POLITECNICO DI TORINO

Master's Degree in Data science and engineering

Master's Degree Thesis

Automatic contradiction detection in clinical trial proposal documents

Supervisors

Prof. Paolo GARZA

Prof. Luca ANSELMA

Prof. Luca PIOVESAN

Candidate

Laurentiu Aurelian APOSTOL

11 2024

Summary

In the domain of clinical trials, the approval process represents a critical bottleneck of the initial stages. This approval is based on the writing and revision of clinical trial proposal documents that are evaluated and approved based on a variety of criteria, including document consistency. In this setting, a rejection heavily increases the total time of the overall process. The work presented in this thesis addresses the challenge of automated inconsistency detection in these documents, with the goal of reducing approval times by identifying potential issues before submission. In particular, this project is aimed at creating an automatic system for detecting contradictions between tables and text related to said tables. The methodology adopted for this work and the assessment of its results have been greatly influenced by a limited dataset, comprised of non machine readable files, limited computational resources and the added difficulty of a specialized subfield. This work does, however, still potentially represent a starting point for the creation of tools that can help domain experts in this field during the first stages of a clinical trial proposal, even in the absence of curated datasets and with limited computational resources.

The initial stages of the work envisioned a generalized pipeline for contradiction detection, however this approach proved impractical upon analysis of the actual error types present in the documents. The evaluation of available materials revealed two predominant categories of issues: inconsistencies between tables and their associated text and missing text according to a given standard and/or guideline. Given these findings and given that the problems are very different in nature and require very different solutions, the approach chosen was that of trying to solve them one at a time by creating an ensemble system with different components focused on the different common issues and inconsistencies found in these documents. Hence, the project's scope was narrowed to focus specifically only on inconsistencies between tables and text.

The final implementation of the adopted solution is comprised of a multi-phase pipeline, only partially automated due to the documents' format constraints. The system consists of an initial phase, that requires a hybrid approach, combining automated OCR processing with manual intervention, a second phase that employs

handcrafted rules and semantic embeddings in order to identify relevant text for each table and a final phase that uses a prompt chain in order to classify text and table pairs as either contradictory or non contradictory while also providing an explanation for each choice.

In order to address the limited availability of contradictory examples in the dataset provided, a synthetic data generation process was developed. This process uses the real data points (table and text pairs), assumed to be non contradictory, as a basis for creating modified versions of the textual data with the aim of generating text that is in contradiction with the table it was originally associated to.

Finally, the synthetic and real data points have been used in order to test the classification stage of the pipeline. The results of the experiments are not easily interpretable, but they are, however, at least in part encouraging. The results also suggest that simply increasing the amount of text present for each data point, such as is the case with this project, greatly increases the difficulty of the resulting problem when compared with similar datasets comprised of pairs of sentences and tables.

The implemented solution represents a first prototype for addressing the challenge of automated inconsistency detection in clinical trial documents, with potential for future expansion and refinement.

ACKNOWLEDGMENTS

We thank Fondazione Italiana Linfomi - ETS (Italian Lymphoma Foundation) for providing us the documents necessary, whose support was essential for the development of this project.

Table of Contents

List of Tables

List of Figures

Acronyms

NLP
: Natural Language Processing

NLI
: Natural Language Inference

LLM
: Large Language Model

CoT prompting
: Chain-of-Thought prompting

OCR
: Optical character recognition

Chapter 1

Introduction

In the domain of medicine, in order to be able to conduct a clinical trial, several steps need to be cleared. Among them, a document detailing the schedule, requirements and goals of the trial needs to be redacted and approved by an ethics commission.

This process is usually pretty slow, requiring months at a time. Among the requirements for approval, there is that about the document being formally correct, meaning that inconsistencies, if found in the text, need to be corrected and the document needs to be re-submitted to the commission. If this happens, the process is lengthened by additional months. Another important requirement is that these documents adhere to both European and specific standards related to the sub-field the trial belongs to.

The goal of this work is that of trying to create an automatic system capable of finding said inconsistencies in real time or near-real time, so as to allow for the correction to happen before submission, possibly avoiding months of potentially wasted time. In particular, this work represents a first prototype that might be later further expanded upon.

1.1 Limitations

The first limitation with regards to this work was that of the availability of a dataset. The only material obtainable was a set of 12 PDF files, 5 of which contained mistakes for witch they were rejected, and a list of the objections a given commission levied at one of these 5 documents, objections regarding, among others, the type of inconsistencies that were the target of this project. With regards to the PDF files themselves, they were not machine readable source files, so OCR was required in order to process them. Due to these restrictions, accurate testing of the results of this work could not be achieved, and only secondary methods for trying to evaluate the results could be employed. Moreover, fully automating

the entire contradictions discovery process proved too challenging because of the documents' format and structure, so the first part has remained partially manual but could become automatic given the right document formats and a certain level of standardization.

The second limitation concerns the availability of computational resources. Since the conception of this project, it was clear that LLM models would be involved in solving at least some, if not most, of the tasks and challenges deriving from the goal of this work. In particular, from early experiments and also from searching through the literature it seemed likely that the only models capable of performing at a satisfiable level would be ones with at least 70 billion parameters, which require specialized hardware in order to be run. It was, however, deemed unlikely that this project would be able to receive the kind of hardware required for such experiments, so it has been completed using only personal hardware and free resources available online. The main consequences of this limitation concern the size of the models that could be employed, with its direct correlation to performance, heavy restrictions on the usage of the bigger models and very limited control over the medium sized models (to some extent even over the smaller ones). With regards to medium sized models in particular, one crucial parameter that was not changeable was the temperature of a model. This means that said parameter could not be set to 0, meaning that the results of the various experiments could not be ever fully replicated.

The third limitation pertains to the highly specific field of medical literature, with respect to which the clinical trial proposal documents represent an even smaller sub-field. Medical domain documents have a specialized language and structure, and NLP in this field tends to require ad-hoc datasets and models in order to be successful. These datasets and models are, however, often not released, or require special access.

1.2 Initial goal and restrictions

The initial goal of the work was to try to create a general system for finding inconsistencies/contradictions in the clinical trial proposal documents. Stated as such, the problem was very similar to an expanded version of NLI, Natural Language Inference. Some intermediate clustering steps would have been required in order to group together pieces of text (and tables) and then the generality of LLM models could have been harnessed in order to find contradictions among the textual fragments that have been clustered together, while also avoiding the risk of exceeding the LLM's context window.

To be more precise, the initial envisioned pipeline would have contained:

- a text extraction and tokenization phase

- an embedding phase, using a model akin to SBERT[1], but specialized for the medical field
- a dimensionality reduction and clustering phase
- a classification phase, using specialized LLM models and comparing them with NLI fine tuned BERT models, again specialized for the medical field

This initial idea has proven, however, to not be practical or adherent to the types of errors that were usually found in these documents.

1.2.1 Dataset concerns

As stated in the previous section, the dataset provided contains few data points, most of which are assumed to not contain any errors. This constraint does not allow for statistically relevant testing, at least not by solely employing the dataset itself. To our knowledge, there is also no available open dataset that resembles the type of documents this thesis is concerned with.

For this reason, generating a synthetic dataset in order to test our resulting pipeline has been the only practical option available. This method, however, can only be applied to the classification step, since it is unclear how something similar could have been done in order to test the validity of the embedding and clustering phases. The embedding and something similar to the clustering phase has been, nonetheless, maintained in the reduced final version of the pipeline, but its role is less crucial and its purpose is that of a redundancy measure.

1.2.2 Error types

After evaluating all the documents at our disposal and the various objections written by the commissions, we discarded the types of objections that were not related to the correctness of the document itself or for which we could not foresee an automatic way of discovering the related issues before the document was submitted. Afterwards, we have decided to categorize the remaining issues. It has to be noted again that the sample size was too small in order to be able to asses trends, but, barring few exceptions, most of the problems seemed to regard two types of errors:

- inconsistencies between a table and text related said table
- missing text according to a given standard and/or guideline

Those two types of problems require two very different types of solutions in order to be solved. The second one, in particular, also relies on a set of standards and protocols that are not explicitly stated or compiled in a unified corpus of text.

Given the nature of these prevalent problems, it was deemed as better to try and solve them one at a time, creating an ensemble system, with different parts focused on finding different types of issues regarding the document, rather then trying to create a single monolithical pipeline.

1.3 Proposed pipeline

Having concluded the initial analysis of the literature and available material, the practical scope of this thesis was narrowed to that of trying to find the first type of problems present in some of the clinical trial proposal documents, that being contradictions between a given table and the related text.

1.3.1 First experiments and prototype

In order to test whether the solution we envisioned could work in practice, a first simplified prototype of the pipeline was created. The main idea was that of using an OCR python package (pdfplumber) in order to extract all the tables and text and then using this as a basis for creating a very small handcrafted dataset containing 5 tables, for each of which unrelated and entailed sentences were selected. 2 of the 5 tables contained the examples of genuine contradictions, between table and text, found in our dataset, and thus for these 2 tables also contradictory sentences were included. All the sentences were also appropriately labeled.

In order to obtain a more balanced initial dataset, a synthetic data point generation process was devised so as to create more contradictory examples. This process involved using a separate NLI dataset[2] in order to draw contradictory examples to be included in a three-shot-learning prompt aimed at using an LLM in order to create contradictory statements given one of the selected sentences related to one of the tables and in accordance with it. This process has been included, with slight modifications, in the final pipeline as well, so it will be discussed in more detail in the appropriate chapter.

Finally, the small dataset was tested, employing both BERT derived NLI models, as suggested by the work of Mubashara Akhta et al.[3], and zero shot prompting of LMM models.

1.3.2 Complete pipeline and paradigm changes

After some encouraging first results, it was decided to proceed with the creation of a full pipeline, which would start with data extraction from the source files and end with the classification of potential contradictions. The following is a brief overview, while the full details are present in the dedicated chapter.

The first part of the pipeline, concerning the tables and text extraction, could not be fully automated, due to the files' format (PDF) and due to the nature of some of the tables contained in the documents. As such, the process was in part automatic, with manual intervention required in order to properly reconcile tables spanning multiple pages, to correct OCR mistakes and to classify tables used only for formatting purposes as such, considering they are not useful to the purposes of this work.

After the extraction of text and tables, the second part of the pipeline is concerned with the selection of the appropriate text for a given table. After experimenting with different alternatives, what proved to work best was the selection of entire pages of text deemed to be related to a given table. This was in part done by simply selecting the pages which the table was embedded in and also the adjacent pages, and in part by employing SBERT-like models in order to find pages, similar in content to the table's content, in the rest of the document, as mentioned in a previous section.

Finally, having obtained a set composed of the table and several (supposed to be) related pages, a multi-phase prompt chain was employed in order to decide whether any contradictions were present each table-page pair, given that simple zero-shot learning proved unsuccessful with these more complex types of inputs. This prompt chain consists of an initial "chunking phase", in which the LLM is instructed to divide the table into chunks and to assign related parts of the text to each chunk, if present, a subsequent "argumentation phase", in which the LLM is instructed to generate an argument for why a given table chunk and its assigned text are in contradiction with each other and then it is instructed to create the opposite argument, and finally a "decision phase", in which the LLM is prompted with a table chunk, the text assigned to it and the two arguments derived in the previous steps and it is instructed to decide if a contradiction is present or not.

A quick mention has to be made with regards to the generation of synthetic data points, required also in this second iteration of the pipeline. As stated in the previous subsection, the same basic idea of prompting an LLM, using an auxiliary NLI dataset in order to create examples, so as to generate contradictory statements, given premises extracted from the documents and related to one of the extracted tables, was maintained. Considering, however, that the starting dataset for this prompt chain is comprised of data points that each contain a full page of text, whose content might only in part be related to the assigned table, or could be entirely unrelated, an adjustment was required in order to be able to actually generate synthetic pages which were contradictory w.r.t. to the tables' content. Said adjustment pertains to the problem of extracting only the relevant text and modifying said text. The LLM is also instructed to output the entire page with only the relevant chunk being modified, in order to have the same type of synthetic data points as the original ones.

Chapter 2

Related works

The problem we attempted to tackle with this thesis is in part adjacent to NLI (Natural Language Inference). The main difference regards the broader scope of this project, which encompasses entire documents at a time, and not just pairs of sentences. This entails a greater amount of text, which requires preprocessing techniques, and it also entails different formats, such as tables.

Another key difference, as also noted by Sharon Jiang [4], is the fact that NLI is concerned with a directional entailment problem (if the hypothesis is entailed by the premise) while our goal is that of detecting bidirectional contradictions. Closely related is also that NLI is a three way classification (entailment, neutral, contradiction) whereas the classification goal we are concerned with is binary (contradiction or non contradiction).

2.1 General problem

Concerning the general problem of contradiction detection, the most similar work to ours, in terms of scope and goals, to our knowledge, is the work of Sharon Jiang [4]. In the related paper, he describes a possible pipeline for detecting contradictions in clinical notes. The proposed method consists of the following stages:

- A textual conversion stage and tokenization. In this stage, structured text formats, such as tables, are transformed into sentences. This is done via a series of handcrafted rules and regular expressions, when possible, and following heuristic strategies when it is not.
- A sentence pairs formation stage. In this stage, the scispaCy [5] tool is used in order to perform named entity extraction. Once entities are extracted, biomedical ontologies are used in order to group said entities, using common broader categories, and thus sentences with entities belonging to the same

group are also grouped together. Finally, within the same group a one-hot encoding of the sentences is performed, based on the medical concepts identified within each sentence, and cosine similarity is used in order to form the final sentence pairs.

- A classification stage. In this stage, a machine learning based algorithm is employed in order to determine whether a pair of sentences is contradictory or not. The particular algorithm selected for this work was a mixture of handcrafted rules and an ensemble of decision trees model.

Another interesting aspect is how the experimental results were obtained. The entire pipeline was not experimentally tested as a whole (no such results are mentioned), but only the final classification stage was tested. With regards to this final stage, a small dataset of hand-annotated pairs of sentences was extracted from the MIMIC-III database [6] and the MedNLI dataset [7], which is an NLI database, in the medical domain, whose premises are sampled from MIMIC-III, while the hypotheses are handcrafted by experts. Both the code and the small created dataset appear to not have been released.

Another very different type of approach present in the literature is based on the SemRep tool [8] and the SemMedDB database [9]. SemRep is a rule-based system that is used for extracting subject-predicate-object relation triplets from sentences in the medical domain. Using this tool on the entire set of PubMed citations, the SemMedDB database was created.

The approaches based on these resources employ handcrafted rules based on the aforementioned triplets in order to form pairs of contradictory sentences, such as the work of Graciela Rosemblat et al. [10] and the work of Prajwol Lamichhane et al. [11]. In these works and similar, there is a first tokenization phase, using SemRep, and then a second phase that fuses together the formation of pairs and their classification.

Outside of the medical domain, the work of Cheng Hsu et al. can be mentioned in relation to the creation of a pipeline for classifying a given wikipedia article as contradictory or non contradictory [12]. According to their definition "if an article possesses at least two statements that contradict one another, we can say that this article contradicts itself, i.e., is self-contradiction". In order to obtain their desired result, the authors limit the search for contradictions to only within each individual paragraph, removing the necessity for a clustering phase, but also potentially missing contradictions between sentences present in two different paragraphs. Their proposed pipeline consists of the following:

- Sentence tokenization

- SBERT embeddings of each sentence
- Formation of all possible sentence pairs within each paragraph and fusion of the embeddings for each pair
- Assignment of the probability of contradiction for each pair, using a feed forward neural network and the fused embeddings
- Selection of the top-k most likely contradictory pairs
- Combination of the top-k embeddings and a second feed forward neural network for classifying the article as potentially contradictory or not contradictory

Concerning the dataset employed, due to the way Wikipedia functions, the authors were able to obtain the versions of articles annotated as contradictory by the Wikipedia editors, and that have subsequently been corrected. These annotations also allow the precise identification of the problematic part of an article.

2.2 Table contradictions

Focusing on the prevailing task of this thesis, which is that of finding contradictions between table and related text, there are two broad categories related to our work: synthetic data generation and table-based fact verification.

With regards to synthetic data generation, Mohammad Javad Hosseini et al. propose a method for data augmentation for the task of NLI, via the creation of a general synthetic NLI dataset [13]. They also show improvements in NLI benchmarks w.r.t. just training on regular datasets. The biggest improvements were obtained when fine tuning smaller size LLM model (T5-small) on this dataset.

The main steps involved a chain of LLM tasks tuned to generate the dataset:

- First they fine tune an LLM in order to generate domain names (and also the length of the text to be generated and a sample of in-domain text).
- After sampling some of the domains (and text span lengths generated) they use prompting in order to generate the premises.
- Finally, they prompt-tune an LLM (FLAN-PaLM 540B) in order to generate the hypothesis, given the generated premises and the labels (entailment, neutral, contradiction).

The authors also note that a Large LM is necessary in order to achieve good results w.r.t. the generation process. They also point out that just prompting the model did not seem to yield good results.

Tu Vu et al. had the goal of using data augmentation in the context of tasks with very few domain annotated examples (and many non-annotated ones) [14]. It must be emphasized that the tasks they were focusing on were not NLI.

Their proposed method is that of first generating in-domain labeled data for an intermediate "general" task (NLI was chosen as the intermediate task), using this data to fine-tune the model, and adapting it to the domain. Afterwards, they further fine tune the model on the labeled in-domain available data. Finally, the model is used to annotate all the available not yet annotated samples, creating pseudo labels. The newly annotated set is used to further fine-tune the model, and then the cycle of pseudo-annotations and fine-tuning is continued until a convergence criterion is reached.

The main point of interest is the NLI data generation method employed. The authors fine-tuned a T5-3B model on a big NLI dataset, with the goal of hypothesis generation, given a premise and a label. At generation time, they employed an "overgeneration and filtering" approach, which means that for each label and premise they generated several (up to 100) hypotheses and then they used an NLI-fine-tuned BERT model in order to filter out the samples whose label did not match the one assigned by the BERT model.

Note that the "overgeneration and filtering" approach can be employed when the NLI task is an intermediate task, but it is unlikely to work in the context of synthetic data generation for the NLI task itself, unless paired with knowledge distillation, employing a bigger model for the filtering process and using the dataset in order to fine tune a smaller model.

The aforementioned table-based fact verification task is in principle a similar task to NLI: given a table and a sentence, the goal is that of identifying whether the claim expressed by the sentence is supported by the information present in the table, it contradicts said information or if there isn't enough information present to either support or reject the claim.

In the specific domain of medical table-based fact verification, Mubashara Akhta et al. focused on the creation of a table-based dataset for evidence-based fact checking [3]. Aside from the 3 categories mentioned above ("supports", "refutes" and "related but not enough information"), during the data annotation process a forth category was introduced, that of "unrelated" claims, which have then been removed from the dataset. The authors also "heuristically removed all tables that were used purely for formatting reasons", which is a problem we have encountered in our dataset creation as well.

Finally, after creating the dataset, they used it in order to test several BERT derived models, pre-trained on NLI task. They compared various models and also various table serialization techniques, including simple text concatenation, TAPAS

(table-aware positional embeddings in BERT models), and descriptions of the tables generated using a T5 LM model.

Zihui Gu et al.'s work focuses on creating an end-to-end system for fact verification, given a claim and a dataset of tables [15]. There are two main points of interest regarding their work:

- they retrieve the tables relevant to a claim firstly by employing classic keywords based techniques, using the nltk library, in order to filter most of the irrelevant tables and they secondly use a semantic based approach (TAPAS) on the remaining tables.
- they deal with the context length limit by trying to sample only the most informative columns of the table, given a particular claim.

Note that, given our goals and the type of data we were dealing with, the task we focused on had the reversed requirements, that is to say that given a table and an entire document, we needed to retrieve the text that was most informative w.r.t. the table.

Moving to field of prompts and prompt engineering, a good introductory point for both a review of the entire field and for a classification and standardization of concepts and nomenclature is the work of Sander Schulhoff et al. [16]. Their work is also a great compilation of the state of the art techniques regarding prompt engineering. Two of the techniques reported are the ones based on the works of Takeshi Kojima et al. [17] and Xuezhi Wang et al. [18].

In their paper, Takeshi Kojima et al. propose a new method for improving the performance of LLM models by adapting the chain-of-thought paradigm to the context of zero-shot learning. Chain-of-thought (CoT) is normally based on the addition of examples (few-shot learning) that also include the reasoning steps required for obtaining the right answer given the initial question or statement. This is done in order to bias the model towards the production of intermediate steps, which has been shown to increase performance [19]. Based on the same logic, Takeshi Kojima et al. propose the addition of the sentence "Let's think step by step." at the end of the prompt in order to simulate the same effect as CoT, showing improvements in performance.

Xuezhi Wang et al. also propose a method for bettering the performance of an LLM in reasoning tasks. They combine chain-of-thought prompting with a decoding strategy that they call "self-consistency". The chain-of-thought prompting is used

in order to guide the model into expressing intermediate reasoning steps before giving an answer to a question. However, instead of accepting the first answer that the LMM provides ("naive greedy decoding"), they instead sample multiple answers, given the same prompt, so as to "to generate a diverse set of reasoning paths". Finally, they choose an answer using various methods, which they subsequently compare: some require knowing the logits associated with the generated tokens (weighted options) while others only require knowing the generated answers, like choosing the majority answer.

Regarding our work, this strategy would have proven interesting to explore in its full capacity, however it was not possible to pursue given the computational limitations, which constrained the level of control we had on the bigger LLM models employed, allowing for only the simplest form (multiple sampling and majority voting) to be explored. However, the generation of a pair of arguments, for and against the presence of a contradiction, can be seen as a type of alternative to this work, that requires less resources and still follows the same main concept of exploring "a diverse set of reasoning paths" while using an LLM in order to answer a given question.

Chapter 3

Problem statement and pipeline definition

This chapter focuses on the technical details of the project. It firstly precisely defines the problem statement, the inputs and the desired outputs and then describes in detail the pipeline devised in order to try to achieve the stated goals. A graphical overview of the pipeline can be observed in figure 3.1. It is comprised of a partially manual data processing stage, an intermediate stage in which tables and textual chunks are associated, forming data points each comprised of a table and a page of text, and, finally, a classification stage for each data point. The core of the classification stage is a prompting chain which employs a "divide and conquer" strategy, dividing each data point into smaller chunks, and different prompting strategies in order to classify each chunk. What is presented in the figure is one particular classification strategy, but other were tested. An aside is represented by the synthetic data generation process, which was required given the scarcity of the contradictory data points.

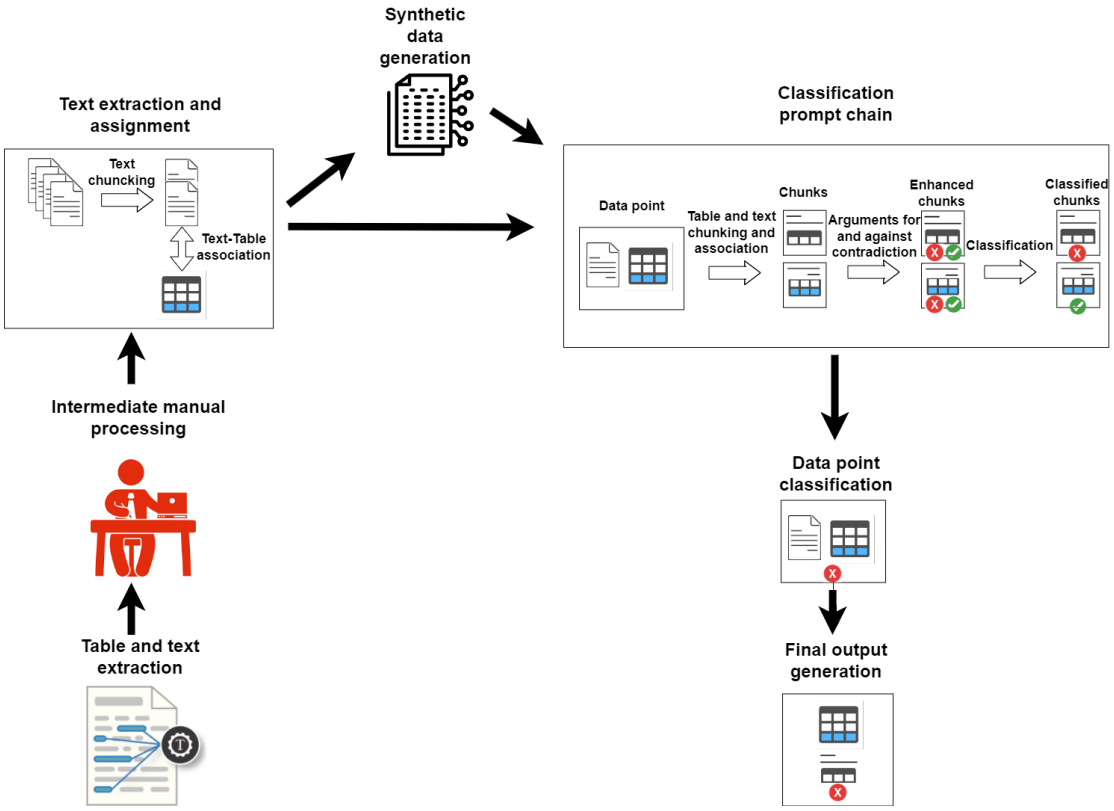

Figure 3.1: A general graphical overview of the pipeline. Firstly, text and tables are separated and extracted from the PDF documents via OCR. Then, the tables are manually processed in order to remove all the inconsistencies introduced by the OCR. Afterward, the pages of text can be associated with the manually elaborated tables via a mixture of handcrafted rules and semantic embeddings. Each page-table pair is treated as a data point and these data points are used as a basis for generating synthetic data, in order to add more contradictory examples. Finally, the synthetic and real data points are used as input to a classification prompt chain, which divides each data point into smaller chunks and classifies each individual chunk. If, given a data point, even one of its chunks is classified as contradictory, the entire chunk is classified as contradictory and the identified contradictions are included in the output of the pipeline.

3.1 Problem statement

Given as inputs a collection of PDF files, assumed to all be clinical trial proposal documents, the goal of this project is to create a pipeline that outputs potential contradictions present between the tables present in these documents and text related to these tables. In particular, the desired output should specify the table, the

contradictory text snippet and an explanation for why said text snippet might be in contradiction with the table. The pipeline is supposed to be fully automatic, but, due to the constraints imposed by the source documents, the initial preprocessing steps require manual intervention, but this process could be rendered fully automatic in the future.

The specific use case for this type of technology is that of aiding the researchers/clinicians in finding potential problems present in their proposal documents, but it is not intended as an automatic system to be used without supervision. The researcher/clinician should decide themselves whether a contradiction is present or not, aided by the explanation provided by the software. In light of this, it is worth pointing out that having high recall w.r.t. the contradictions would be much more preferable then having high precision. Having too low precision, however, would render the software hard to use because it would require too many false positives to be manually inspected and rejected.

Aside from the practical use cases of the software, another important goal is that of trying to assess the goodness of the pipeline itself, both as a whole and of the different parts in isolation. This has proven to be greatly challenging, and thus only partial results could be obtained. With regard to this aspect, as stated above, the recall of contradictions should be a key metric in evaluating the performance of the entire pipeline, but the very small set of contradictory data points made only indirect methods of recall estimation possible.

In this chapter, the details of the pipeline will be explored, whereas in the next chapter concerns the results of the partial experiments aimed at evaluating the performance of the software.

3.2 Tables extraction

Given a set of PDF files as input, this section of the pipeline should output, for each document: the document's pages without any table contents, the tables themselves and the span of pages occupied by each table. This is, however, not possible in any trivial way, given the input files.

The biggest hurdle with regards to automation are the PDF files themselves and their format. Said format does not appear to maintain any of the structural elements of the documents, which renders parsing the documents only possible through OCR. A corollary is that mistakes in interpreting characters and structures, although not common, still occur.

This first issue is compounded with the prevalence of multi-page tables. These tables are not be recognized as single entities, because of OCR, and given the nature of the multi-page tables, there are no trivial heuristics that can be applied in order to recognize that two different instances, extracted using OCR, are, in fact,

two parts of the same table. In most of the cases present, and understanding of the meaning of the tables themselves would be required in order to recognize two extracted parts as belonging to the same table. Headers also tend to be repeated in some of the multi-page tables present, which also poses a challenge when trying to recognize the different parts as not being separate individual tables.

	Screening (All)	Induction							Maintenance/Follow-up		Extended FU
	D-28 to D1 as per clinical practice	Cy1	Cy1	Cy 2-4	After Cy 4	Cy 5	Cy 6	EOI	Every 6 months assessment	EOT*	as per local clinical practice
		D1	D10±2	D1		D1	D1				
Informed consent	X										
Inclusion/Exclusion	X										
Histological diagnosis *(made within 3 months before study entering)*	X										
Complete Medical History	X										
Physical exam [a]	X	X	X [b]	X	X	X	X	X	X *(D1 All Cy)*	X	
ECOG PS	X	X	X [b]	X	X	X	X	X	X *(D1 All Cy)*	X	
B symptoms	X										
Hematology tests [c]	X	X	X [b]	X	X	X	X	X	X *(D1 All Cy)*	X	
Blood chemistry [d]	X	X	X [b]	X	X	X	X	X	X *(D1 All Cy)*	X	
Serum IgM, IgG, IgA	X				X			X		X	
Coagulation (PT, PTT)	X										
Viral markers [e]	X										
Bone marrow biopsy [f]	X							X		X	
Tumor assessment (TB CT scan, MRI, PET) [g]	X				X			X	X	X	
Response assessment [h]					X			X	X	X	
12-lead ECG	X							X		X [i]	

	Screening (All)	Induction							Maintenance/Follow-up		Extended FU
	D-28 to D1 as per clinical practice	Cy1	Cy1	Cy 2-4	After Cy 4	Cy 5	Cy 6	EOI	Every 6 months assessment	EOT*	as per local clinical practice
		D1	D10±2	D1		D1	D1				
LVEF echo [i]	X							X		X	
Urine pregnancy test [j]	X										
Other procedures [k]	X				X			X		X	
PROs (FACT-Lym) [l]	X				X			X	*month +12 and +24 only*		
Peripheral blood samples in BCT tubes (MRD; cfDNA in plasma)	X			*D1 Cy3 only*	X			X	*month +12, +24 and at relapse/progression*		
Bone marrow aspirate (MRD)	X				X			X	*month +12, +24 and at relapse/progression*		
Concomitant medications [m]	Recorded from screening through 30 days after the last dose of treatment.										
Adverse event reporting [n]	Recorded from first dose of treatment through 30 days after the last dose of treatment.										
Serious adverse event [n]	Recorded from signing of the informed consent form through LPLV.										

Figure 3.2: Example of a table spanning two different pages: both parts have their own headers and there is also text in-between the two. It is also an example of a complex table structure, although not the most extreme.

Another complication concerns the complexity of table structures employed. A simple table structure would entail the same number of columns for each rows. This is directly machine translatable even for OCR tools, which encode the recognized structure row by row, in a 2D array format, whereas the more extreme the deviation from this simple format the harder it is to encapsulate the nuance of the original table using said encoding strategy. Having a mark-up format would alleviate also this problem, allowing for access to a more expressive description of the table itself.

The complexity is also an issue that impacts the accuracy of the OCR itself: more complex tables proved to more frequently contain structure recognition errors w.r.t. more conventional formats. An exemplary type of problematic table structure is embodied by the tables nested within other tables, which in the scope of this project are treated as tables containing only a textual version of the nested table, without the structure.

Tab 2: Guidelines for management of specific adverse events due to DHAP:

<table>
<tr><td colspan="4">Hematologic Toxicity</td></tr>
<tr><td>Blood counts</td><td colspan="3">Cytarabine and cisplatin dose</td></tr>
<tr><td>ANC < 1,000/μL and/or platelets < 75,000/μL</td><td colspan="3">Delay 1 week*.
After 1 week (day21-22), if ANC > 1000 /μL and/or platelets > 75,000/μL, may restart at same dose with G-CSF** and platelet support
OR
if ANC < 1000/μL and/or platelets < 75000/μL, check counts q 3-4 days. When both ANC ≥ 1000 /μL and platelets ≥ 75,000/μL, restart at same dose and with GCSF** and platelet support</td></tr>
<tr><td>ANC < 1,000/μL and/or platelets ≥ 75,000/μL</td><td colspan="3">Delay 1 week*. Restart at the same dose if ANC ≥ 1000/μL.</td></tr>
<tr><td>ANC ≥ 1,000/μL and/or platelets < 75,000/μL</td><td colspan="3">Delay 1 week*. Restart at the same dose if platelets ≥ 75,000/μL.
If platelets 50,000/μL to <75,000/μL after 1 week, may restart at the same dose with platelet support as necessary.
If after one week the platelets count is still not permissive, discontinue treatment</td></tr>
<tr><td colspan="4">*If counts presumed to be low due to marrow involvement, treat after 1-week delay (e.g. at 3 weeks or Day 21) despite counts.
** G-CSF should be given prophylactically for all future cycles.</td></tr>
<tr><td colspan="4">Non-Hematologic Toxicity:</td></tr>
<tr><td>Worst Toxicity in Previous Cycle†</td><td>Cisplatin (% previous dose)* for this cycle</td><td>Cytarabine (% previous dose) for this cycle*</td><td>Dexamethasone</td></tr>
<tr><td>Grade 3 related non-hematologic toxicity, except nausea, vomiting, alopecia</td><td>Hold*, then 75%</td><td>Hold*, then 75%</td><td>No change</td></tr>
<tr><td>Creatinine 1.5 to 3 x ULN</td><td>Hold*, then 75%</td><td>No change</td><td>No change</td></tr>
<tr><td>Grade 3 or 4 neurotoxicity/ototoxicity (cerebellar/ gait dysfunction related to cytarabine; sensory/peripheral)</td><td>Discontinue</td><td>Discontinue</td><td>Discontinue</td></tr>
<tr><td>Other grade 4 non-hematologic/ organ toxicity or creatinine > 3 x ULN</td><td colspan="3">Hold for one week, hydrate. Discontinue if it does not resolve to ≤ grade 2.</td></tr>
<tr><td>Hemolysis, optic neuritis, arterial thromboembolism, severe hypersensitivity reactions</td><td>Discontinue</td><td>Discontinue</td><td>Discontinue</td></tr>
<tr><td>Hepatic Impairment</td><td colspan="3">No dosage adjustment required for cisplatin. Cytarabine dose should be reduced with impaired hepatic function; no details available.</td></tr>
<tr><td>Renal Impairment</td><td colspan="3">For cisplatin, renal function should have normalized before patients are retreated. If continued treatment is considered to be mandatory, the following dose modifications could be considered at the physician's discretion [53].
<table><tr><th>Creatinine clearance</th><th>Cisplatin (% previous dose)</th></tr><tr><td>46-60</td><td>75%</td></tr><tr><td>30-45</td><td>50%</td></tr><tr><td><30</td><td>Discontinue</td></tr></table>
For cytarabine, no adjustment required for standard doses.</td></tr>
</table>

Figure 3.3: Example of a table with a complex structure and a nested table contained within itself.

3.2.1 Automatic extraction

Many python libraries for parsing PDF files were tested[1], but only pdfplumber proved to be reliable enough to be used for the purposes of this project. This tool can be used for the detection of text and tables, via OCR, within a given page. It is possible to extract only the tables without the surrounding text, however the opposite is not possible natively, and it required the integration of custom behaviour, as suggested in the project's GitHub issues page[2]. Aside from text and tables, the software is also capable of exporting a page as an image.

Given the software's capabilities, the automatic part of the extraction process is as follows:

- for each document, for each page, if the page contains one or more tables, the object described in table 3.1 is created. The reason why the "tables" field contains also the tables present in the next page is in order to ease the manual process of joining together the parts of same table present in multiple places. This object is assigned a name and then is added to the list of tables present in the document. Finally the list is sorted by page number.
- for each document, each page containing a table, and the previous and next pages, are converted to a PNG image and compiled together, ordered by page number. This is done so as to allow for an easier visual inspection of the tables and of the context surrounding them.

Field	Type	Description
pages	Array	The pages spanned by the tables. In this phase it can only contain one page.
tables	Array	A list containing all the tables (2D arrays) present in the current page and in the next page, if present.

Table 3.1: The JSON object extracted for each page containing a table in the first step of the pipeline.

One thing to note is that, at this stage of the pipeline, the text is not yet extracted, as it is the target of the next pipeline step.

3.2.2 Manual elaboration

For each document, the previous automatic process yields JSON file, which is a list of objects (at most one per page) containing the tables, and a PNG with all the relevant pages. The manual processing consists in changing the original JSON objects by:

[1] pdfplumber, PyPDF2, aspose-pdf, tabula-py

[2] https://github.com/jsvine/pdfplumber/issues/242

- identifying the number of tables present in the page
- identifying if a table continued on the next pages and, if it did, modifying the structure of the table contained within the JSON object by fusing all of the parts together accordingly
- classifying all the tables as either "fake", meaning that the table was used only for formatting purposes, or "not fake"
- correcting, when possible, OCR errors in both the structure and the text of the tables
- modifying the "pages" field in accordance with the number and location of the tables that have been identified

For each JSON object present in the list of tables of each document, the resulting object has the structure described in table 3.2. Any future modifications to this step of pipeline, in order to render it fully automatic, would have to yield the same type of structure.

Field	Type	Description
pages	Array	The pages spanned by the tables. In particular, for each table it contains an array of all the pages spanned by that table.
is_fake_table	Array	An array of boolean values which are, for each table, true if the table is used for formatting purposes, false otherwise.
tables	Array	A list containing the tables objects, which are arrays of arrays.

Table 3.2: The JSON object resulting from the manual elaboration. This is the final product of the tables' extraction pipeline step.

3.2.3 Suggestions for future automation

As stated before, the main reason why it proved difficult to fully automate this step was due to the files' format and to the complex nature of the tables employed in these types of documents.

With regards to the first issue, the preservation and utilisation of the source documents' formats would solve most of the problems that required manual intervention. It also bears notice, however, that the problem of complex tables is not to be underestimated. The lack of standardization, when compared with other fields, means that only the more general types of LLM models are able to perform well in tasks related to the type of tables present in these documents. Moreover, this level of complexity also greatly reduces the amount of serialization options available that can fully express the original content. The complexity also makes the availability of a source files even more of a requirement, since these types of tables tend to encounter many problems when relying on OCR systems as well.

Finally, another type of complex structure that bears mentioning is the case of multiple connected tables, whose meaning is defined in relations with the others, like in Example 3.4. In this case, in order to have a full understanding of the structure of the table, all the parts need to be available as well as the text surrounding it. This type of complex structures could often be reduced to a single table with more fields, which would greatly aid in automating the contradictions detection process. This is not a problem of the table extraction process per se, but since the structures are considered separately, it is likely to affect the performance of the overall pipeline.

Cycles 5-6		**Q28 days**
Rituximab	375 mg/m², i.v. OR 1400 mg flat dose s.c. *	day 1
Bendamustine	90 mg/m², i.v.	day 1, 2 **
Cycles 7-8		**Q28 days**
Rituximab	375 mg/m², i.v. OR 1400 mg flat dose s.c. *	day 1

Figure 3.4: Example of two connected tables that could be fused into one. Also, the text above each one could be part of the tables themselves.

3.3 Text extraction and assignment

Given as inputs the PDF files and the manually processed JSON files, this phase of the pipeline is concerned with assigning text to tables, that is to say that the objective is that of assigning to each table all the text related to it and present in the document containing said table, while refraining from assigning to it text that is unrelated.

3.3.1 Chunking

In order to be able to assign the text to tables, first a strategy for dividing the text into smaller pieces must be devised. The initial intent was that of dividing all the text into individual sentences, so as to have a final output resembling that of the table-based fact verification task. This, however, proved to create too many artifacts upon inspection of the intermediate results. By also testing the pipeline as a whole, using only the real contradiction cases, the same conclusion was reached. The problems with this method of dividing text were mainly two:

- the tokenization process that divides the text into sentences proved to be too imprecise. This was due both to the necessary simplicity of the models and to the complexity of the structures present in the text. For the experiments, the spaCy[3] python library was used.
- often single sentences are not enough to provide sufficient context to the meaning of the words that they contain. The entire paragraph is required in order to contextualize the meaning of the sentences enough as to determine whether they are related to a given table or not and whether they might be in contradiction (or not) with a said table.

The ideal way of overcoming these limitations would be that of dividing the text into paragraphs, but, with consideration for avoiding over-engineering and also wanting to avoid similar issues as with the process of sentence tokenization, we chose to divide the text into the individual pages.

It must be noted that the bigger the chunks, the more likely it is that text neutral w.r.t to the table it is assigned to will be present alongside text that is indeed related. This problem has been addressed in the next step of the pipeline, involving LLM models.

3.3.2 Simple assignment criteria

Given a table, the text that is contained in the page/pages in which the table is embedded, or in the previous/next page, is the most likely to be related to said table. Following this observation, the first heuristic method for assigning pages to a table is that of assigning to it the text present in the table's page/pages and the previous and next pages. This method ensures that no local potential contradictions are missed. Global contradictions might, however, still be missed. A table might be referenced several pages after it is presented or something related to the same argument of the table, and potentially contradictory, might be present in another section or chapter. For this reason, alongside the heuristic method, a second more global approach for text assignment is present, based on the semantic similarities conveyed by SBERT-like embeddings.

3.3.3 Embeddings

This second, more global approach, entails, for each document, the embedding of text pages and tables and the assignment of the pages to tables that are above a previously selected cosine similarity threshold [4].

[3]Both the "en_core_web_sm" and "en_core_sci_scibert" spaCy models were tested

[4]If a page has already been selected in previous step, it will not be selected again

Before embedding the tables, they are converted from their original 2D array form to Markdown, using the Pandas python library. This is in light of the considerations of Dehai Min et al., who have found that Markdown shows "unexpected effectiveness" in the context of RAG's (Retrieval augmented generation) retrieval phase, which is very similar to the task performed at this stage of our pipeline.[20]

With regards to the embedding models, three SBERT models, specialized for medical text, were tested[5], and the final pipeline employs the PubMedBERT model. The selection criteria was based both on qualitative observations of the resulting selected text for the three models and on a heuristical quantitative approach. The latter was the only available quantitative approach that we could devise, given of a lack of ground truth available for performance comparisons of this segment of the pipeline; an ablation study was also not possible, given that the sample size of the contradictory data points was too small to be statistically significant. This heuristical approach is based on the same observation as above, that is to say that we expect the text in the pages containing and surrounding a table to be highly correlated with the text present in the table, and we expect this similarity to decay relatively fast the further a page is from the table's page. Given this consideration, we expected that a good embedding model would yield a similarity distribution akin to a Gaussian distribution, centered on the page of the table (or on the central page of the pages that contain the table, if it is a multi-page table). An example of this sort of distribution can be seen in Figure 3.5.

[5]"kamalkraj/BioSimCSE-BioLinkBERT-BASE", "neuml/pubmedbert-base-embeddings" and "sentence-transformers/allenai-specter"

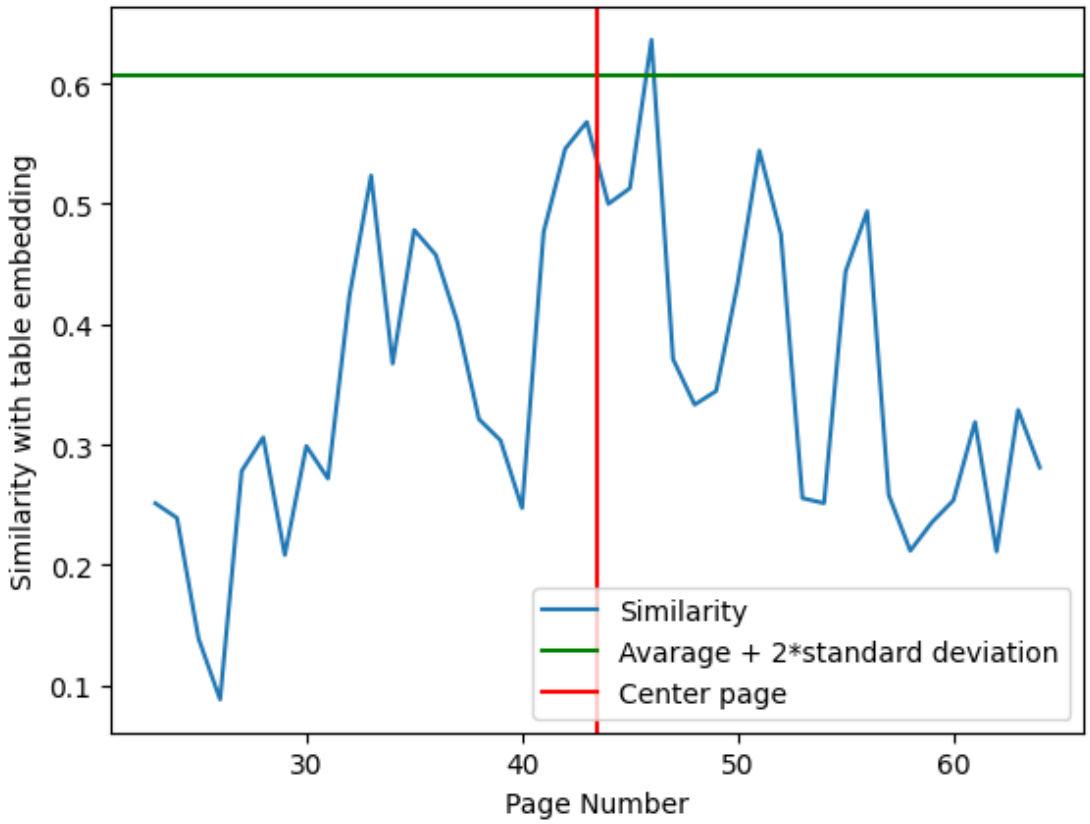

Figure 3.5: Example of a similarity distribution of one table and the surrounding pages, with Gaussian features. "Central page" refers to the page of the table or the central page of all the pages that contain the table, in case of a multi-page table.

Considering this Gaussian assumption, the models were compared on their skewness and kurtosis. In particular, the average and standard deviation of skewness and kurtosis all the tables' similarity distributions, considering the 20 pages surrounding them, were measured and compared. The results are displayed in Table 3.3.

Model	AVG skewness	STD skewness	AVG kurtosis	STD kurtosis
BioSimCSE	0.15	0.56	2.71	0.87
PubMedBERT	0.02	0.64	2.86	1.03
SPECTER	-0.31	0.72	3.00	1.41

Table 3.3: The results for average and standard deviation of skewness and kurtosis all the tables' similarity distributions. The closer to 0 and 3, for skewness and kurtosis respectively, the better.

Finally, the selected threshold, for each table's similarity distribution, is equivalent to the average similarity + 2 times the standard deviation of the distribution. This was chosen in continuity with the Gaussian hypothesis, selecting only the most similar pages without having to decide on a fixed amount of pages, like it would be the case with a top-k selection. The exact amount was chosen as it empirically seemed to be the best conservative cut-off point that consistently selects the pages in which a table is contained, meaning that if any other page is selected with this threshold, it can be assumed to be highly related to the table, at least when disregarding the lack of precision of the embedding model itself.

The final JSON file resulting from this pipeline phase is an array containing, for each table of the dataset, a JSON object with the characteristics described in Table 3.4.

Field	Type	Description
selected_pages_content	Array	The pages selected for the table. Each entry is a string containing the contents of the page.
pages	Array	The pages spanned by the table. Each entry is an integer representing the page in which the table is found.
is_fake_table	Array	A boolean value which is true if the table is used for formatting purposes, false otherwise.
table	Array	It contains the table object, which is an array of arrays.
md	String	A string containing the Markdown serialization of the table.
table_num	Integer	An progressive integer to be used as an identifier of the data point.
study_title	String	The name of the document file the table and text were extracted from.

Table 3.4: The JSON object created for each table during the text extraction and assignment phase.

3.4 Classification prompt chain

The first attempts at classification consisted in trying to include one page's text at a time, and the table, into a simple classification prompt, describing the inputs and the task. These firsts attempts were performed using powerful closed source LLM models, like Claude 3.5 Sonnet and GPT-4o as a proof of concept.

The subsequent attempts, still using the closed sourced models, involved trying to guide the LLM into generating multiple predefined reasoning steps, similar to the chain-of-thought prompting method. In particular, the steps involved dividing the table into chunks, assigning to each chunk the related text present in the table, if present, to formulate, for each table chunk, arguments for why a contradiction was present and why it was not present and finally to chose for each chunk if a contradiction was indeed present or not. The full prompt is available in the Appendix A.1. With these changes, the performance improved for the bigger models, but the only model available for testing via API and with open weights was llama 3.1 70B, which proved to not be powerful enough for such a long and complex prompt. Aside from this issue, having the LLM generate arguments for and against contradiction within the same context is likely to cause unwanted

interference. For these reasons, it has been decided to divide the prompt into several individual prompting steps: table chunking and textual assignment and, for each chunk, contradiction argument, non contradiction argument and final classification.

The chain will be presented as just described, however the final components of the chain (arguments for and against contradiction and the subsequent classification based on them) have been implemented with different variants, based on different LLM prompting techniques. These variants will be explored more in depth in the experimental results chapter.

3.4.1 Models selection and limitations

The first LLM models we tested in the beginning of the project were the smaller models[6], since they were the ones that we could use directly, given our computational resources, but in early mock experiments, with less complexity then the final tasks, they were not able to follow the instructions provided in the prompts. This made it clear that they were not a viable alternative for our purposes.

As stated above, big closed sourced models, like Claude 3.5 Sonnet and GPT-4o, were used via browser prompting, in order to test the maximum results that could be obtained. They were, however, not viable for our purposes either, since the free versions allow for very limited access and they do not allow API calls. Their closed source nature also makes them less useful for this type of work.

The bulk of the experiments, hence, were done using Groq. Groq is a platform that allows for the prompting of several open weights models, via both browsers and API calls. The free tier has a limited rate of tokens for the API calls, which was a limiting factor w.r.t. the scale and number of experiments that were possible. A thing to note is that the models available on Groq are not the biggest available open weights models; the bigger/more performant ones present on the platform are llama-3.1 70B and mixtral 8x7B. After some preliminary results, llama-3.1 70B was chosen for the rest of the experiments.

3.4.2 Text chunking

This prompting step has as input a page and a table and yields as output multiple table chunks with associated related text, if any is present in the page. The table chunks are required to partition the table. This is done in order to force the LLM to take every part of the table into consideration.

This step has two main goals: removing neutral text and serving as the core of a “divide and conquer“ strategy for finding contradictions.

[6]biogpt, biogpt-large and llama-3 8B-Q4

With regards to the neutral text removal, the entire pipeline of this project can be though of as a two-stage text selection process, with a first broader selection, performed using heuristics and a smaller more efficient model, and a second more precise selection performed by a bigger model. This is done in order to reduce computational demands while maintaining a reasonable level of performance.

As for the "divide and conquer" strategy, it is made possible by the nature of the problem: text and tables are in contradiction if any of their parts are in contradiction with one another. This allows for the decomposition of the bigger problem into smaller problems that can actually be solved by the LLM. It has proven to be an essential strategy for the bigger models as well. As can be observed in the prompt used for Claude and GPT (A.1), before chunking the LLM is prompted to asses if the overall table and text are in contradiction with one another. Even when answering correctly, by finding a contradiction in one of the chunks, both models failed in all the experiments to asses the overall text and table as being contradictory before performing the chunking operation.

The final version of the standalone chunking prompt is present in the Appendix A.2. One thing of note is that the LLM takes as input the 2D array table format exported by the OCR. This is not ideal, but it is done in order to avoid possible artifacts from the conversion to Markdown that could impede the LLM's capability of interpreting the meaning of the table. The LLM is, however, prompted to output tables chunks, with associated text, in the Markdown format, since any artifacts present at this point are assumed to be due to the original 2D array representation or due to the LLM model's table comprehension capabilities.

After some trivial text manipulation, using intermediate anchor tokens the LLM is prompted to insert, the resulting output of this step is a list of table chunks, with the necessary header and written in the Markdown format, and the associated text for each chunk.

3.4.3 Arguments in favour and against contradiction

In this step of the prompt chain, for each chunk resulting from the previous step, an argument for and against a contradiction being present is generated. As stated above, this is inspired by the chain-of-thought prompting method.

As mentioned, chain-of-thought prompting is a popular few-shot learning method for improving the performance of LLM models. It involves the insertion, in the solved tasks examples provided within the prompt, of the reasoning steps required in order to arrive at the solutions provided with said examples.

This method does, however, not fit our dataset. Considering the scarcity of diverse data points and the lack of annotated reasoning steps, the zero-shot-learning method is the only practical choice for our purposes. Without examples it is still, however, possible to obtain an improvement by just prompting the LLM for an

explanation, as proved by Takeshi Kojima et al.[17].

Ideally, an integration of the zero-shot chain-of-thought prompting with the answer multi-sampling technique of Xuezhi Wang et al. [18] could have been employed at this prompting stage, in order to assign a "contradiction" or "non contradiction" label to each chunk created in the previous step. We did implement such an integrated method, however it could only be based on multi-sampling and majority voting, since none of the intermediate LLM products were available. It is also note worthy that this sampling method requires at least 3 different generations (in order to reach a majority consensus), which makes it costly in terms of requested resources.

The method we devised, in order to try to maintain the same exploration of "a diverse set of reasoning paths", while also reducing the amount of required generations, consists in first prompting the LLM for an argument in favor of a contradiction being present and then, with a second prompt, requiring an argument for why a contradiction is not present. These two arguments will then be injected in the next prompting phase, allowing the LLM to evaluate the two and decide for the more convincing argument. Ideally this allows the LLM to explore two radically different reasoning paths, as with [18], and then evaluate the results. This also reduces the number of required generations to only two, allowing for more computational efficiency.

Appendix A.3 and A.4 contain the prompts for the contradiction and non contradiction arguments respectively. Noteworthy is that the prompt in favor of contradiction is much better specified, and part of this specification is also present in the classification prompting step. The argument for contradiction prompt also contains the instruction to not leave the answer blank, unless there is no text associated with the table. These differences all have the aim of biasing the model towards deciding that a given pair of table and text chunks is contradictory. As stated before, as long as there is not an excessive amount of false positives, it is preferable to have more false positives to be manually examined, while also increasing the amount of true positives, rather then having them both be lower. The first case might lead to more time wasted on the part of the researcher, but the second case might lead to the rejection of the document, resulting in orders of magnitude more wasted time.

3.4.4 Final classification

The final prompting step is the classification step. In this phase, for each of the chunks generated in the chunking phase, a prompt is created in which the chunk and the two arguments for and against contradiction are embedded. The LLM is then instructed to decide whether a given chunk is contradictory or not. The prompt used in this phase is present in the Appendix A.5. Finally, for each page

and table pair, if even one of the chunks derived from it is considered contradictory, the entire page-table pair is considered contradictory.

It is worth mentioning the type of output that can be provided by this pipeline. If a given contradiction is identified, it is possible to obtain: the full table and page in which a contradiction is detected, a table and text snippets that are assumed to be problematic and a description of the contradiction that was identified (the argument in favor of the contradiction generated by the LLM). This provides explainability and it also allows the domain expert to easily accept or reject the contradiction detected.

3.4.5 Synthetic data generation

Although not part of the prompt chain, the process of synthetic data points generation was fundamental for testing the possible capabilities of the software. Said process is based on the creation of contradictory text starting from real data points, using LLM prompting.

A point of concern in the generation of fake data points has been that the models used to classify the contradictions have been the same ones used to generate the synthetic dataset. This represents an issue since it is reasonable to assume that an LLM can generate a contradictory data point only if it can identify it as contradictory, which would mean that our synthetic examples would, by definition, be too easy for the model to identify and would not represent a good proxy for the performance on real data points. One way we tried to mitigate this problem is by generating text that is contradictory w.r.t. to original text without taking into consideration the table the text was associated to. The synthetic text should contradict the original and, if the original is coherent with the table, it should contradict the table as well, but, crucially, the contradiction between synthetic text and table is not directly generated by the model, which might allow for types of contradictions that the model is not able to identify. This is, however, just a mitigation strategy, and the solution is most likely still biased towards a dataset that is easier than real world examples.

Given that a genuine data point, used as an input to the prompt chain, is comprised of a table and a page of text, a synthetic data point needs to have the same format. Starting from this premise, the generation of a fake contradictory data point using a real (assumed to be) non contradictory one as basis has three main steps: text snippet extraction, contradiction generation and modified snippet insertion.

The reason for this added complexity is that a generic page might only have a small (or null) portion of text related to the table it is associated to. This is the expected generic case for the majority of the data points. For the creation of a contradictory page, w.r.t. the table, the only part that should be modified is the

part regarding the table, and nothing else. For this reason, the textual snippet of interest, if existent, needs to be identified (and extracted), then modified and reinserted into the original page. In theory all the process could be condensed into a single step of rewriting the page while modifying only the textual parts related to the table, but this would render the generation of the contradiction dependent on the content of the table, which we wanted to avoid, and also in practice dividing the process into steps seems to yield better results.

With regards to text snippet extraction, two methods were tested. The first method was based on the same chunking mechanism discussed above, included in the prompt chain. For each genuine data point, several chunks would be created. Among the non null chunks, one would be randomly selected and the text present in it would be the extracted snippet for the data point. The second tested method was to prompt the LLM for a single sentence that is entailed by the table. The entailment criteria implies a much stronger connection between text and table, and also implies the necessity that a sentence that is in contradiction with the original sentence is also in contradiction with the table.

The contradiction generation and modified snippet insertion have been combined into a single step, for computational efficiency. The contradiction generation is based on a three shot learning prompting strategy. The prompt is comprised of three random pairs (premise and contradiction) of sentences extracted from the SNLI dataset[2] and then the real snippet extracted from our dataset, and the page the snippet is extracted from. The random SNLI samples were included with the intention of adding more variety to the type of contradictions generated, hopefully creating a distribution that more closely resembles a real world distribution. The prompt is then concluded by requiring that the page be re-written, with the modified textual snippet.

Chapter 4

Experimental results

Most of the experimental results are focused on testing and analysing the performance of the final part of the pipeline, which is the classification component of the prompt chain. As stated in previous chapters as well, testing the entire pipeline using synthetic data was not feasible, since it would imply generating entire fake documents, moreover doing so in a highly specialized field. Without being able to generate such documents as inputs, in order to asses the performance of the entire pipeline, only real cases can be used. Considering, however, that the real cases of contradictions available were only 2, the only real statistically significant testing of the entire pipeline that we could perform with our dataset was that of evaluating the number of false positives yielded by our software.

Another key factor to consider, in this instance as well, is the limitation imposed by the restrictions on computational resources. After pairing pages and tables, using the first components of the pipeline, the resulting number of data points present in our dataset (each data point comprised of a table and a page) were 1006. This was one order of magnitude more than what could be reasonably analyzed/experimented upon with the resources available for this project. For this reason, most of the results shown have been obtained with a sample size of 200. For the experiments where this is not true, it will be explicitly stated.

4.1 Simplified pipeline exploratory experiments

As described in Subsection 1.3.1 of the introduction, before the creation of the entire pipeline, a set of experiments, reduced in scope, were devised in order to asses the feasibility of the project and in order to also try to estimate the possible future performance of the pipeline.

These experiments were based on manually extracted data points, each comprised of a handpicked table and a handpicked sentence, with the tables having

been preprocessed by hand. Two of the originally extracted data points were contradictory, five were considered entailments and seven were considered neutral. This three way classification was based on NLI, and was adopted in the earlier stages of the project, but later abandoned in favour of the binary classification present in the rest of the project. It is however noteworthy that the three way classification task is still necessary in order to be able to use BERT-like models, fine tuned for the NLI task.

Given the lack of contradictory data points, synthetic data points were created, based on the sentences present in the data points classified as "entailment" and using the same three shot learning prompting mechanism later employed in the full pipeline.

The performance was tested on a large commercially available LLM (Claude), on llama-3.1 and on Roberta-large-snli, a BERT-like NLI fine tuned model, whose capabilities in finding contradictions between text and tables were promising, according to [3].

The results of these first experiment are shown in table 4.1, whereas table 4.2 shows the conversion of the results into the binary classification task, considering neutral and entailment as belonging to the single category of "not a contradiction". As shown by both tables, the results were very promising for the two LLM models, whereas the Roberta model did not achieve a satisfactory level of performance on this simplified task and was not considered for the full pipeline.

Model	Contradictions	Entailments	Neutrals
Claude-3.5-Sonnet	6/7	4/5	7/7
llama-3.1-70b-versatile	6/7	2/5	7/7
Roberta-large-snli	6/7	1/5	2/6

Table 4.1: First experiment results

Model	Contradictions	Not contradictions
Claude-3.5-Sonnet	6/7	12/12
llama-3.1-70b-versatile	5/7	12/12
Roberta-large-snli	6/7	4/12

Table 4.2: First experiment results when considered as a binary classification task

4.2 Full pipeline experiments and ablation studies

As stated above, the majority of the experiments were focused on the final classification prompt chain, and in particular on the final classification prompts. However, one notable exception is the experiment on the false positive rate of the pipeline.

4.2.1 Aside: Claude and GPT prompts

Before setting up the prompting classification pipeline, some experiments were done using Claude 3.5 Sonnet and GPT-4o, using the prompt shown in the Appendix A.1, in order to try and classify correctly the two real contradictory examples. Not being able to set the the temperature parameter meant that different experiments with the same prompt yielded different results, but the majority of the experiments succeeded in identifying both data points as contradictory. Although these results provide only indications, considering the small sample size and the methodological concerns, they seem to suggest that substantially better results could be achieved with bigger models, considering that:

- these results were achieved using a single prompt, without the benefits of splitting the classification prompt into a prompt chain, as seen with the llama model

- even with the prompt chain, the llama-3.1-70B model could not consistently classify both real data points as contradictory

4.2.2 False positive rate

The false positive rate is an important metric for our project since each falsely labeled contradiction requires manual elaboration. It is less vital than contradiction recall, but if too high, it would render the software unusable.

As with the experiments that followed, the false positive rate was a performance assessment of different prompting strategies. These strategies are based on the arguments for and against contradictions method, described in subsection 3.4.3, and on the findings of [17] and [18]. The model employed was llama-3.1-70b-versatile model. The results are shown in table 4.3.

The “Arguments for and against“ prompting strategy is the one already described in the appendix, whereas the “Arguments for and against & Zero-shot-CoT“ only adds the “Let’s think step by step“ sentence at the end of the prompt, as suggested by [17].

The "Zero-shot-CoT" strategy involves removing the arguments for and against a contradiction being present, leaving only the classification prompt with "Let's think step by step" at the end. The full prompt is present in appendix A.6.

The "Baseline" prompting strategy consists in the same simplified classification process as the "Zero-shot-CoT" strategy, but it does not contain the "Let's think step by step" ending phrase.

Finally "Self consistency & Zero-shot-CoT" combines the works of [17] and [18]. The same prompting strategy as "Zero-shot-CoT" is used, but 5 answers are sampled from the LLM model and the majority classification answer is selected. Note that this is the simplest self consistency strategy presented by the [18] paper, but it was also the only one available without the access to all the LLM model's outputs (e.g. logits).

Prompt type	**false positive rate**
Baseline	0.12
Arguments for and against	0.115
Arguments for and against & Zero-shot-CoT	0.23
Zero-shot-CoT	0.12
Self consistency & Zero-shot-CoT	0.1

Table 4.3: False positive rate for different prompting strategies on only real data points, supposed to all be non contradictory

The results on the real data points are very promising, however contradictions recall is the fundamental metric for assessing the success of the project. Noteworthy is the drop in performance of the "Arguments for and against & Zero-shot-CoT" strategy. This strategy seems to have a bias towards classifying samples as contradictions, hypothesis that is also supported by later experiments. This does indeed degrade the performance with regards to precision, however it also improves the recall, which is a trade-off that could be beneficial, as suggested previously.

Finally, a qualitative analysis of the resulting descriptions' output by the pipeline also shows that a majority of the false positives can be easily identified as such by a human operator since they often either lack a text or table chunk or they contain a hallucinated table chunk not present in the original.

4.2.3 Ablation studies: classification prompts

The majority of the experiments were focused on trying to estimate the capabilities of different prompting classification strategies, as done with the false positive rate. This time, however, synthetic data points were employed in order to try assessing

the actual performance of the software on both contradictory and non contradictory data points. As stated before, two different synthetic datasets were used, derived with two slightly different methodologies. The results of the experiments on the two different datasets are presented in table 4.4 and table 4.5. A thing of note is the ratio of contradictory to non contradictory data points chosen. In order to maintain this experiments comparable with the ones that followed it, the contradictory data points represented 26.5% of the total data points, in line with the proportion found in the PubHealthTab dataset. All the experiments were performed using the llama-3.1-70b-versatile model.

The best result achieved on the first synthetic dataset was an accuracy of 0.765, with a contradiction recall of 0.358, achieved by the "Zero-shot-CoT" strategy. Considering only the contradiction recall metric, however, the best result was 0.444, achieved by the "Arguments for and against & Zero-shot-CoT". This performance was below our expectations, considering also the previous results on the simplified version of the task and the ones regarding the false positive rate. Suspecting that one of the issues was at least partially related to the the synthetic data generation process, a second process (3.4.5) was devised in order to try to increase the likelihood that the generated contradictions were in actual contradiction with the table.

With this second synthetic dataset the best performing strategy was still the "Zero-shot-CoT", with an accuracy of 0.805 and a contradiction recall of 0.547 but again, considering only recall, the best strategy was the "Arguments for and against & Zero-shot-CoT", with a score of 0.585. The contradiction recall did indeed improve (as did the precision, to a lesser degree), but it was still not satisfactory. In order to understand whether this drop in performance was due to a flaw in the classification prompt chain, a flaw in the synthetic data creation process or due to the increased complexity of the task (handling entire pages instead of just sentences) further experiments were devised.

Noteworthy are also the following observations:

- the "Self consistency & Zero-shot-CoT" strategy's performance was always slightly worse than that of "Zero-shot-CoT", although the difference is not significant. A possible explanation is that the model tends to give consistent answers across multiple samples. If that is the case, integrating the "Arguments for and against" strategy with Self consistency could possibly improve the performance, by allowing more diverse paths to be explored.

- "Arguments for and against & Zero-shot-CoT" performs significantly better than just the "Arguments for and against" strategy, when considering just recall on the first dataset and in regards to overall performance with the second dataset. It is not the best overall performer, however it is the only strategy that managed to at least once classify correctly both of the real contradictory

data points. This should be further explored with more data and for now represents only an indication that the performance of this method might be misleading w.r.t. its performance on real world data. As stated before, though, this strategy appears to have an overall bias towards labeling data points as contradictory, so the results might be slightly misleading.

Prompt type	Real contradictions accuracy	Accuracy	Contradiction recall	Contradiction precision
Baseline	0.5	0.74	0.415	0.512
Arguments for and against	0.5	0.695	0.321	0.405
Arguments for and against & Zero-shot-CoT	0.5	0.675	0.444	0.407
Zero-shot-CoT	0.5	0.765	0.358	0.594
Self consistency & Zero-shot-CoT	0	0.75	0.302	0.552

Table 4.4: Performance of different prompting strategies on the first synthetic dataset

Prompt type	Real contradictions accuracy	Accuracy	Contradiction recall	Contradiction precision
Baseline	0.5	0.74	0.377	0.513
Arguments for and against	0.5	0.57	0.426	0.657
Arguments for and against & Zero-shot-CoT	1	0.69	0.585	0.437
Zero-shot-CoT	0.5	0.805	0.547	0.659
Self consistency & Zero-shot-CoT	0.5	0.8	0.528	0.651

Table 4.5: Performance of different prompting strategies on the second synthetic dataset

4.2.4 PubHealthTab performance

In order to investigate whether there was a major flaw in the classification prompt chain, said chain was also tested on the PubHealthTab dataset. PubHealthTab's data points are comprised of a sentence (claim) and a table, and the relation between the two can be "supports", "refutes" and "related but not enough information". "supports" and "related but not enough information" have been treated as "not a contradiction" whereas "refutes" has been treated as a contradiction. The fact that the textual part of PubHealthTab is comprised of a single sentence did require some modifications to the chain and, to a lesser extent, to the prompts, since chunking was not applicable. The performance of the different strategies can be seen in table 4.6.

An analysis of the results reveals that the accuracy scores seem to be higher and more homogeneous, whereas the contradiction recall is significantly higher and the same trends observed in the synthetically generated datasets can be observed here as well. A notable exception is that the best performing strategy, for what concerns accuracy is, in this case, the "Self consistency & Zero-shot-CoT" whereas the best one in regards to recall is "Zero-shot-CoT", but again the differences do not appear to be significant. "Arguments for and against & Zero-shot-CoT" does

not appear to perform as well on this dataset, especially on the recall metric, but it continues to improve on the performance of "Arguments for and against".

Overall, the results of the software on this dataset have proven to be significantly higher, with recall levels that are more in line with the requirements of real world applications. This suggests that the drop in performance is unlikely to be attributable to a flaw in the classification prompt chain itself. These tests cannot, however, account for the chunking section of the pipeline, as stated above, so this conclusion is only partial. It does, however, appear that the drop could be due to a flaw in the synthetic data creation process or due to the increased complexity of the classification task required by our dataset.

Prompt type	Accuracy	Contradiction recall	Contradiction precision
Baseline	0.84	0.849	0.652
Arguments for and against	0.795	0.509	0.643
Arguments for and against & Zero-shot-CoT	0.8	0.660	0.614
Zero-shot-CoT	0.875	0.868	0.719
Self consistency & Zero-shot-CoT	0.895	0.830	0.786

Table 4.6: Performance of different prompting strategies on PubHealthTab

4.2.5 Investigating the performance drop: simplified second synthetic dataset

Following the intuition that the synthetic data creation process or the added complexity might be the reason why the performance drop occurred, we created a new, simplified version of the second synthetic dataset, using the same extracted sentences, and their contradictory counterparts, employed in the creation of said synthetic dataset. For each (original) sentence and its contradiction, two data points were created (one contradictory and one not), containing the same associated table and one of the two sentences. Afterwards, a subset with the same distribution as the PubHealthTab's sample was randomly selected and tested, with the same prompting strategies. The results are shown in table 4.7.

Looking at the results, the performance metrics appear much more similar to the ones related to the PubHealthTab dataset. The accuracy and recall metrics are also almost all inferior or equal to the ones obtained on the PubHealthTab dataset, suggesting that the contradictory data points generated by the LLM model were indeed not trivial to detect. The trends do appear to be, however, different and harder to interpret. For instance, the best performing strategy is the "Baseline" strategy, for all the metrics considered. These differences in the observed trends might be an indication that the data generation process might not yield a completely realistic distribution when compared with real world data. The substantial increase in performance, when compared with the results of the

complete prompt chain, suggests, however, that the overall drop in performance is due to the increased complexity of the task of classifying data points comprised of a full page of text, rather than of a single sentence.

Prompt type	Accuracy	Contradiction recall	Contradiction precision
Baseline	0.895	0.830	0.786
Arguments for and against	0.795	0.623	0.611
Arguments for and against & Zero-shot-CoT	0.725	0.528	0.483
Zero-shot-CoT	0.86	0.792	0.712
Self consistency & Zero-shot-CoT	0.855	0.774	0.707

Table 4.7: Performance of different prompting strategies on simplified second synthetic dataset

4.2.6 Overall analysis

Unfortunately the results do not lend themselves to an easy interpretation. Considering only real data points, the results of the experiments suggest that the performance of the pipeline might be in line with what required for a usage of the software in real world scenarios. Also considering, however, the real contradictory data points, that might not be the case, but the sample size is too small to draw any meaningful conclusion.

With regards to the synthetic datasets generated, the results on the simplified version of the second dataset suggest that the generation process could have yielded samples that indeed reproduce a realistic distribution, at least when considering the second synthetic dataset. The inconsistencies in the resulting performance trends, considering the expected results and the ones obtained on the PubHealthTab dataset, could however be considered an indication of artifacts present in these datasets that are not present in real world distributions.

A more clear conclusion is that the length of the textual passage considered for each data point significantly degrades performance, especially regarding the contradiction recall. This more likely is because the contradictory text, if present, is usually a very small percentage of the overall text provided, making the task also in part related to the needle in a haystack problem. The chunking section of the prompt chain was in part able to mitigate this effect, but these results indicate that the impact of this section on the overall performance might be even greater than what initially hypothesized. As such, focusing on better developing and understanding the effects of different prompting strategies in this phase might lead to significant gains in performance.

Finally, based on the results, the most likely candidate strategy for real world application would the the “Zero-shot-CoT“ one. With more data it would, however,

be beneficial to also study the effects of the "Arguments for and against & Zero-shot-CoT" as it is unclear whether it could function better with real world distributions.

4.2.7 Qualitative analysis by domain experts

The results of one of the false positives experiments, alongside the correct classification of the two real contradictory cases, were reviewed by domain experts in order to asses the viability of the outputs in the context of real world usage. As described in the previous chapter, the output of the software, for each contradiction found, consists of the full table and page in which a contradiction is detected, a table and text snippets, that are assumed to be contradictory, and an explanation of the contradiction that was identified.

With regard to the true positives, the explanations provided by the model were deemed as appropriate and accurate, whereas the false positives were considered, for the most part, easily identifiable, since the model would either quote or use table/textual snippets not actually present in the original table/text of the data point being classified. The residual cases were harder to identify, since they required parsing through the information and arguments provided, but in one instance it was also discovered that one of the false positives contained a useful suggestion for a potential inconsistency that was not originally identified and that could have led to a rejection.

Finally, there was also a request for integrating more information in each of the presented samples, such as page numbers, document names, etc., but the overall judgment regarding the software was positive and, if possible, real world testing would be encouraged.

Chapter 5

Conclusions

This project represents both a prototype and, potentially, the first part of a software ensemble aimed at solving a real world problem, in the field of medical documentation, by employing the new avenues afforded by the emergence of new predominant deep learning models, such as LLM models.

It appears to have succeeded in its goals as a proof of concept, in so far as defining a bounded and clear objective, constructing a software component to (for the most part) automatically achieve said objective and obtaining a final result that, already in its current form, could be tested by experts of the field.

As an exploratory project we also succeeded in more clearly defining the general problem and also in defining what components the complete software product would need in order to achieve its original goal of finding any potential contradictions and/or missing textual elements in clinical trial proposal documents.

One area in which the project did not achieve a fully satisfactory conclusion, unfortunately, is that of result analysis and enhancements, two areas closely related. The scarcity of real contradictory data points has impeded the analysis of the results of most segments of the created pipeline, and the results that were achieved with the usage of synthetic data points can only be used as possible indications, while also presenting hints of anomalies introduced by the generation process. On the other hand, the trade-off between the size of the LLM models and direct control over the code and the generated products has limited the fine tuning capabilities to only some forms of prompt engineering, while also not allowing for fully replicable results (e.g. not being able to set the temperature parameter to 0).

With regards to possible future endeavors, as stated before, one easy improvement of great impact is rendering the pipeline fully automatic, by having access to the source files and using the resulting computer-interpretable structure in order to rewrite the software component that now requires manual processing in order to function. This step would also allow for the possibility of using the software in real life contexts and being able to qualitatively asses its usefulness. Closely

related to this end is also the possible creation of guidelines for writing more computer friendly clinical trial proposal documents, especially when it comes to tables. More standardization and better structuring would render the entire pipeline more effective in its stated final goal.

In general, also improving the availability of real word examples of errors found in clinical trial proposal documents, by increasing the quantity of available data by some orders of magnitude, would greatly improve the analysis of the general problem and would also allow for better evaluation of the performance of the pipeline as a whole. With regards to the analysis of the general problem, it would allow us to confirm that the major common issues found in these documents are indeed the ones identified in the present work, with the same proportions. Assuming that the same proportions are present, this would provide possibly sufficient data points in order to better characterize the performance of the pipeline when applied to real world data.

Another area of improvement is that of the analysis and further development of the pipeline module dedicated to associating text and tables. In this case real world data is required since generating it synthetically would be akin to trying to generate full clinical trial proposal documents, as stated before. Two key areas of analysis of performance and possible improvements are that of the SBERT-like models employed and that of the textual subdivision. With the latter, it would appear from the results of the project that finer grain segmentation leads to better the results. In this regard, the chunking phase of the final prompt chain could be in part replicated at this stage. Smaller-size LLM models could be employed for better textual segmentation, maintaining a balance between fidelity and performance. Even if the chunking mechanism was to be kept as a component of the prompt chain, in some capacity, further ablation studies on its effects and experimentation with different prompts would constitute additional avenues for further improvements.

Finally, being able to deploy the llama model that formed the basis of most of the experiments would allow for the testing of more sophisticated prompting methodologies (e.g. logits based self-consistency), more replicability and more possibilities for the improvement of the experimental results, via methods like LoRA fine-tuning [21] or prompt tuning [22].

Appendix A

Prompt templates

The following are a collection of the prompt templates used in the various experiments. The parts in "<>" brackets represent the variables injected into each template.

A.1 Prompt for direct classification without a pipeline

The following are rows of a table and a page of sentences probably related to the table, in the domain of clinical trial proposals. The formatting is simplified, so complex structures, such as superscripts and subscripts, need to be inferred. A 'None' element in a row of the table implies that the first not 'None' element occupies that column of the row as well.
Page: <page>
Table: <table>
Given the table and the page, firstly asses whether the content of the page contradicts in any way the content of the table. Only answer with either "contradiction" or "not a contradiction". In case of a contradiction, a separate line, give a brief explanation of why the choice was made.
Afterwards, add a "<SEP>" token and then divide the table into rows with common meaning (chunks).
The chunks need to partition the table completely, and they need to all be expressed.
For each table chunk:
1. Write, in valid markdown and in full, the rows of the table belonging to that chunk, with all the necessary header rows included for each chunk.
2. Assign a title to each table chunk, underlining the commonality.
3. Write, in full, all the passages present in the page that are the relevant or related

to the table chunk. The same passage can be related or relevant to multiple table chunks and it is required that to be present in each one of them.
4. Write a proof against the passages being in contradiction with the table
5. Write a proof in favour of the passages being in contradiction with the table. DO NOT LEAVE THE PROOF EMPTY OR "NONE" UNLESS NO RELEVANT PASSAGES WERE SELECTED. If there is information implied in the passage that is not present in the table chunk and it is incompatible with the table chunk, it should be considered a contradiction. If the information implied by the passage is inconsistent with what is written in the table chunk, it should be considered a contradiction. Finally, if the information could be interpreted as contradictory, it should be considered a contradiction.
6. After this, add an "<EXP>" token.
7. At the end of the proofs, and after the "<EXP>" token, only answer with either "contradiction" or "not a contradiction", by choosing the most plausible proof. If there is information implied in the passage that is not present in the table chunk and it is incompatible with the table chunk, it should be considered a contradiction. If the information implied by the passage is inconsistent with what is written in the table chunk, it should be considered a contradiction. Finally, if the information could be interpreted as contradictory, it should be considered a contradiction.
8. After the chunk, the proof and the "contradiction"/"not a contradiction", add a "<SEP>" token.

A.2 Prompt for chunking

The following are rows of a table and a page of sentences probably related to the table, in the domain of clinical trial proposals. The formatting is simplified, so complex structures, such as superscripts and subscripts, need to be inferred. A 'None' element in a row of the table implies that the first not 'None' element occupies that column of the row as well.
Page: <page>
Table: <table>
Divide the table into rows with common meaning (chunks). The chunks need to partition the table completely, and they need to all be expressed.
For each table chunk:
1. Assign a title to each table chunk, underlining the commonality.
2. Write, in valid markdown and in full, the rows of the table belonging to that chunk, with all the necessary header rows included for each chunk.
3. Add a "<TXT>" token
4. Write, in full, all the passages present in the page that are the relevant or related to the table chunk. The same passage can be related or relevant to multiple table

chunks and it is required that to be present in each one of them.
5. Add a "<SEP>" token
If possible, try to assign each line of text to at least a chunk, but only if the two are at least in some way logically related.

A.3 Prompt for argument in favour of contradiction

The following is an extracted chunk of a table, written in markdown, and a list of sentences, that are supposed to be related to the chunk, in the domain of clinical trial proposals. At the beginning of the chunk there is a title, implying the commonality among the selected rows. The formatting is simplified, so complex structures, such as superscripts and subscripts, need to be inferred.
Table chunk: <Table chunk>
Selected text: <Text chunk>
Given the table chunk and the associated text, write a proof for why the text contradicts what's written in the table.
DO NOT LEAVE THE PROOF EMPTY OR "NONE" UNLESS NO RELEVANT PASSAGES WERE SELECTED.
If there is information implied in the passage that is not present in the table chunk and it is incompatible with the table chunk, it should be included in the proof for contradiction.
If the information implied by the passage is inconsistent with what is written in the table chunk,it should be included in the proof for contradiction.
Finally, if the information could be interpreted as contradictory, such interpretation should be included in the proof for contradiction.

A.4 Prompt for argument against contradiction

The following is an extracted chunk of a table, written in markdown, and a list of sentences, that are supposed to be related to the chunk, in the domain of clinical trial proposals. At the beginning of the chunk there is a title, implying the commonality among the selected rows. The formatting is simplified, so complex structures, such as superscripts and subscripts, need to be inferred.
Table chunk: <Table chunk>
Selected text: <Text chunk>
Given the table chunk and the associated text, write a proof for why the text does not contradict what's written in the table.
The proof is required to be logically sound and cannot simply state that text and

table are coherent without proving how.

A.5 Prompt for chunk classification: arguments for and against

The following is an extracted chunk of a table, written in markdown, and a list of sentences, that are supposed to be related to the chunk, in the domain of clinical trial proposals. At the beginning of the chunk there is a title, implying the commonality among the selected rows. The formatting is simplified, so complex structures, such as superscripts and subscripts, need to be inferred. After the table chunk and text, an argument for why the text is contradictory w.r.t. table is presented and then a argument for why it is not contradictory.

Table chunk: <Table chunk>
Selected text: <Text chunk>
Argument in favour of the contradiction: <Argument in favour of the contradiction>
Argument against the contradiction: <Argument against the contradiction>

Given the table chunk, the associated text, and the two arguments, decide if the selected text is contradictory w.r.t. the table chunk or not.
If no text is provided, then it cannot be a contradiction.
If there is information implied in the passage that is not present in the table chunk and it is incompatible with the table chunk, it should be considered a contradiction.
If the information could be reasonably interpreted as contradictory or inconsistent, it should be considered a contradiction.
If the text chunks or table chunks are missing or are devoid of meaning, it should not be considered a contradiction.
Base the decision on the two arguments provided. Only answer with either "contradiction" or "not a contradiction".

A.6 Prompt for chunk classification: Zero-shot CoT

The following is an extracted chunk of a table, written in markdown, and a list of sentences, that are supposed to be related to the chunk, in the domain of clinical trial proposals.
At the beginning of the chunk there is a title, implying the commonality among

the selected rows.
The formatting is simplified, so complex structures, such as superscripts and subscripts, need to be inferred.

Table chunk: <Table chunk>
Selected text: <Text chunk>

Given the table chunk and the associated text, decide if the selected text is contradictory w.r.t. the table chunk or not. If no text is provided, then it cannot be a contradiction. If there is information implied in the passage that is not present in the table chunk and it is incompatible with the table chunk, it should be considered a contradiction. If the information could be reasonably interpreted as contradictory or inconsistent, it should be considered a contradiction. If the text or table chunk are missing or are devoid of meaning, it should not be considered a contradiction. First reason step by step and then, after inserting a "<SEP>" token, only answer with either "contradiction" or "not a contradiction".

Let's think step by step

Bibliography

[1] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: 1908.10084 [cs.CL]. URL: https://arxiv.org/abs/1908.10084 (cit. on p. 3).

[2] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. *A large annotated corpus for learning natural language inference*. 2015. arXiv: 1508.05326 [cs.CL]. URL: https://arxiv.org/abs/1508.05326 (cit. on pp. 4, 29).

[3] Mubashara Akhtar, Oana Cocarascu, and Elena Simperl. «PubHealthTab: A Public Health Table-based Dataset for Evidence-based Fact Checking». In: *Findings of the Association for Computational Linguistics: NAACL 2022*. Ed. by Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 1–16. DOI: 10.18653/v1/2022.findings-naacl.1. URL: https://aclanthology.org/2022.findings-naacl.1 (cit. on pp. 4, 9, 31).

[4] Sharon Jiang. «ConflictClassifier: An Interpretable Pipeline to Identify Contradictions in Clinical Notes». eng. In: *2021 IEEE MIT Undergraduate Research Technology Conference (URTC)*. IEEE, 2021, pp. 1–5. ISBN: 9781665405959 (cit. on p. 6).

[5] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. «ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing». In: *Proceedings of the 18th BioNLP Workshop and Shared Task*. Association for Computational Linguistics, 2019. DOI: 10.18653/v1/w19-5034. URL: http://dx.doi.org/10.18653/v1/W19-5034 (cit. on p. 6).

[6] Alistair E.W. Johnson et al. «MIMIC-III, a freely accessible critical care database». In: *Scientific Data* 3.1 (May 2016), p. 160035. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.35. URL: https://doi.org/10.1038/sdata.2016.35 (cit. on p. 7).

[7] Alexey Romanov and Chaitanya Shivade. «Lessons from Natural Language Inference in the Clinical Domain». In: *arXiv:1808.06752 [cs]* (Aug. 21, 2018). arXiv: 1808.06752. URL: http://arxiv.org/abs/1808.06752 (visited on 08/27/2018) (cit. on p. 7).

[8] Halil Kilicoglu, Graciela Rosemblat, Marcelo Fiszman, and Dongwook Shin. «Broad-coverage biomedical relation extraction with SemRep». In: *BMC Bioinformatics* 21.1 (May 2020), p. 188. ISSN: 1471-2105. DOI: 10.1186/s12859-020-3517-7. URL: https://doi.org/10.1186/s12859-020-3517-7 (cit. on p. 7).

[9] Halil Kilicoglu, Dongwook Shin, Marcelo Fiszman, Graciela Rosemblat, and Thomas C. Rindflesch. «SemMedDB: a PubMed-scale repository of biomedical semantic predications». In: *Bioinformatics* 28.23 (Oct. 2012), pp. 3158–3160. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts591. eprint: https://academic.oup.com/bioinformatics/article-pdf/28/23/3158/50695211/bioinformatics_28_23_3158.pdf. URL: https://doi.org/10.1093/bioinformatics/bts591 (cit. on p. 7).

[10] Graciela Rosemblat, Marcelo Fiszman, Dongwook Shin, and Halil Kilicoglu. «Towards a characterization of apparent contradictions in the biomedical literature using context analysis». en. In: *J Biomed Inform* 98 (Aug. 2019), p. 103275 (cit. on p. 7).

[11] Prajwol Lamichhane, Indika Kahanda, Xudong Liu, Karthikeyan Umapathy, Sandeep Reddivari, Catherine Christie, Andrea Arikawa, and Jenifer Ross. «Poster: BioFactCheck: Exploring the Feasibility of Explainable Automated Inconsistency Detection in Biomedical and Health Literature». In: *2023 IEEE/ACM Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. 2023, pp. 196–197. DOI: 10.1145/3580252.3589435 (cit. on p. 7).

[12] Cheng Hsu, Cheng-Te Li, Diego Saez-Trumper, and Yi-Zhan Hsu. *WikiContradiction: Detecting Self-Contradiction Articles on Wikipedia*. 2021. arXiv: 2111.08543 [cs.CL]. URL: https://arxiv.org/abs/2111.08543 (cit. on p. 7).

[13] Mohammad Javad Hosseini, Andrey Petrov, Alex Fabrikant, and Annie Louis. *A synthetic data approach for domain generalization of NLI models*. 2024. arXiv: 2402.12368 [cs.CL]. URL: https://arxiv.org/abs/2402.12368 (cit. on p. 8).

[14] Tu Vu, Minh-Thang Luong, Quoc V. Le, Grady Simon, and Mohit Iyyer. *STraTA: Self-Training with Task Augmentation for Better Few-shot Learning*. 2022. arXiv: 2109.06270 [cs.CL]. URL: https://arxiv.org/abs/2109.06270 (cit. on p. 9).

[15] Zihui Gu, Ruixue Fan, Xiaoman Zhao, Meihui Zhang, Ju Fan, and Xiaoyong Du. «OpenTFV: An Open Domain Table-Based Fact Verification System». In: *Proceedings of the 2022 International Conference on Management of Data*. SIGMOD '22. Philadelphia, PA, USA: Association for Computing Machinery, 2022, pp. 2405–2408. ISBN: 9781450392495. DOI: 10.1145/3514221.3520163. URL: https://doi.org/10.1145/3514221.3520163 (cit. on p. 10).

[16] Sander Schulhoff et al. *The Prompt Report: A Systematic Survey of Prompting Techniques*. 2024. arXiv: 2406.06608 [cs.CL]. URL: https://arxiv.org/abs/2406.06608 (cit. on p. 10).

[17] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. *Large Language Models are Zero-Shot Reasoners*. 2023. arXiv: 2205.11916 [cs.CL]. URL: https://arxiv.org/abs/2205.11916 (cit. on pp. 10, 27, 32, 33).

[18] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. 2023. arXiv: 2203.11171 [cs.CL]. URL: https://arxiv.org/abs/2203.11171 (cit. on pp. 10, 27, 32, 33).

[19] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter brian, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. «Chain-of-Thought Prompting Elicits Reasoning in Large Language Models». In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 24824–24837. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf (cit. on p. 10).

[20] Dehai Min et al. *Exploring the Impact of Table-to-Text Methods on Augmenting LLM-based Question Answering with Domain Hybrid Data*. 2024. arXiv: 2402.12869 [cs.CL]. URL: https://arxiv.org/abs/2402.12869 (cit. on p. 22).

[21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL]. URL: https://arxiv.org/abs/2106.09685 (cit. on p. 40).

[22] Brian Lester, Rami Al-Rfou, and Noah Constant. *The Power of Scale for Parameter-Efficient Prompt Tuning*. 2021. arXiv: 2104.08691 [cs.CL]. URL: https://arxiv.org/abs/2104.08691 (cit. on p. 40).