviews have been judged most relevant by experts and could be automatically labelled as the most relevant for their respective topics.
Due to time constraints, we have not done this task yet. However, CEDR's code offers several models that we can use without further training, by keeping the neural networks in their initial states. These models have been evaluated by the authors [39] and give very satisfying results on the Robust04 and WebTrack 2012-14 datasets for document ranking, even without fine-tuning. We chose to use them, and to leave the fine-tuning for a future version of the system.

4. Project management

4.1. Research and formation phase

None of us had strong Machine Learning and NLP skills. To be able to understand our project, we followed some online courses about machine learning and NLP applied to text. After those courses, we all had a basic understanding of both machine learning and NLP.

4.2. Assigning work to each member according to skills

We've decided to split the work by assigning tasks according to each member's skills. Jose Daniel Organista had previously worked with API and directed this section. Djihadi Ahamdy has strong skills in web development and decided to lead this part. Pierre and Keziah were interested in working on the document ranking by NLP. Paul was interested in each part and especially helped Jose on the API part.

4.3. Following the progress of each part by planning weekly meeting

Each week, we planned a meeting to monitor the progress of the project overall. Dr. Hassan led and supervised the DeepBib project and broke some deadlocks in our work. We set the objectives and the new tasks at the end of every meeting, and then every member of our team worked on average around 4 hours a week on their assigned tasks. Since we also had to deal with other projects and exams, it was necessary to stay focused on our tasks every week and the team followed what was initially scheduled.

The API section worked in parallel on Springer, arXiv and I3E explorer where arXiv is the most relevant API. The NLP section studied BM25, BERT and CEDR. Djihadi did the web and graphic design at the same time as the front-end.

We overestimated our productivity by planning the end of the project at the end of April and were delayed because our machines were not powerful enough. We tried to work with AWS but were not successful. Fortunately, our school gave us the rights to use their powerful machine learning server.

When members finished their parts, they started to support other team tasks.

The whole team got together to work on the CEDR project on Docker. This solution allowed us to package the application into containers, to execute our models in the same environment, despite using different operating systems.

We also used Git with our application because it allows all members to work together and contribute to the project, it also enables users to showcase their work and see the work done by others on a regular basis.

We created two git repositories for our application, one for the front end and the other one for the back end. All members could contribute to the deployment of the application by pushing the modifications and the work done during different classes.

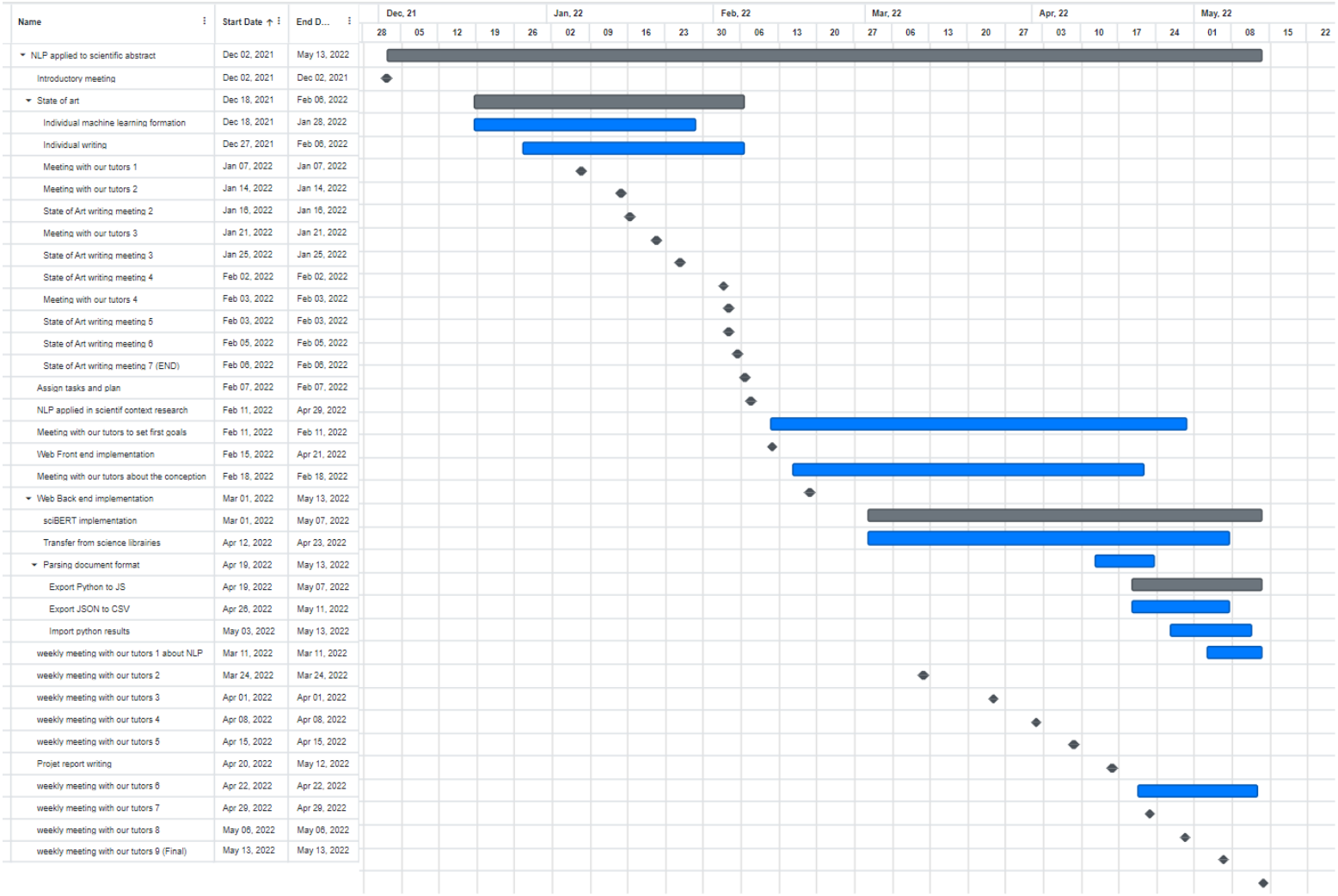

Figure 12: Gantt Diagram

5. Conclusion

Bilan

In this work, we present a first version of DeepBib, a web search engine for bibliographic research. Our system leverages a neural network for natural language processing to improve arXiv's search results. We use MacAvaney et al.'s CEDR [39] architecture for document ranking, based on BERT, a pre-trained model for natural language processing.

Perspectives

We regret not having been able to deploy the search engine on a server, fine-tune our neural ranking model, or use the SciBERT model for better quality ranking. Aside from these three things, there could be several ways to further improve our system. As suggested in our state of the art, relevance judgments could include out-of-text criteria that may be used by experts to judge if scientific papers are relevant: publication date, citation count and other research metrics could contribute to the relevance scores of query-document pairs. This solution would require tuning weights for these additional criteria, either manually or with an appropriate dataset.

Using full papers instead of abstracts could also vastly improve our system's relevance judgments, but this could only be done with full access to the arXiv database. Using several APIs to retrieve documents from several libraries may also improve our results. Finally, giving particular importance to keyword matches (scientific paper keywords occurring in the user's query) would be a sure way to improve our system.

We would be very glad to see our work on this project reused and improved by other students, and may work on it ourselves in the future. We are grateful for the insight this project gave us on natural language processing, the use of deep learning models, and bibliographic research.

References

[1] A. Torfi, R. A. Shirvani, Y. Keneshloo, N. Tavaf, et E. A. Fox, « Natural Language Processing Advancements By Deep Learning: A Survey », 2021 [Online]. Available at: http://arxiv.org/abs/2003.01200. [Consulté le: 4 février 2022]

[2] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, et S. R. Bowman, « GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding », 2019 [Online]. Available at : http://arxiv.org/abs/1804.07461. [Consulté le: 4 février 2022]

[3] R. Raina, A. Madhavan, et A. Y. Ng, « Large-scale deep unsupervised learning using graphics processors », in *Proceedings of the 26th Annual International Conference on Machine Learning*, New York, NY, USA, 2009, p. 873-880, doi: 10.1145/1553374.1553486 [Online]. Available at: https://doi.org/10.1145/1553374.1553486. [Consulté le: 4 février 2022]

[4] Y. Liu *et al.*, « RoBERTa: A Robustly Optimized BERT Pretraining Approach », 2019 [Online]. Available at: http://arxiv.org/abs/1907.11692. [Consulté le: 4 février 2022]

[5] L. Martin *et al.*, « CamemBERT: a Tasty French Language Model », *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 7203-7219, 2020, doi: 10.18653/v1/2020.acl-main.645 [Online]. Available at: http://arxiv.org/abs/1911.03894. [Consulté le: 4 février 2022]

[6] D. W. Otter, J. R. Medina, et J. K. Kalita, « A Survey of the Usages of Deep Learning in Natural Language Processing », 2019, pp. 1–21, doi: 10.1109/TNNLS.2020.2979670 [Online]. Available at: http://arxiv.org/abs/1807.10854. [Consulté le: 4 février 2022]

[7] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, et J. Gao, « Deep Learning Based Text Classification: A Comprehensive Review », 2021 [Online]. Available at: http://arxiv.org/abs/2004.03705. [Consulté le: 25 janvier 2022]

[8] B. Mitra et N. Craswell, « An Introduction to Neural Information Retrieval », p. 119, 2018 [Online]. Available at: https://ieeexplore-ieee-org.gorgone.univ-toulouse.fr/document/8620670. [Consulté le: 04 février 2022]

[9] S. MacAvaney, A. Cohan, et N. Goharian, « SLEDGE-Z: A Zero-Shot Baseline for COVID-19 Literature Search », 2020 [Online]. Available at: http://arxiv.org/abs/2010.05987. [Consulté le: 4 février 2022]

[10] J. Devlin, M.-W. Chang, K. Lee, et K. Toutanova, « BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding », 2019 [Online]. Available at: http://arxiv.org/abs/1810.04805. [Consulté le: 4 février 2022]

[11] P. Bajaj *et al.*, « MS MARCO: A Human Generated MAchine Reading COmprehension Dataset », 2018 [Online]. Available at: http://arxiv.org/abs/1611.09268. [Consulté le: 5 février 2022]

[12] « MS MARCO ». [Online]. Available at: https://microsoft.github.io/msmarco/. [Consulté le: 4 février 2022]

[13] Munot, N., & S. Govilkar, S. "Comparative Study of Text Summarization Methods. International Journal of Computer Applications", vol. 12, n°12, p.33–37, 2014, [Online]. Available at: https://doi.org/10.5120/17870-8810 [Consulté le: 4 février 2022]

[14] « DeepL Traduction – DeepL Translate : le meilleur traducteur au monde ». [Online]. Available at: https://www.DeepL.com/translator. [Consulté le: 4 février 2022]

[15] M. R. Costa-jussà, « From Feature To Paradigm: Deep Learning In Machine Translation », Journal of Artificial Intelligence Research, vol. 61, pp. 947-974, 2018, doi: 10.1613/jair.1.11198

[16] Jasper (Formerly Jarvis) - #1 AI Writing Assistant ». [Online]. Available at: https://www.jarvis.ai/. [Consulté le: 4 février 2022]

[17] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural Language Processing (almost) from Scratch,” 2011 [Online]. Available at: https://arxiv.org/pdf/1103.0398.pdf. [Consulté le: 13 janvier 2022]

[18] Y. Kim, « Convolutional Neural Networks for Sentence Classification », 2014 [Online]. Available at: http://arxiv.org/abs/1408.5882. [Consulté le: 4 février 2022]

[19] N. Kalchbrenner, E. Grefenstette, et P. Blunsom, « A Convolutional Neural Network for Modelling Sentences », 2014 [Online]. Available at: http://arxiv.org/abs/1404.2188. [Consulté le: 4 février 2022]

[20] R. Socher, C. C.-Y. Lin, A. Y. Ng, et C. D. Manning, « Parsing Natural Scenes and Natural Language with Recursive Neural Networks », p. 8, 2011 [Online]. Available at: https://icml.cc/2011/papers/125_icmlpaper.pdf. [Consulté le: 4 février 2022]

[21] S. Hochreiter and J. Schmidhuber, « Long Short-Term Memory », in Neural Computation, vol. 9, n°8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[22] Samuel Kierszbaum, Laurent Lapasset. « Applying Distilled BERT for Question Answering on ASRS Reports.», NTCA 2020 New Trends in Civil Aviation, Nov 2020, Prague, Czech Republic. pp.33-38, 10.23919/ntca50409.2020.9291241.hal-03094753f

[23] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, et R. Soricut, « ALBERT: A Lite BERT for Self-supervised Learning of Language Representations », 2020 [Online]. Available at: http://arxiv.org/abs/1909.11942. [Consulté le: 4 février 2022]

[24] R. Nogueira, W. Yang, K. Cho, et J. Lin, « Multi-Stage Document Ranking with BERT », 2019 [Online]. Available at: http://arxiv.org/abs/1910.14424. [Consulté le: 16 janvier 2022]

[25] J. Lee *et al.*, « BioBERT: a pre-trained biomedical language representation model for biomedical text mining », *Bioinformatics*, 2019, doi: 10.1093/bioinformatics/btz682, [Online]. Available at: https://arxiv.org/ftp/arxiv/papers/1901/1901.08746.pdf. [Consulté le: 4 février 2022]

[26] I. Beltagy, K. Lo, et A. Cohan, « SciBERT: A Pretrained Language Model for Scientific Text », in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, nov. 2019, p. 3615-3620, doi: 10.18653/v1/D19-1371 [Online]. Available at: https://aclanthology.org/D19-1371. [Consulté le: 19 janvier 2022]

[27] A. Borghesi, F. Baldo, et M. Milano, « Improving Deep Learning Models via Constraint-Based Domain Knowledge: a Brief Survey », 2020 [Online]. Available at: http://arxiv.org/abs/2005.10691. [Consulté le: 4 février 2022]

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, et R. Salakhutdinov, « Dropout: A Simple Way to Prevent Neural Networks from Overfitting »,, J. Mach. Learn. Res., vol 15, bll 1929–1958, p. 30, 2014. Available at: https://dl.acm.org/doi/pdf/10.5555/2627435.2670313. [Consulté le: 4 février 2022]

[29] M. Belkin, D. Hsu, et P. Mitra, « Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate », 2018 [Online]. Available atr: http://arxiv.org/abs/1806.05161. [Consulté le: 4 février 2022]

[30] H. K. Jabbar et R. Z. Khan, « Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study) », in *Computer Science, Communication and Instrumentation Devices*, 2014, p. 163-172, doi: 10.3850/978-981-09-5247-1_017 [Online]. Available at: http://rpsonline.com.sg/proceedings/9789810952471/html/017.xml. [Consulté le: 4 février 2022]

[31] L. Prechelt, « Early Stopping | but when? », in Neural Networks: Tricks of the Trade, p.15, 2012 [Online]. Available at: http://page.mi.fu-berlin.de/prechelt/Biblio/stop_tricks1997.pdf. [Consulté le: 4 février 2022]

[32] B. Hassibi et D. Stork, « Second order derivatives for network pruning: Optimal Brain Surgeon », in *Advances in Neural Information Processing Systems*, 1993, vol. 5 [Online]. Available at:

https://proceedings.neurips.cc/paper/1992/hash/303ed4c69846ab36c2904d3ba8573050-Abstract.html. [Consulté le: 4 février 2022]

[33] A. Levin, T. Leen, et J. Moody, « Fast Pruning Using Principal Components », in *Advances in Neural Information Processing Systems*, 1994, vol. 6 [Online]. Available at: https://proceedings.neurips.cc/paper/1993/hash/872488f88d1b2db54d55bc8bba2fad1b-Abstract.html. [Consulté le: 4 février 2022]

[34] Steven J. Nowlan and Georey E. Hinton, « Simplifying neural networks by soft weight-sharing». In *Neural Computation*, 1992, [Online]. Available at: http://www.cs.toronto.edu/~fritz/absps/sunspots.pdf. [Consulté le: 4 février 2022]

[35] Y. Yang, Y. Qiao, J. Shao, M. Anand, X. Yan, et T. Yang, « Composite Re-Ranking for Efficient Document Search with BERT », 2022, doi: 10.1145/3488560.3498495. [Online]. Available at: http://arxiv.org/abs/2103.06499. [Consulté le: 4 février 2022]

[36] « What is Underfitting? », 25 mars 2021. [Online]. Available at: https://www.ibm.com/cloud/learn/underfitting. [Consulté le: 4 février 2022]

[37] J. L. Schafer et J. W. Graham, « Missing data: Our view of the state of the art. », *Psychological Methods*, vol. 7, nº 2, p. 147-177, 2002, doi: 10.1037/1082-989X.7.2.147.

[38] F. Tang et H. Ishwaran, « Random Forest Missing Data Algorithms », *Stat Anal Data Min*, vol. 10, nº 6, p. 363-377, 2017, doi: 10.1002/sam.11348.

[39] S. MacAvaney, A. Yates, A. Cohan, et N. Goharian, « CEDR: Contextualized Embeddings for Document Ranking », *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 1101-1104, 2019, doi: 10.1145/3331184.3331317.

[40] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, Noah A. Smith, « Don't Stop Pretraining: Adapt Language Models to Domains and Tasks », 2020 [Online]. Available at: http://arxiv.org/abs/2004.10964. [Consulté le: 4 février 2022]

[41] N. Garneau, J.-S. Leboeuf, Y. Pinter, et L. Lamontagne, « Attending Form and Context to Generate Specialized Out-of-VocabularyWords Representations », 2019 [Online]. Available at: http://arxiv.org/abs/1912.06876. [Consulté le: 4 février 2022]

[42] M. Neumann, D. King, I. Beltagy, et W. Ammar, « ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing », *Proceedings of the 18th BioNLP Workshop and Shared Task*, p. 319-327, 2019, doi: 10.18653/v1/W19-5034.

[43] W. Tai, H. T. Kung, X. Dong, M. Comiter, et C.-F. Kuo, « exBERT: Extending Pre-trained Models with Domain-specific Vocabulary Under Constrained Training Resources », in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online, nov. 2020, p. 1433-1439, doi: 10.18653/v1/2020.findings-emnlp.129 [Online]. Available at: https://aclanthology.org/2020.findings-emnlp.129. [Consulté le: 5 février 2022]

[44] Honnibal, M., & Montani, I, « spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing », *Sentometrics Research*, 2017. [Online]. Available at: https://sentometrics-research.com/publication/72/. [Consulté le: 4 février 2022]

[45] S. J. Pan et Q. Yang, « A Survey on Transfer Learning », *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no 10, p. 1345-1359, oct. 2010, doi: 10.1109/TKDE.2009.191.

[46] J. Beel et B. Gipp, « Google Scholar's Ranking Algorithm: An Introductory Overview », p. 6, 2009, [Online]. Available at: https://www.researchgate.net/publication/200610388_Google_Scholar's_Ranking_Algorithm_An_Introductory_Overview [Consulté le: 4 février 2022]

[47] N. Fraser et al., « The evolving role of preprints in the dissemination of COVID-19 research and their impact on the science communication landscape », PLoS Biol, vol. 19, nº 4, p. e3000959, 2021, doi: 10.1371/journal.pbio.3000959.

[48] N. L. Yozwiak, S. F. Schaffner, et P. C. Sabeti, « Data sharing: Make outbreak research open access », *Nature*, vol. 518, no 7540, p. 477-479, 2015, doi: 10.1038/518477a.

[49] Z. Wang, P. Ng, X. Ma, R. Nallapati, et B. Xiang, « Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering », 2019 [Online]. Available at: http://arxiv.org/abs/1908.08167. [Consulté le: 4 février 2022]

[50] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, et X. Huang, « Pre-trained models for natural language processing: A survey », *Sci. China Technol. Sci.*, vol. 63, no 10, p. 1872-1897, 2020, doi: 10.1007/s11431-020-1647-3.

[51] A. Vaswani *et al.*, « Attention Is All You Need », 2017 [Online]. Available at: http://arxiv.org/abs/1706.03762. [Consulté le: 6 février 2022]

[52] K. Järvelin et J. Kekäläinen, « Cumulated gain-based evaluation of IR techniques », *ACM Trans. Inf. Syst.*, vol. 20, no 4, p. 422-446, 2002, doi: 10.1145/582415.582418.

[53] A. Yates et N. Goharian, « ADRTrace: Detecting Expected and Unexpected Adverse Drug Reactions from User Reviews on Social Media Sites », in Advances in Information Retrieval, Berlin, Heidelberg, 2013, p. 816-819, doi: 10.1007/978-3-642-36973-5_92.

Table of illustrations

Figure 1: Transformer architecture……6

Figure 2: BERT Base Model pre-training phase……7

Figure 3: BERT Base Model tuning phase…….…...7

Figure 4: Underfit vs Overfit…….9

Figure 5: F(specificity)=Sensitivity……..……11

Figure 6: Web interface of the search engine……...15

Figure 7: Results of the search…….15

Figure 8: DeepBib architecture……16

Figure 9: Interface class diagram…….17

Figure 10: Neural re-ranking architecture……18

Figure 11: BM25 Function……...18

Figure 12: Gantt Diagram……23

Table 1: Mean Reciprocal Ranking (MRR) comparison of three text ranking models on MSMARCO's document ranking leaderboard……..……5

Table 2: Comparison between two document ranking NLP models: Nogueira et al.'s multi-stage document ranking system [24] and CEDR-KRNM [39], original and as implemented by Yang et al. [35]……11

Table 3: Number of false positive documents in the top 50 retrieved by the re-ranker……..20

Abstract

Deep Learning (DL) technology is responsible for significant advances in many different areas: cancer diagnosis and self-driving cars, for example, have been impacted by the rise of neural networks. Natural Language Processing (NLP), the branch of artificial intelligence that helps machines understand natural languages, has also made remarkable progress thanks to DL in recent years. Pre-trained models (PTMs) particularly stand out as the most used technological solution in state-of-the-art systems for NLP applications.

We investigate these advances to summarize NLP uses and their potential application in scientific context. As our goal is to build a web portal for bibliographical research, we give particular attention to NLP applications that may be used for this project. We discuss scientific NLP and its specific features, along with state-of-the-art systems for NLP applications in a scientific context.

Keywords: *natural language processing (NLP), deep learning, neural networks, scientific NLP, pre-trained model*