ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

COMPREHENSIVE STUDY OF CLINICAL ENTITY EXTRACTION AND CLASSIFICATION USING LARGE LANGUAGE MODELS

Candidate
Michele Faedi

Supervisor
Prof. Paolo Torroni

Co-Supervisors
Dott. Andrea Galassi
Dott. Giulia Grundler
Dott. Vieri Emiliani

Academic year 2022-2023

Session 2^{nd}

To my sister Giulia,

for being my pillar of strength and motivation

in every step of this journey.

Contents

List of Figures

List of Tables

Abstract

Clinical entities are terms used by specialist doctors to address specific biomedical concepts. Nowadays, NLP tasks have received a boost due to the development of large language models that can understand the semantics of a sentence and reason on it. A key feature of large language models is their ability to learn during training and apply this knowledge as needed, a capability crucial for biomedical natural language analysis.

Entity extraction is a task that given an unstructured text, aims to locate and classify the concepts in order to use the retrieved information in a subsequent task. This is a well-known task in the literature known as *Named Entity Recognition (NER)*. The state-of-the-art models perform very well when provided with enough data and when the entities are generic.

We investigate the efficacy of various techniques for NER in the clinical domain, where the amount of available data to train models is limited. In this challenging domain, MAPS S.P.A. developed a rule-based pipeline that extracts concepts from unstructured text. This pipeline uses a "suggester" to produce candidates that will be later filtered and processed to return desired concepts.

The aim of this project is to study the accuracy of the "suggester" in various environments to derive general conclusions that can be adapted to Italian clinical documents.

Our investigation encompasses three distinct methods to tackle the problem: the EntityRecognizer, the SpanCategorizer, and various generative approaches.

Chapter 1

Introduction

1.1 Motivation

This project aims to evaluate various techniques for extraction and categorization of clinical terms in unstructured documents. The purpose of this study is to assist automated systems operating in the biomedical domain by highlighting relevant terms inside a document. Extracting entities from unstructured texts is a necessary step to allow automated systems to draw conclusions or perform logical reasoning in a verifiable manner even in highly specific medical contexts. Assigning automated systems the task of analyzing documents to uncover hidden information can significantly aid medical personnel in their work.

Recently new technologies, and in particular generative AI, allow automatic analysis and basic reasoning over text, audio and images. These technologies open the door to new types of automation that previously were not possible.

One of the main problems with automated systems is that they should not be used as standalone products because they cannot guarantee correct answers, lack verification of reasoning, and have limited usage of structured data. This last issue is particularly relevant in this domain because basing reasoning on

specialized knowledge, such as human anatomy or chemical formulas, is required to produce accurate results.

The biomedical domain requires the usage of background knowledge to correctly perform reasoning. For this reason, it is advisable to create automated systems that do not use AI as the main source of reasoning but instead use a rule-based automated system only to assist humans. A rule-based automated system gives the developer more control over reasoning, thus adding verifiability and explicability, and can make use of AI just as a tool for specific and closed tasks.

The topic of this research is to study the potential of existing techniques for the extraction of biomedical terms in unstructured documents.

1.2 Contents

The content of this document is organized in seven chapters:

- **Chapter 2 - Background**: In this chapter, a comprehensive foundation is established through the discussion of relevant theoretical concepts and existing research. It begins with a detailed exploration of various representations in the field, such as Named Entity Recognition and Token Embedding, and moves on to discuss encoder-only models. The chapter also examines generative models and their application in processing clinical documents, concluding with an overview of the Unified Medical Language System (UMLS)

- **Chapter 3 - Datasets, Metrics, and Scoring**: Here, the focus shifts to the practical aspects of research methodology. The chapter discusses various datasets used in the study, namely MedMentions, EBM NLP, I2B2 2010, and a dataset in Italian called *IT Dataset*. It also explains the metrics employed to measure dataset characteristics like size and density. Additionally, it introduces the scoring function used in the study,

particularly emphasizing its role in supervised learning performance

- **Chapter 4 - Methods and Technologies**: This chapter delves into the specific methods and technologies employed in the research. It provides an in-depth look at spaCy and its components, such as Tok2Vec, EntityRecognizer, and SpanCategorizer. The chapter also discusses encoder-only architectures, mentioning three pre-trained models, and introduces decoder-only models, including techniques like prompt tuning and fine-tuning

- **Chapter 5 - Experimental Setting**: This chapter describes the experimental setup and the various approaches tested. It covers a range of methods, from a naive baseline, used to obtain additional metrics, to state-of-the-art techniques and GPT-based approaches. It also discusses the implications of different techniques in entity extraction and classification, as well as the impact of dataset size scaling

- **Chapter 6 - Discussion of the results**: This chapter presents the findings of the study, focusing on the performance of different methods in entity extraction and classification. It provides a comparative analysis of various techniques and their effectiveness in handling different datasets. The chapter concludes with a summary of key takeaways and insights gained from the experimental results

- **Chapter 7 - Conclusions**: The final chapter synthesizes the entire research, summarizing the major findings and contributions of the project. It discusses the implications of the results and suggests potential areas for future research. The chapter aims to provide a comprehensive closure to the study, encapsulating its significance and impact in the field

Chapter 2

Background

2.1 Well-known techniques

This section reports the main sources to understand well-known techniques for entity extraction and entity classification.

2.1.1 Named Entity Recognition

Named Entity Recognition (NER) is a task in natural language processing that involves identifying and categorizing key information in text, such as names of people, organizations, and locations. A pivotal point in NER research was the CoNLL-2003 shared task, which standardized the evaluation of NER systems as detailed in the paper by Tjong Kim Sang and De Meulder [30]. The advent of deep learning further refined NER, with neural network architectures, especially *LSTM* combined with *CRFs*, as shown by Lample et al. [16], significantly enhancing performance. The introduction of transformer models like *BERT*, by Devlin et al. [11], marked a revolution in NER by leveraging pre-training on large text corpora for superior context understanding.

2.1.2 Token Embedding

Token embedding, such as *Glove* by Pennington et al.[26] or *Word2Vec* by Mikolov et al.[21], create static vectors representations for each token in the vocabulary using unstructured documents for training. These techniques do not allow for unknown words or new terms to be used because they cannot represent words not seen during training. For this reason, techniques that aim to solve this problem have better performances. Peters et al.[27] used recurrent Neural Networks, and in particular *LSTM*, to compute embedding for tokens using the entire context, enhancing generalization and performance for out-of-vocabulary words.

Vaswani et al.[34] created a new architecture called *Transformer* used for Machine Translation. After its successful applications in many tasks, Devlin et al.[11] used the encoder part to compute automatic token embeddings for NER tasks, making encoder-only, or *BERT-like*, architectures the de facto standard for NER.

2.1.3 Encoder-only models

After the successful applications of BERT, subsequent research has been conducted to increase the performance and, possibly, to understand the main factors contributing to its effectiveness. Y Liu et al.[19] showed that tuning the training hyperparameters, such as size of the training dataset, different tokenizer algorithm, training time, etc. allows for additional performance increases, leading to the release of a new model called *RoBERTa*. Subsequent BERT based models, such as *AlBERT* by Lan et al.[17] or *ELECTRA* by Clark et al.[9], have also been developed using different strategies to increase the embedding expressiveness.

RoBERTa and subsequent BERT based models are trained using datasets with general knowledge. To enhance performance in a closed domain, it is possible to fine-tune the models using domain-specific datasets to let the

model learn knowledge on the subject. For example, *BioBERT* by Lee et al.[18] is a model trained on biomedical datasets comprising a total of 18 billion tokens.

2.2 Generative models

Since the inception of Natural Language Processing the long-term goal has been to build an intelligent dialog agent. Weston[36] investigates techniques for dialog-based language learning, focusing on creating models that mimic "how human learn, where language is both learned by, and used for, communication".

McCann et al. [20] created a benchmark (*decaNLP*) and a model (*MQAN*) for multitask learning and question answering. They frame all tasks as question answering, giving the model more flexibility and, using the context in the question, zero-shot capability on new and unseen tasks.

Radford et al. [28], with *GPT-1*, a decoder-only *Transformer*, improved the generalization ability by pre-training the model using lots of text in an unsupervised setting, then fine-tuning it with supervised learning. Radford et al. [29] created *GPT-2*[29] increasing the dataset for pre-training and showed that it performs surprisingly well in zero-shot setting on a large amount of tasks without the need for supervised learning.

Brown et al. [6] scaled *GPT-2* to 175 billion of parameters, creating *GPT-3*, that "*shows strong performance on many NLP tasks and benchmarks in the zero-shot, one-shot, and few-shot settings, in some cases nearly matching the performance of state-of-the-art fine-tuned systems*". *GPT-3* is trained with a subset of Common Crawl [1] with the objective of generating the next most likely token given the previous. The original task referenced by Weston[36] is to produce the most correct answer and this has some ethical implication:

[1] Dataset containing nearly unlimited text from the web. There are various versions, the last one's size (as in 2023) is 390 TiB.

generate formal, truthful, nontoxic, and non harmful responses.

Ouyang et al. [24] created a new version of *GPT* called *InstructGPT* or *GPT-3.5*, showing that an aligned model is better at human evaluation compared to 100x misaligned models. Another desired feature of *InstructGPT* is the ability to follow instructions, a precise description of tasks to follow. This feature is obtained with *Instruction Tuning*, i.e., fine-tuning the model with prompts to produce the desired response. Zhang et al. [39] performed a survey on techniques for *Instruction Tuning*.

OpenAI[23] studied the scalability of their architecture and released another model called *GPT-4*, but the implementation details are not public. From the paper the results that matter to this project is that *GPT-4* outperforms the previous version and has better performance in Italian than *GPT-3.5* in English.

After the release of *GPT-3* other research teams (Chinchilla by Hoffman et al. [14], PaLM by Chowdhery et al. [8]) created their own version of decoder-only *Transformer*. The most relevant research for this project is by Touvron et al. [31], by MetaAI, where they focus on smaller models trained on more data. They released four versions (7B, 13B, 30B, 65B of parameters) with a noncommercial licence on the code and weights. *LLaMA 13B* outperforms *GPT-3* on most benchmarks despite being ten times smaller. After *LLaMA*, the same team released *LLaMA 2* by Touvron et al. [31], by MetaAI, with chat alignment using Reinforcement Learning from Human Feedback (RLHF).

LLaMA demonstrated that it is possible to have relative small models that match or outperform big models like *GPT-3.5*, especially if fine-tuned on custom datasets. Zhou et al. [40] released a model called *UniNER* (Universal NER) based on *LLaMA* and instruction tuned specifically for NER across various domains (General, Biomeds, Clinics, STEM, Programming, Social media, Law, Finance, Transport). *UniNER* outperformed other instruction tuned models, like *ChatGPT* and *Vicuna*[7].

Chain-of-Thought (CoT) prompting is a technique that can be used to improve the reasoning capabilities of large language models (LLMs). It was introduced by Wei et al. [35]. By providing LLMs with examples of how to chain multiple pieces of information together to solve a problem, CoT prompting can help LLMs learn to reason more effectively. This can be particularly useful for tasks that require multiple steps or require the LLM to consider multiple factors.

2.3 Clinical documents

In this study we focus on applications within the clinical domain, specifically concerning human medicine. Clinical documents, which are typically authored by and intended for medical professionals, pose significant challenges for analysis.

An automated system designed for processing these documents must address several key issues:

- The scarcity of databases and established benchmarks for training
- The syntactic peculiarities unique to this domain
- The necessity for generalization capabilities to ensure satisfactory performance across various and unseen domains
- The prevalence of unstructured text, poor grammar, and the use of abbreviations, synonyms, and overlapping entities
- Lack of knowledge and training data in languages other than English

Fig 2.1 and 2.2 illustrate the diversity of entities and domains encountered in this field. The first figure presents an abstract from a scientific paper annotated with biomedical entity classes, while the second displays an Italian radiology report, further exemplifying the linguistic and contextual variety.

In Chapter 3 there will be a complete evaluation of the available datasets, showing the pros and cons and the motivation of the chosen datasets.

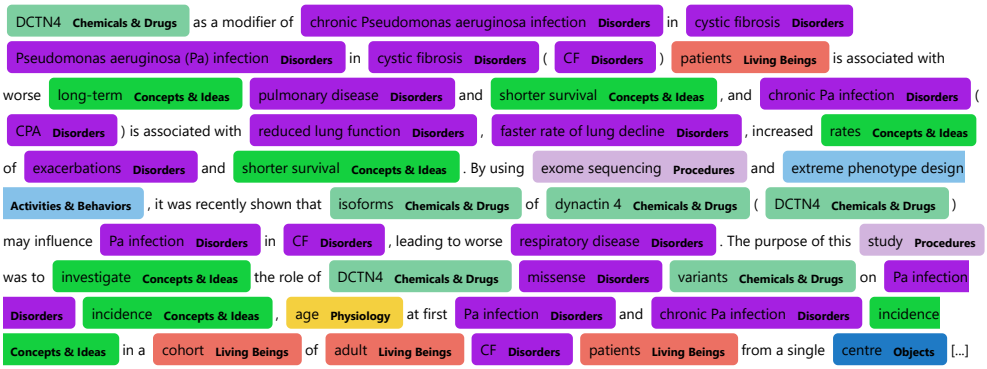

Figure 2.1: Example of document from MedMentions

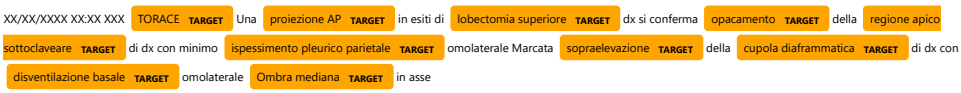

Figure 2.2: Example of document from the Italian dataset

2.4 Unified Medical Language System (UMLS)

Although UMLS is not the main focus of this project, an introduction is required to understand *MedMentions*, one of the four datasets used in this project.

We begin by delving into the concept of ontologies, fundamental elements in the domain of knowledge representation. The term 'ontologies', borrowed from philosophy, refers to the formal representation of a set of concepts within a domain and the relationships among those concepts. Ontologies are essential for the sharing and reuse of knowledge bases, enabling different systems to 'understand' and process information based on the same knowledge base. In the biomedical domain there are various thesauri for different applications, which encompass many different concepts. For example, the Italian National Health Service uses *ICD9CM* (International Classification of Diseases, Ninth Revision, Clinical Modification), a thesaurus containing many

types of concepts (i.e., Diseases, Diagnostic procedure, Therapeutic Procedure, etc.). Even if *ICD9CM* is widely used, it is not sufficient for many use cases. For this reason, it became necessary to unify the biomedical thesauri into one meta-thesaurus.

UMLS[4], or Unified Medical Language System, is the main knowledge base used for many applications. It merges concepts from many thesauri maintaining or creating new relationships. Many concepts with the same meaning are merged keeping all the attributes, and thus giving different labels to the same concept. Each concept is identified with a unique code called *CUI* (Concept Unique Identifier).

For example, the concept "`Femoral Neck Fractures`" (identified with the CUI `C0015806`) has many labels: "`Femoral Neck Fractures`", "`Fractured femoral neck`", "`fractura de cuello de fémur`", "`Frattura del collo del femore`", etc.

UMLS has some drawbacks given by the immense volume of data:

- many concepts are redundant
- many thesauri have language adaptation but are not exhaustive
- scarcity of Italian concepts
- many relationships make it difficult to navigate the graph
- different levels of precision across different thesauri
- non-standard relationship rules

Fig. 2.3 shows a bar plot of UMLS concepts grouped by languages. English is the main source while the contribute of other languages is almost negligible. In Italian there are only 262k concepts that correspond to the 1.67% of the total amount, but a smart system may use English as the main graph and search for the nearest and correct concept in the desired language.

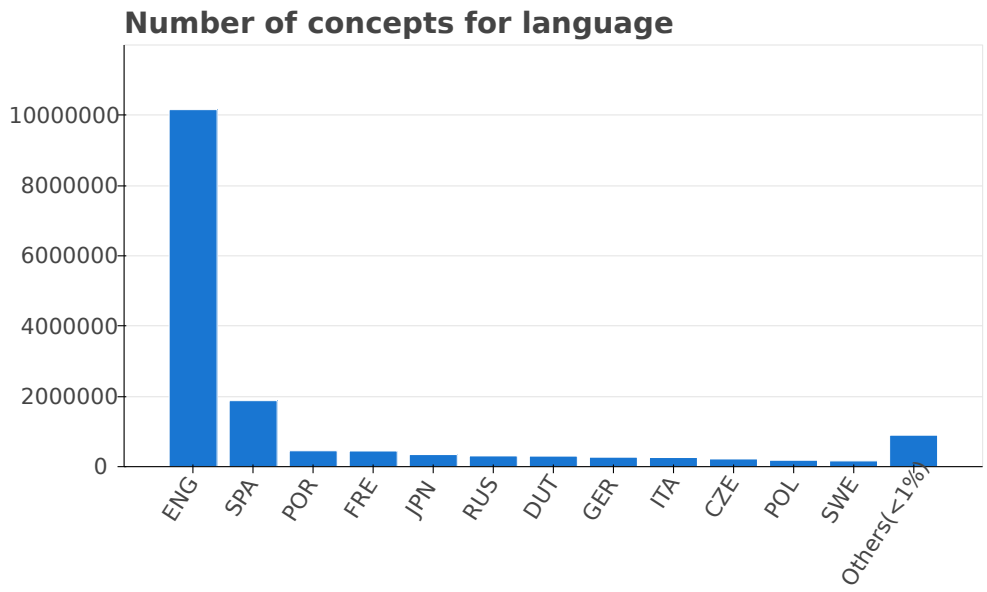

Figure 2.3: Concepts of UMLS grouped by languages.

In Table 2.4 there is a list of Semantic Groups with the number of concepts associated with each of them.

Even though UMLS has many problems, it is considered the best meta-thesaurus because it is the most complete one and can be used to solve some thesaurus specific problems.

Semantic Group	Concepts	%
Living Beings	1951563	42.61%
Chemicals & Drugs	929935	14.24%
Disorders	642428	8.61%
Procedures	435413	5.37%
Physiology	169310	1.98%
Anatomy	153633	1.76%
Concepts & Ideas	92476	1.04%
Genes & Molecular Sequences	80501	0.90%
Devices	68811	0.76%
Objects	24770	0.27%
Phenomena	15035	0.16%
Activities & Behaviors	5741	0.06%
Geographic Areas	4633	0.05%
Organizations	4050	0.04%
Occupations	2074	0.02%

Table 2.1: Concept of UMLS grouped by semantic groups

Chapter 3

Datasets, Metrics and Scoring

3.1 Datasets

In many research fields the datasets for training and testing are closed source, due to privacy reasons and because it is usually costly to produce good data. Significant improvement is often achievable by increasing the quality of the dataset. Hence, the datasets are usually kept private.

These challenges are particularly present in the biomedical domain. Datasets can include patients' clinical information, and anonymizing this data is not a trivial task. Under the General Data Protection Regulation (GDPR), Article 4(1)[2], the definition of personal data is as follows:

> *'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;*

This clause indicates that even indirect identification qualifies as personal

data and must be treated accordingly. For this reason, due to the aforementioned privacy concerns, the availability of open-source resources is limited.

Another problem with clinical datasets is that they are costly to produce since they require specialized personnel from different disciplines to annotate the data correctly and thoroughly, especially with UMLS. For this reason, the clinical datasets in this project are not annotated with UMLS concepts, but with simpler classes, eliminating the need for specialist input to ensure quality.

To address these issues, we expanded our research beyond clinical data to include other biomedical sources, and we investigated the correlations between various dataset types and qualities and their impact on the observed results.

For this project we evaluated four different datasets:

1. *MedMentions*: 4392 biomedical paper abstracts annotated with UMLS concepts.
2. *EBM NLP*: 4,993 paper abstracts annotated with three classes.
3. *I2B2 2010*: 169 clinical documents annotated with three classes.
4. *IT-Target Guesser*: 468 clinical reports from radiology department. Uses only one class (`TARGET`).

3.1.1 MedMentions

MedMentions[22] is a dataset composed of 4392 biomedical papers, selected randomly from the *PubMed* database in 2016. All papers are in English.

PubMed is a search engine, operated by the United States National Library of Medicine (NLM) at the National Institutes of Health, which uses the *MEDLINE* database to find relevant literature for healthcare professionals, researchers, and scholars.

For the annotation process, a group of experts with a background in the biomedical field was hired to annotate all UMLS entities present in the titles

and in the abstracts of the selected papers.

The quality of *MedMentions* is assessed by measuring the agreement between two groups of annotators, which shows an estimated precision of 97.3%.

In *MedMentions*, given the presence of UMLS, the high precision and frequency of entities, we investigate the influence of a high-quality dataset by comparing the results on a truncated version. This truncation involves omitting entities associated to concepts that do not have an Italian or Spanish label.

Table A.2 presents a list of classes with the count of entities from the complete and the truncated versions.

3.1.2 EBM NLP

EBM NLP is a dataset composed of 4,993 paper abstracts. This dataset contains sixteen classes organized in three macro groups, namely `Participants`, `Interventions` and `Outcomes`.

The training labels are created using AMT workers[1], and then aggregated to reduce the noise. The test set is annotated by medical professionals to have a better reference.

This dataset contains some overlap between entities. For our studies we decided to remove the overlap by considering only the longer entity. This decision facilitates a clearer comparison of algorithms that do not permit entity overlap.

The task assigned to AMT workers involves extracting and classifying data into seventeen distinct categories, as detailed in Table 3.1.2.

We decided to use only the macro groups in order to have a dataset that bridges the gap between *MedMentions* and *I2B2 2010*, retaining paper abstracts but focusing on three classes similar to those in *I2B2 2010*.

Another advantage of this approach is the resolution of certain class imbalances. Table A.2 report the class counts with the macro groups.

Participants	Interventions	Outcomes
Age	Surgical	Physical
Sex	Physical	Pain
Sample size	Drug	Mortality
Condition	Educational	Adverse effects
	Psychological	Mental
	Other	Other
	Control	

Table 3.1: Class of entities of EBM-NLP

3.1.3 I2B2 2010

I2B2 2010[32] is a multipurpose dataset (originally created by NIH-funded National Center for Biomedical Computing (NCBC)) known as i2b2, or Informatics for Integrating Biology and the Bedside. It was created for a challenge with three main objectives: 1. Extraction of medical problems, tests, and treatments. 2. Classification of assertions made on medical problems. 3. Relations between classes.

The source data includes discharge summaries from MIMIC II database, provided by Beth Israel Deaconess Medical Center. MIMIC II is a database, containing anonymized clinical data from a medical center, collected between 2001 and 2008.

The dataset is licensed and must be used for research only. The licence also prohibits its use with external tools such as ChatGPT, GPT 3.5 or GPT 4.

The documents are quite long, but we decided against any form of data augmentation, such as splitting them into sections, to better assess the effectiveness of the methods used on long documents.

The dataset contains only three classes: `Problem`, `Test`, and `Treatment`.

3.1.4 IT dataset

The IT dataset, created for internal use by MAPS S.P.A., comprises 468 anonymized radiology reports.

The dataset was created by manually annotating fifty documents and then using a trained spaCy's span categorizer to annotate and correct the mistakes in the remaining 418 documents.

The entities are only extracted; that is, the sole class associated with these entities is `TARGET`.

3.2 Metrics

The project focuses on studying available techniques for entity extraction and classification in four different biomedical datasets. It also aims to identify metrics and draw conclusions that are applicable to other datasets. As a result, the study includes a review of available metrics for evaluating dataset quality.

The main metrics used to evaluate the datasets are reported in the following sections.

3.2.1 Size of the datasets

The size of the dataset is quantified in terms of the number of documents. Table 3.2.1 contains the number of documents available for the four different datasets, differentiating between training and test sets.

Generally, having more data increases the performance; however, beyond a certain threshold, the improvement does not justify the cost of acquiring additional data. For this reason, we investigated how scalability is affected by varying the size of the datasets, by adjusting the size of the training sets and measuring the models' performance outcomes.

Name	Training	Validation
MedMentions	2634	878
EBM NLP	4801	191
I2B2 2010	136	33
IT dataset	375	93

Table 3.2: Number of documents available for the datasets

3.2.2 Density

This metric is computed by looking at the number of entities over the number of documents. Table 3.2.2 reports the average densities, with the standard deviation, of the datasets; it also includes the minimum and maximum value found in the dataset.

This value indicates the average density of a document. Additionally, we investigated the effectiveness of the techniques used, given the varying densities of annotations. A dataset with low density may be more difficult for the model to learn from due to the scarcity of data in the training batch. On the other hand, a dataset with a high density of annotations may pose a greater learning challenge for the model due to the task's complexity. This effect will be assessed by comparing *MedMentions* with two different versions.

Name	Avergage	Min	Max
MedMentions	0.29 ($\pm$0.05)	0.19	0.41
MedMentions Truncated	0.20 ($\pm$0.04)	0.11	0.30
EBM NLP	0.06 ($\pm$0.02)	0.02	0.15
I2B2 2010	0.08 ($\pm$0.03)	0.02	1.12
IT dataset	0.21 ($\pm$0.03)	0.14	0.29

Table 3.3: Density of the datasets

3.2.3 Entities length

The number of tokens the entities are composed of may represent a challenge for sequential techniques. Table 3.2.3 reports the average number of tokens, with the standard deviation, of the entities.

The datasets are simplified by cutting the entities that exceed a token threshold. In any case, these cuts remove only a small portion of the annotated entities (less than 0.3%)

The complexity of a sentence is measured by counting the number of tokens it contains, while the complexity of a document is assessed by the number of sentences it includes.

The two metrics tend to vary significantly across the datasets. In particular, *I2B2 2010* has long sentences and lengthy documents, which may affect the results. We decided not to split the documents into sentences or sections. This approach allows us to measure the effectiveness of the proposed methods across differently structured datasets without the need for additional augmentation or preprocessing.

Name	Avergage	Min	Max
MedMentions	1.30 (± 0.54)	1	3
EBM NLP	2.56 (± 1.72)	1	8
I2B2 2010	2.00 (± 1.18)	1	6
IT dataset	1.65 (± 0.68)	1	4

Table 3.4: Size of the entities

3.3 Score function

Comparing different methods requires a unified algorithm capable of computing scores independently of the techniques employed.

In entity extraction, metrics based solely on entities are not considered insufficient for effective comparison. Therefore, additional metrics have been

developed to address imprecise extraction, focusing on examining the entities' tokens. Comparing scores both by entities and by tokens provides insights into the precision with which the method identifies and centers around the entities.

3.3.1 Supervised learning performance

In supervised learning, a confusion matrix is used to measure the performance of an algorithm. The confusion matrix represents the discrepancies between the actual outcomes (ground truth) and the algorithm's predictions. For each prediction, the confusion matrix is updated by incrementing the value in the cell corresponding to the true class index (row) and the predicted class index (column). The ideal model has a confusion matrix where all values greater than zero are located on the main diagonal.

In the case of binary classification, the confusion matrix is two by two and has four values called:

- **True Positives** (TP): instances correctly predicted as positive
- **False Positives** (FP): instances incorrectly predicted as positive
- **True Negatives** (TN): instances correctly predicted as negative
- **False Negatives** (FN): instances incorrectly predicted as negative

From the confusion matrix, it is possible to extract the *Precision*, *Recall*, and *F1* scores. The precision measures how much we can trust the algorithm if it predicts true, while the recall indicates how good the algorithm is at identifying the positive cases. Precision and recall are often in a trade-off relationship; improving one can led to a decrease in the other. The F1 score, the harmonic mean of precision and recall, provides a single metric that balances both aspects.

Equation 3.1 presents the formulas for Precision, Recall, and F1

$$\begin{aligned} \textbf{Precision} &= \frac{\texttt{TP}}{\texttt{TP} + \texttt{FP}} \\ \textbf{Recall} &= \frac{\texttt{TP}}{\texttt{TP} + \texttt{FN}} \\ \textbf{F1} &= \frac{2\texttt{TP}}{2\texttt{TP} + \texttt{FP} + \texttt{FN}} \end{aligned} \tag{3.1}$$

These formulas can be extended to multi-class settings by computing the metrics for each class and then aggregating the results. There are two types of aggregation:

- **Micro**: a weighted average based on the counts of true labels for each class, assigning less importance to less represented classes
- **Macro**: a uniformly weighted average that treats all classes with equal importance, regardless of their representation

Chapter 4

Methods and technologies

4.1 spaCy

spaCy is an open-source software library for NLP, written in Python. It is specifically designed for production use, offering the most efficient implementation of common algorithms. It is designed with extensibility in mind, allowing for the addition of custom tokenization rules, custom components and models, and integration with deep learning frameworks like TensorFlow or PyTorch. It is easily integrable with HuggingFace's Transformers, making it compatible with a large number of pre-trained models.

For our experiments, we mostly use spaCy's functionalities to standardize the process and offer results independent of our ability to train and evaluate the models. spaCy can be used for almost all use cases, and it comes with built-in components for various NLP tasks. The majority of these tools can be trained for specific use-cases and then adapted to different languages or contexts.

Processing natural language usually involves several steps. For example, dependency parsing requires POS tagging, and POS tagging requires a vector representation of the tokens. For this reason, spaCy manages the cascade of processes into a single object called *pipeline*.

Pipeline

The pipeline is composed of two main parts: the tokenizer, and the list of components. The tokenizer is the first essential step of the pipeline; it separates the document into tokens, i.e., groups of characters that represent a semantic unit.

SpaCy implements a custom word-based tokenizer. "Word-based" means that it tokenizes the document into words and punctuation. This approach is human-manageable and useful for rule-based systems. However, it does have some drawbacks, including a large vocabulary and an inability to process out-of-vocabulary terms

The tokenization takes a text as input and returns a *Doc* object. The list of components is applied in a cascade, where each component takes in input a Doc, saves the results, and outputs the same Doc object.

Doc

The Doc object represents a tokenized document. Apart from the list of tokens, it also contains other data:

- **Span**: a contiguous group of tokens with an assigned label, used to store entities that have been extracted and classified
- **SpanGroup**: organizes multiple spans under a single name
- **Entities**: represent a unique, non-overlapping group of spans, forming a special SpanGroup specifically for NER components
- **Extensions**: additional data can be added to the previous categories, including tokens. For example, a variable can be added to the Span to store the CUI of entities.

4.1.1 Tok2Vec

Many components require a vector representation of tokens to process the text effectively. SpaCy has implemented an embedding function known as *Tok2Vec* [15]. It was implemented prior to the Transformer architecture by Vaswani et al.[34], necessitating the development of sophisticated logic to achieve good results.

Vectorization is achieved in two steps:

1. **Embed**: convert each token into a context-independent vector
2. **Encode**: provide context to each token vector.

Embed

The algorithm used is called *MultiHashEmbed*; it relies on another algorithm called *HashEmbed* and is designed to solve the problem of out-of-vocabulary terms.

HashEmbed The HashEmbed algorithm operates through a series of steps for each token:

1. the token is hashed multiple times with different seeds
2. each hash is modularized with the size of the embedding table to generate indexes
3. the indexes are mapped to 96-dimension vectors using the embedding table
4. the vectors are summed to produce a singular vector for each token.

This algorithm has several advantages:

- flexibility in the size of the embedding table
- the ability to encode out-of-vocabulary tokens

- a potentially smaller embedding table compared to traditional, sequentially-numbered tables
- the elimination of the need to store a vocabulary mapping from token to vector due to the index generation through token hashing

Instead of hashing the same token with different seeds, MultiHashEmbed uses four different variants of the token — like prefix, suffix, shape, and original — and four seeds for each variant. Each variant uses a different embedding table with different sizes. The process for embedding a word is illustrated in Image 4.1.

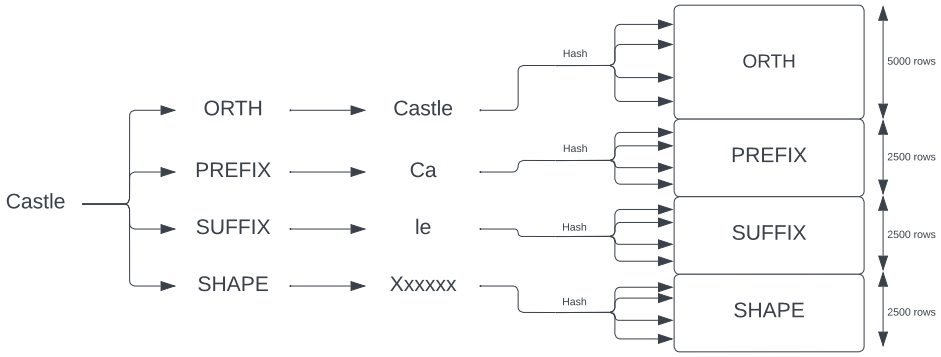

Figure 4.1: Example of MultiHashEmbed using four seed for four variants

Encode

The encoding process provides context to the tokens, i.e., it adds information about the entire document to each token vector. SpaCy's default encoding model is a convolutional neural network with residual connections, layers normalization and maxout activation. For completeness, these techniques are discussed hereafter; however, for simplicity, many concepts are not explored in depth.

Residual connections Introduced by K. He et al.[13], residual connections have enhanced the performance of convolutional neural networks. Research

has shown that residual connections enhance the performance of convolutional neural networks, providing more stability during training, especially in deep architectures.

With residual connections, the blocks (i.e., groups of three layers of convolutions) can be considered as perturbations of the input. In our case the blocks output can be seen as context information being added to the vector.

Layer normalization Layer Normalization is a technique that standardizes the inputs across features within a layer by normalizing the inputs for each sample independently. This is achieved using scaling and shifting parameters, making it effective for recurrent and Transformer networks, regardless of batch size.

Maxout activation The maxout activation function, as introduced by Goodfellow et al.[12], operates by splitting the inputs into groups. It applies a learnable linear transformation to each group and then takes the maximum value across the transformed outputs of each group.

4.1.2 Entity Recognizer

EntityRecognizer is a spaCy component that performs sequence labeling for NER. Sequence labeling is a type of task with the objective of assigning a label to each element in a sequence. Entity extraction can be accomplished by using three classes: (B)egin, (I)nside and (O)utside. The label (B) indicates the beginning of an entity and also marks the end of another entity; the label (I), in conjunction with (B) or another (I), indicates that the previously started entity continues; the label (O) indicates that the token does not belong to any entity and also marks the end of the previous entity.

For multi-class sequence labeling, distinct (B)egin and (I)nside labels are employed for each class, allowing for precise categorization of entities into different classes.

Figure 4.2 shows an example of sequence labeling for entity extraction and classification.

Why	do	Dutch	people	use	dietary	supplements?
O	O	B	I	O	B	I
O	O	B-Persons	I-Persons	O	B-Objects	I-Objects

Figure 4.2: Example of Sequence labeling for Entity Extraction and Entity Classification. In this example the entities are “Dutch people” and “dietary supplements”

SpaCy uses another standard, called *BILUO*, for sequence labeling that adds two classes, namely (L)ast and (U)nit. (B) indicates the beginning of an entity composed of two or more tokens, (I) indicates the inside of an entity composed of three or more tokens, (L) indicates the end of an entity of two or more tokens, and (U) indicates an entity composed of only one token.

With this technique, it is not possible to extract entities with overlaps, and this method may have some difficulties in extracting long entities with lots of tokens. Unlike other techniques, sequence labeling cannot be used solely for entity classification, as it would also extract the entities.

The focus of optimization is to increase the accuracy of entire entities. Therefore, if there is low annotation agreement on the boundaries of the entities, this component will likely perform poorly.

4.1.3 Span Categorizer

This component is designed to address the limitations of the entity recognizer, particularly its inability to allow overlaps.

The standard NER task does not include overlapping entities. As explained in Chapter 3, Section 3.2.3, we removed entities that overlap from our datasets. However, this technique has some important advantages:

- Two distinct steps: one for entity extraction and one for entity classification

- The problem of entity lengths is mitigated by a pooling strategy
- Gives a confidence score to the generated entities
- It is less sensitive to the boundaries of entities

This component is composed of two distinct sub-components called *SpanSuggester* and *EntityLabeler*.

SpanSuggester

This component performs entity extraction for all entities independent of their classes. While SpaCy employs several techniques, for our tests we utilized only two: the *Ngram Suggester* and the *SpanFinder*.

Ngram suggester The Ngram Suggester sends all spans with specific lengths to the EntityLabeler, which will later classify these spans as entities of a particular class or discard them. Although this mechanism is simple, it is quite effective, mainly due to the classifier's ability to discriminate between good and bad candidates.

The main problem with this technique is that if the dataset has long entities, it generates a lot of candidates, thereby increasing the time required to process the document.

SpanFinder This component is designed to use a more sophisticated technique than the Ngram suggester.

The SpanFinder is similar to the entity recognizer and uses only three classes: (B)egin, (L)ast, and (O)utside. It marks as candidates all the possible spans that start and end in correspondence with the marked tokens.

This technique generates fewer candidates with more precision, offering the advantage of making the EntityLabeler significantly faster during both training and testing by providing only a fraction of the candidates compared to the Ngram suggester.

EntityLabeler

This component is composed of two parts: the reducer and the scorer. The default reducer mixes the average and max pool, then sends them to a linear layer, which adjusts the vector's length. The scorer is a multi layer perceptron with a softmax that predicts whether a candidate is a discard, or it is an entity that is classified into one of the predefined classes.

4.2 Encoder-only architecture

This section explains the Transformer architecture, focusing on the encoder-only, or BERT-like, architecture. The decoder-only architecture is explained in Section 4.4, dedicated to generative approaches.

Primarily used for automatic token embedding, the BERT model serves as an alternative to the Tok2Vec model, offering advanced contextual capabilities.

Similar to Tok2Vec, the encoder-only architecture includes two sections: an embedding section, which converts each token into a vector representation, and an encoding section, which adds context information to the token vectors. Figure 4.3 provides a diagram of the architecture.

4.2.1 Embedding

BERT utilizes an embedding matrix, also known as an embedding table, which stores the vector representation of each token in the vocabulary as rows. The matrix is initially randomized and subsequently refined through learning. The size of the vector scales from 128 in the 'tiny' version to 2048 in the 'xlarge' version.

By construction, the transformer architecture is not a sequence model. To enable it to process sequential data, a positional embedding is added, providing information about the position of each token in the sequence. Positional

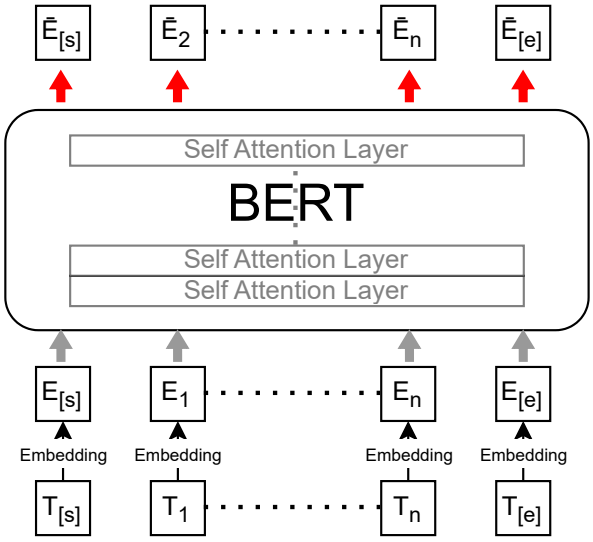

Figure 4.3: BERT architecture

embeddings for each position are learned during model training. Vaswani et al.[34] noted that the quality difference between using fixed or learned positional embeddings is minimal. Nonetheless, BERT and subsequent models employ learned positional embeddings.

4.2.2 Encoding

Encoding is achieved through repeated application of the attention mechanism. The Transformer architectures employ an attention mechanism known as *Multi-Head Attention*. It can be applied to the same input vectors, known as self-attention, or jointly with different inputs, known as cross-attention. Figure 4.4 and 4.5 illustrate the operational flow of Multi-Head Attention [1].

[1] Images from the paper 'Attention is All You Need' by Vaswani et al. [34]

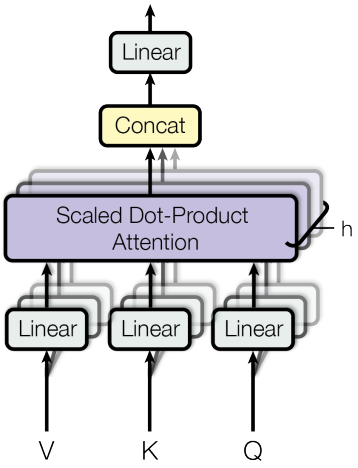

Figure 4.4: Multi-Head Attention mechanism. The input vectors undergo parallel processing in several Dot-product attention layers

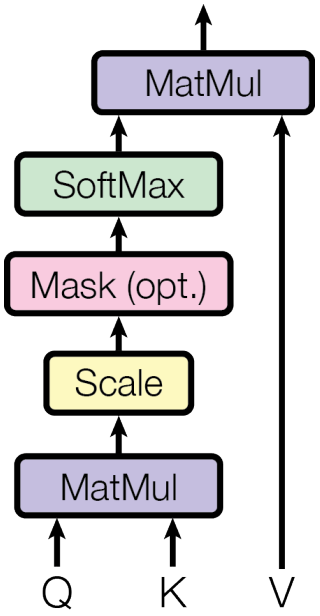

Figure 4.5: Scaled Dot-product Attention layer. In self-attention, the matrices K (Key), V (Value), and Q (Query) are derived from the same input vectors. For cross-attention, K and V matrices are sourced from different inputs

4.3 Encoder-only models

For our experiments, we tested five different encoder-only, or BERT-like, models for automatic token embeddings, namely `RoBERTa-base`, `MedBERT`, `Coder`, `UmBERTo`, and `xml-RoBERta-base`. The models have similar number of parameters, which makes them comparable and trainable with the same hardware resources.

4.3.1 RoBERTa-base

RoBERTa is one of the most famous encoder-only models freely available. It was published in 2019 by Y. Liu et al. [19], in a replication study that performed hyperparameter tuning in order to study the potential of encoder-only models. RoBERTa gives us insights on the extent to which a model can

be effective without deep medical knowledge.

RoBERTa was trained with particular care in evaluating all the hyperparameters, and it has some important improvements compared to other BERT variants:

- **Improved Handling of Longer Sequences**: RoBERTa has been trained on longer sequences and can process long texts more effectively
- **Greater efficiency in zero-shot and few-shot learning**: RoBERTa can generalize better from limited data, which is crucial for tasks where annotated data is scarce
- **Better Handling of Ambiguity in Language**: RoBERTa's ability to understand and interpret ambiguous language is enhanced, making it more effective in tasks where the meaning of text depends heavily on context
- **Improved Robustness to Noisy Data**: RoBERTa exhibits enhanced robustness to noisy or unstructured data, thanks to its training on a diverse and extensive corpus

RoBERTa comes with four variations, for our experiments we used RoBERTA-base, with the following hyperparameters:

- **Hidden Size**: 768
- **Intermediate Size**: 3072
- **Attention Heads**: 12
- **Layers**: 12
- **Max Sequence Length**: 512
- **Batch Size**: between 32 and 128
- **Total Parameters**: Ĩ25 million

4.3.2 MedBERT

MedBERT by Vasantharajan et al. [33] is a pre-trained model for biomedical NER. It is initialized with *BIO_ClinicalBERT*, another BERT-like model trained with additional data in the medical domain, and fine-tuned with four additional datasets in the biomedical domain:

- **N2C2 corpus**: articles in the biomedical domain, from the N2C2 2018 and 2022 challenges
- **BioNLP**: articles released for the BioNLP project. The articles vary around bio-molecular sciences (molecular biology, DNA modifications, habitats of bacteria mentioned, etc.)
- **Colorado Richly Annotated Full Text(CRAFT)**:67 biomedical articles from PubMed that cover a wide range of the biomedical domain
- **Wikipedia Corpus**: a curated dataset of selected biomedical-related articles from Wikipedia

MedBERT is used to test how much prior knowledge in the biomedical domain affects performance.

4.3.3 Coder

CODER by Yuan et al. [38], is a model designed to cluster terms that represent the same UMLS concepts. This is a different approach compared to other BERT-like models since it is not trained to produce text.

Contrastive learning is a technique that involves the learning of embedding vector, and bring positive samples closer and moving negatives ones further away in a multi dimensional space. The samples are taken from UMLS: the concept with a relationship that indicates closeness is considered positive and the concepts that do not have a relationship with the anchor are considered negatives. CODER has a good representation of UMLS and can be used for

semantic search (by encoding a string and searching for the closest vector from UMLS).

CODER is used to test the difference between EntityRecognizer and SpanCategorizer. The hypothesis is that SpanCategorizer, when used with CODER, will outperform sequential labeling techniques because it is trained on entire entities.

4.3.4 UmBERTo

UmBERTo, developed by Parisi et al.[25], is a model derived from RoBERTa-base. It includes several enhancements and is specifically trained on Italian documents.

There exist two versions of this model, each trained with a different corpus:

- **wikipedia-uncased**: trained with a small corpus (7 GB) composed of Wikipedia articles
- **commoncrawl-cased**: trained on a larger dataset (69 GB), encompassing a more diverse range of texts

4.3.5 xml-RoBERTa-base

The xml-RoBERTa-base, a multilingual version of RoBERTa-base developed by Conneau et al. [10], is trained on 2.5 TB of data encompassing one hundred languages.

4.4 Decoder-only models

Decoder-only models, such as GPT-like models, can autoregressively compute the next token in a sequence. These models are used to generate text and when instruction-tuned, they serve as versatile, general-purpose tools.

The Decoder-only's architecture is similar to encoder-only with the difference that the attention is not bidirectional.

Using large language models (LLMs) for entity extraction and classification offers an alternative to traditional methods. These models excel in memorizing and retrieving knowledge. Consequently, we investigated their efficacy in analyzing biomedical documents. Similar to many language-related tasks, both entity extraction and classification can be approached as question-answering problems. This is achieved by providing a generative system with the relevant documents and task descriptions.

4.4.1 Prompt tuning

Entity extraction and classification tasks can be approached in various ways, each necessitating a unique task description. Research focusing on these varying approaches is known as prompt engineering.

Since GPT-like models are general purpose, it is necessary to correctly submit the task in order to maximize the results while at the same time reducing, if possible, the "token consumption". Token consumption refers to the number of tokens necessary to solve the task. Reducing the number of tokens necessary for a task has two advantages: it reduces costs and helps the model by not spreading information across the entire context.

One of the main techniques to enhance prompt performance is to include examples in the prompt. This technique is called *few-shot learning*.

The context size limit is more relevant for decoder-only models because a larger context size allows the use of more examples, and longer document in the prompt. OpenAI models have a context size limit ranging from 4,191 to 128,000 tokens.

4.4.2 Fine-tuning

Large Language Models can be retrained on new data. This process is called fine-tuning and is used to adapt the model to a downstream task.

In the paper by Bhatia et al. [3], it is shown that prompt engineering, which

focuses only on task representation, does not change the reasoning abilities of LLMs. The paper also demonstrates that the performance gap is due to their inability to perform simple probabilistic reasoning tasks. This limitation can be solved through fine-tuning, which involves training the model with a specialized and smaller dataset.

OpenAI recently released an API that gives access to fine-tuning utilities for training GPT-3.5. The API requires a dataset containing a list of dialogues between the model and the user. It is also possible to set the number of epochs (number of times each sample is used to train the model) in order to use small datasets more effectively.

Chapter 5

Experimental setting

In this chapter, we report all the experiments designed to study the problem as completely as possible. The focus of this research is to discover which techniques give good results and the reasons behind their success across four distinct datasets, each representing different environments. This chapter begins by introducing a straightforward technique aimed at enhancing our understanding of the datasets. It then proceeds to detail the experiments conducted to compare various available technique.

5.1 Naive Baseline

Many characteristics of the datasets determine the complexities of the tasks. To measure each dataset's complexity, we created a simple model and compared the results across the datasets.

This model is implemented using spaCy Matcher, which builds a dictionary that maps from spans of text to their classes using the training set. This model then searches for exact matches in both the validation and training sets to perform entity extraction.

5.1.1 Entity Extraction - Precision

The precision is the ratio of correctly extracted entities to the total extracted entities. A high value indicates high consistency between training and test annotations.

If the precision of tokens is much higher than that of entities, it indicates agreement on the extracted entities but not on their boundaries. For example, consider the sentence "He also noted a 23 pound weight loss [...]" (where a 23 pound weight loss is an entity). In the test set, this might appear as "The patient shows 23 pound weight loss [...]". Here, the entity is marked both as a false positive and as a false negative due to boundary disagreements. However, only one token ('a') is a false positive, while the rest are true positives. Hence, a marked difference between the precision of the tokens and entities indicates that there is a low boundary agreement.

Low precision, both for entities and tokens, can be attributed to two scenarios. This may indicate either a qualitative deficiency in the annotation of entities or a context-dependent extraction of entities. For instance, in EBM NLP, where entities are classified as `Participants`, `Interventions`, and `Outcomes`, an entity identified as an `Outcome` at the beginning of a document might actually represent an initial illness rather than an outcome.

Determining whether the issue comes from the first or the second scenario can be challenging. However, if a technique that proficiently uses context yields a low score, this outcome is likely due to either the low quality of the dataset or the inherent difficulty of the task.

5.1.2 Entity Extraction - Recall

Recall is calculated by dividing the number of entities correctly extracted by the total number of present entities, which includes both those missed and those caught.

A recall value near one indicates that the training and test sets are very similar, containing mostly identical data. In this scenario, generalization abilities are not as necessary, whereas the ability for memorization becomes crucial.

5.2 Embedding based

This section covers the experiments conducted with conventional techniques, such as EntityRecognizer and SpanCategorizer.

5.2.1 Entity Extraction vs Entity Classification

We decided to split the task into two different sub-tasks for several reasons:

- The Italian dataset can only be tested on Entity Extraction
- The two sub-tasks may require different abilities of the model or technique (e.g., syntactical analysis, biomedical knowledge, memory, generalization, ...)
- The SpanCategorizer may be improved by developing a smart SpanSuggester

5.2.2 EntityRecognizer vs SpanCategorizer

This experiment is used to compare the two main techniques offered by spaCy, namely the EntityRecognizer, explained in Chapter 4 Section 4.1.2, and the SpanCategorizer, detailed in Section 4.1.3.

The SpanCategorizer is designed for a different task of EntityRecognizer but can be adapted. Hence, this experiment measure which technique is superior.

5.2.3 SpanCategorizer overlap filters

The SpanCategorizer, designed to identify overlapping entities, led us to develop two algorithms for isolating non-overlapping entities. This allows a direct comparison between the EntityRecognizer and SpanCategorizer.

We created two different filters:

- **Longest entities**: This filter selects the longest entity among the overlapping entities
- **Highest score**: The SpanCategorizer can return the confidence scores of the classified candidates. This filter uses the score to select the entity with the highest confidence score among the overlapping entities

5.2.4 Ngram vs SpanFinder

The spaCy documentation suggests that the SpanFinder is a faster alternative to the SpanCategorizer. However, their tests show it yielding slightly worse results. We wanted to validate spaCy's results within the biomedical domain.

5.2.5 Mix Entity Extraction and Entity Classification

The SpanCategorizer has two distinct sub-components, one that returns candidates, called SpanSuggester, and another that classifies them. It is logical to assume that a better SpanSuggester would increase the performance of the SpanCategorizer. Therefore, we investigated two different approaches to increase the quality of the SpanSuggester:

- using an EntityRecognizer as a SpanSuggester using the same embedding component as the SpanCategorizer
- using a standalone pipeline pre-trained only for entity extraction

Shared embedding component

The SpanSuggester and SpanCategorizer are trained on the same documents but with different annotations. The former is trained to extract only one class of entities, whereas the latter is trained to classify all the candidates without discarding any.

The advantage of having only one embedding component, either Tok2Vec or BERT-like model, is that both sub-tasks can properly guide the gradients and can improve the models' comprehension of the task. However, due to budget constraints, this technique has not been tested.

Standalone pipeline

The two sub-tasks use different embedding components, and for this reason the classifier can be trained in a second phase.

The already existing models trained for entity extraction, can be used as SpanSuggesters. Hence, this experiment is only about creating a good model for entity classification, taking without any adaptation the best model for entity extraction.

The SpanCategorizer is trained giving only ground truth candidates, in order to not mix entity extraction and entity classification. This allows the learning to focus solely on the classification task, ignoring the entity extraction part.

Another advantage of mixing models for entity extraction and entity classification is that we can use different transformer models for the two tasks, making it possible to leverage, for example, RoBERTa's ability in syntactical analysis and MedBERT's or CODER's biomedical knowledge.

5.3 GPT based

There are several decoder-only models available that we could have used for the experiments, but we focused our research only on three models: GPT-3.5, GPT-4, and UniversalNER.

GPT-3.5 and GPT-4 are general-purpose, whereas UniversalNER is trained specifically for NER.

OpenAI models are frequently updated; in the last six months, a lot has changed. For example, when this project started, GPT-4 had not yet been released, and the maximum context size for GPT-3.5 was limited to 4191 tokens, with no options for fine-tuning available

UniversalNER, a.k.a UniNER, is designed for off-the-shelf use, requiring users to specify only the classes of entities they wish to extract.

We used three different ways to frame the task of NER:

- **All-in-one**: ask the model to perform entity extraction and entity classification of all the entities in a single message

  Entity extraction: ask the model to only return a list of entities

  Entity classification: given the document and a candidate, return the class of the candidate

5.3.1 All-in-one

This formulation asks the model to perform both entity extraction and entity classification.

With MedMentions, we also experimented with generating additional information like the definition and the preferred label (a standard label for the concept), to facilitate additional analysis of the model's comprehension of the entities.

We utilized an external tool named "CUI Guesser", developed by Bollino

et al. [5], to investigate the process of generating the preferred label and definition. This tool employs CODER and a pool strategy to create a vector representation of a concept candidate, followed by a search for the closest matching concept within a densely compiled representation of all the concepts of interest.

This task requires the output to follow a certain format in order to programmatically extract the information.

The prompt that produces the best results is:

Extract medical and clinical entities and their details from text. For each entity, you have to extract the following information: the original text span, definition, semantic group, and the preferred label. Retrieve and return precise information for the entities mentioned in the text.
The semantic group of the entity must be one of the following: Organizations, Activities & Behaviors, Genes & Molecular Sequences, Chemicals & Drugs, Geographic Areas, Living Beings, Disorders, Occupations, Procedures, Anatomy, Concepts & Ideas, Devices, Objects, Phenomena, Physiology.
Use the sentences as context to extract a good definition. Use the original text and the definition to extract the preferred label. The entities must be reported without any change compared to the original text. The output must be in this format:
| span | definition | semantic group | preferred label |

This prompt consumes 179 tokens using the OpenAI tokenizer. Since the task is to generate a definition, the semantic group (the class), and the preferred label, each entity consumes a lot of tokens reducing the space for few-shot

techniques.

5.3.2 Entity Extraction

This formulation asks the model only to extract the entities. This task has less token consumption compared to All-in-one since it just lists the entities, without the definition, semantic group, and preferred label.

This task has been explored on MedMentions, EBM NLP, and IT dataset.

This task requires the generation of fewer tokens, and it is possible to provide more examples in the prompt.

Here is reported the prompt that produces the best results:

```
You are a system that performs Named Entity Extraction in
text.
Given a text, create a list of all the entities from
specific classes.
The classes of entities to extract are: [LIST OF SEMANTIC
GROUP]. The text may contain repetitions; in this case,
also repeat the extracted entities.
The output must be in this format: entity1 || entity2 ||
entity3 || ...
```

This prompt consumes 122 tokens (for MedMentions) but requires far fewer tokens for the examples compared to All-in-one.

5.3.3 Entity Classification

This formulation requires in input the document and a candidate. The task is to classify the candidate using information and context from the document. To classify all the entities in the document the model must be prompted several times, one for each entity.

The prompt per se is quite short but the request, with the documents and

eventually the examples, increasing by several folds the token consumption.

For this prompt we wanted to investigate the Chain-of-Thought approach. The prompt asks the model to produce additional information, like the preferred label, the definition, and finally the semantic group. The idea is that if the model generates additional information, it can reuse the information for a more guided classification. This approach ideally can only be used for MedMentions because we have UMLS knowledge, but we manually annotated one or two examples for EBM NLP.

Here is reported the prompt that produce the best results:

```
You are a system that performs Named Entity
Classification and Information Generation for a given
entity in a document.
Given the document and the entity, you have to classify
and generate an appropriate preferred label and definition
for the specified entity.
To classify the entity, you must use the information
generated for the preferred label and the definition.
The class must be one of the following: [LIST OF SEMANTIC
GROUP].
The output must be in this format: | preferred label |
definition | class |
```

To measure the influence of Chain-of-Thought we evaluated the performance of the above mentioned prompt and another prompt that ask directly to output the class.

5.3.4 Fine-Tuning

As said in Chapter 5, Section 4.4.2, GPT-3.5 can be trained with custom data to specialize the model to a task.

This experiment shows how GPT-3.5 performances scale with more data

rather than few example in the prompt. However, this experiment is costly and, for this reason, we fine-tuned GPT-3.5 only for entity extraction on MedMentions, EBM NLP, and IT dataset. Additionally, we wanted to test how the performances would change with half data availabel.

On MedMentions truncated and EBM NLP we trained two variants, one with a dataset composed of one hundred manually selected documents and trained for five epochs, and another variant with fifty documents for three epochs.

5.4 Dataset scaling

This experiment responds to the questions: how much do the performances scale with datasets size? Can GPT-3.5 be used when the dataset is very small? What is the threshold after which is better to use traditional techniques?

This experiment is designed to compare the generative approach to traditional techniques. It measure how much the datasets can be expanded to increase the results.

The process of annotating a dataset is very costly and with a knowledge on the increases of performance with additional data, is possible to measure how much data, and money, is required to reach a certain quality.

This experiment requires the training of several different models on different subsets of the same dataset. For this reason, we trained only EntityRecognizer with RoBERTa-base.

The experiment is performed only on MedMentions truncated and EBM NLP because they have a large training set.

The datasets are created by using the n-first documents from the training set and keeping intact the validation set. Both MedMentions and EBM NLP are selected with the same cuts: 20, 40, 50, 75, 100, 150, 200, 250, 300, 400, 600, 800, 100, 1300, 1600, 1900.

Chapter 6

Discussion of the results

In our experiments involving various models, we observed significant performance variability, particularly in MedMentions truncated and EBM NLP. We hypothesize that spaCy's training phase, which seems sensitive to randomness, especially with low-quality datasets, may contribute to this variability. Consequently, these results are unsuitable for a much depth analysis of the characteristics that might favor one technique over others.

Although we did not explore the effects of randomness in training extensively, its impact became evident in our dataset scaling experiments. These results will be explained later in this chapter, but the final outcome is that fluctuations are quite relevant, especially for entity extraction.

The comparison between datasets cannot be proven by looking at the results of generative approaches because the results are not reliable. They are black box oracles; therefore, it is not possible to obtain any relevant metrics that reflect some characteristics of the dataset. Hence, the discussion on the results of generative approach does not include an explanation of the results, but offer only some ideas that justify the results.

6.1 Naive baseline results

In Table 6.1 are reported the scores obtained using the naive baseline on the datasets.

	Entities		Tokens	
Name	**Precision**	**Recall**	**Precision**	**Recall**
MedMentions	0.40	0.66	0.68	0.80
MedMentions Truncated	0.32	0.66	0.53	0.78
EBM NLP	0.07	0.34	0.28	0.62
I2B2 2010	0.52	0.49	0.85	0.49
IT dataset	0.73	0.64	0.93	0.67

Table 6.1: Results of the naive baseline across all tested datasets

MedMentions and MedMentions truncated The scores of MedMentions are quite good, indicating that the training set is representative and has good agreement on entity boundaries

The results show that the truncated version has more false positives for both tokens and entities. This implies that numerous entities sharing identical text possess different CUIs, some are included in the dataset while others are not. This inconsistency might confuse the final model, subsequently diminishing the results.

EBM NLP This dataset has the lowest scores for entities and an average score for the recall of tokens. This indicates that the training set is fairly representative, though there is low agreement on entity boundaries. The low precision may also indicate that entity extraction is context-dependent, the dataset quality is low, or that the task is inherently hard.

The quality of the dataset can be assessed by measuring the performance using more sophisticated technique: if they have a low score, it means that the quality is low, or that the dataset is difficult to manage.

I2B2 2010 This dataset has the same recall for entities and tokens, indicating that the training set is representative. The precision gap between entities and tokens suggests a low entities boundary.

IT dataset The scores of this dataset suggest that it is more manageable. The training set is representative, and extensive generalization ability is not required. The results from the naive baseline suggest that, with this dataset, spaCy's components are likely to achieve good scores.

6.2 Entity extraction

These experiments have been conducted with a total of 81 different models across five datasets, 32 of which are specifically for entity extraction.

Figure 6.1 contains the main results of entity extraction and is followed by a description of the results focusing on each dataset.

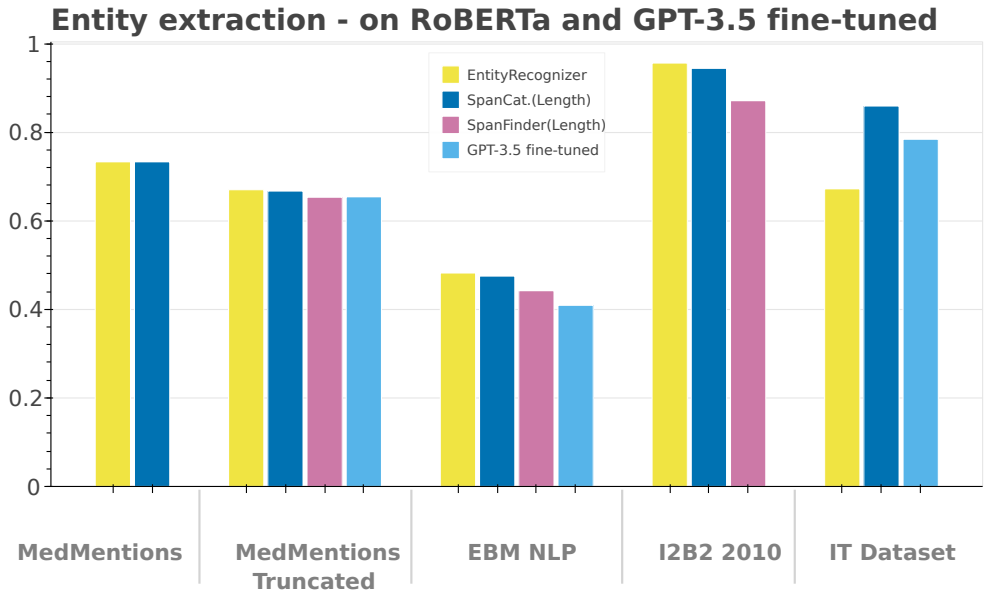

Figure 6.1: Entity extraction results across all datasets, comparing traditional techniques with RoBERTa trained for entity extraction and GPT-3.5

6.2.1 MedMentions

It can be observed that, by comparing the results of models trained for entity extraction with those trained for entity classification, the models trained for entity extraction systematically outperform those trained for entity classification.

For both entity extraction and entity classification, the best results are obtained using EntityRecognizer with RoBERTa-base, as indicated by the F1 scores for entities and tokens.

Given that RoBERTa-base consistently outperforms other models, it suggests that in this dataset, text manipulation capability is more crucial than biomedical knowledge.

6.2.2 MedMentions truncated

By removing some entities from the dataset, we observed the emergence of some outlier models. In this truncated dataset, the difference between EntityRecognizer and SpanCategorizer is negligible, but the former exhibits a slightly better ability at extracting the tokens.

This version shows a decrease in performance, with a 6-point drop in the F1 score for entities and a 13-point reduction for tokens.

6.2.3 EBM NLP

The naive baseline yields lower performance metrics on this dataset due to a very low precision (0.07). The baseline's performance, confirmed by the entity extraction results, suggests a low quality of this dataset.

Additionally, the low quality of dataset leads to numerous outliers. For instance, various models trained for entity classification outperform their counterparts in entity extraction. This phenomenon is unique to this dataset and has not been observed in other datasets.

6.2.4 I2B2 2010

The primary issue with this dataset is its limited size, comprising only 136 documents for training and 33 for validation. The surprising effectiveness observed may be caused by the specific structure of the documents and the high quality of the annotations. The entities are primarily presented in lists, and the models seems to exploit this format quite proficiently.

Another important observation is the performance gap between models trained for entity extraction and those for entity classification. This phenomenon may be attributed to the different skill requirements. The data suggests that, for structured documents, the task of entity extraction and entity classification is very different. Unfortunately, this is the only dataset tested that has this structure, hence this phenomenon cannot be verified.

6.2.5 IT dataset

The naive baseline shows that has similar results compared to I2B2 2010. However, due to the high quality of I2B2 2010, and the language difference, there is a notable nine-point gap.

The results of entity extraction using traditional techniques follow the baseline results. In particular, the models exhibit a very high F1 score for tokens.

The results on this dataset are overall good, but there is an unexpected outcome: In Italian Tok2Vec excels using EntityRecognizer, while the SpanCategorizer shows similar results of BERT-based models.

A common model used across this and other datasets is CODER, which generally underperforms. However, this indicates that the strange score is not due to the overall performance of the Italian models but rather to the exceptionally high performance of Tok2Vec combined with EntityRecognizer in Italian.

6.2.6 Entity extraction conclusion

This experiment demonstrates the performance boost obtained by training a model specifically for entity extraction, as depicted in Figure 6.2, which shows the improvement on EntityRecognizer.

The performance marginally increases in the MedMentions and EBM NLP datasets, whereas in the I2B2 2010 dataset, there is a notable growth of 6 to 9 points in performance, varying by model.

In the EBM NLP dataset, an unexpected phenomenon was observed when using the SpanCategorizer: a performance decline of up to 2 points. This decline may be attributed to the lower quality of the dataset, which benefits from the additional information provided by the classification task.

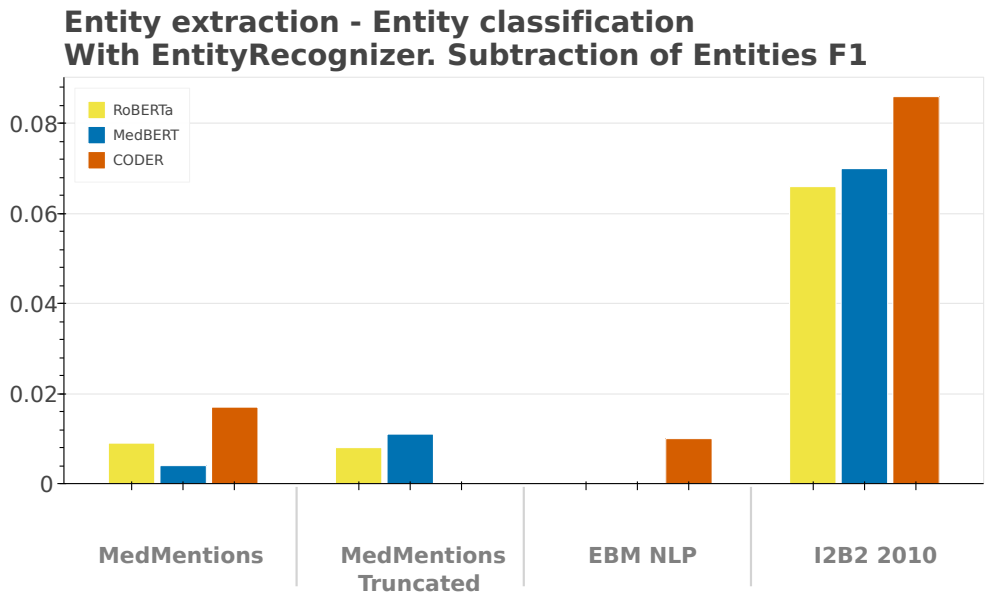

Figure 6.2: Comparison of entities F1 on EntityRecognizer with three different models for each dataset

6.3 Entity classification

For these experiments, we trained a total of 57 models: 16 each for the EntityRecognizer and the normal SpanCategorizer, 9 for the SpanFinder, and 16

for the SpanCategorizer Mix.

Figure 6.3 reports the main results and is followed by a description of the results for each dataset.

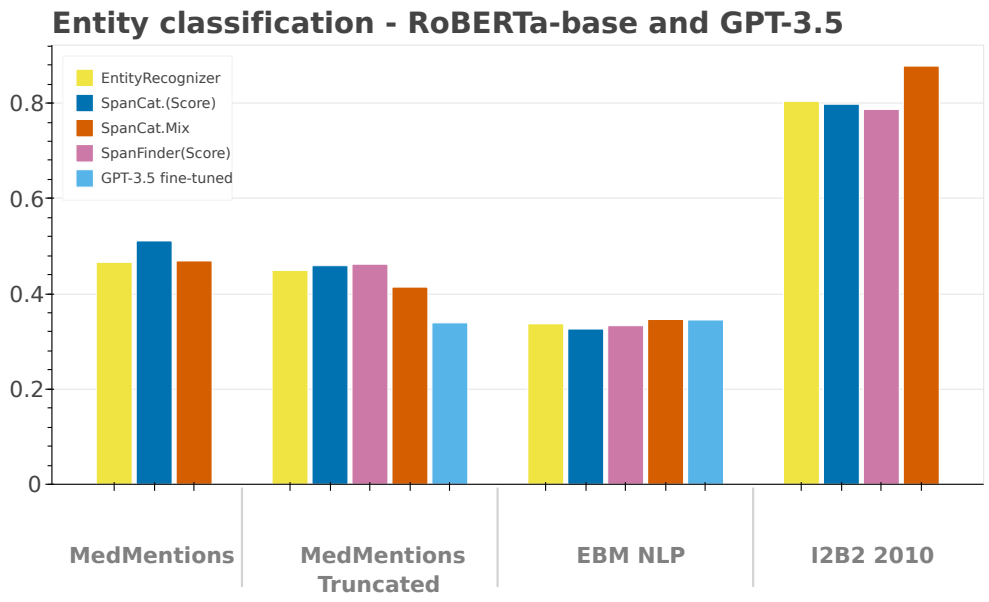

Figure 6.3: Results of entity classification for all the datasets and all the traditional techniques with RoBERTa and the GPT-3.5

6.3.1 MedMentions

MedMentions has a F1 weighted average of 0.51 using SpanCategorizer with RoBERTa-base. RoBERTa-base shows better performance compared to other models. This suggests that with MedMentions, even entity classification requires textual analysis skills.

The SpanCategorizer with RoBERTa-base shows marked better performances compared to the EntityRecognizer with the same model. This phenomenon is unique to this dataset and has not been observed in other datasets, for this reason it is considered an outlier.

6.3.2 MedMentions truncated

The truncated version of MedMentions exhibits marginally inferior performance in comparison to the complete version, contrary to entity extraction. In entity extraction, this version loses 6 F1 points, while for classification, the score is almost the same.

Given that the performance of SpanCategorizer mirrors that of the EntityRecognizer, it is plausible that the SpanCategorizer's superior results with RoBERTa-base on the complete MedMentions may represent an outlier.

6.3.3 EBM NLP

In the section on entity extraction, it has been shown that the quality of this dataset is low, and the classification reflects similar results. The results of the EntityRecognizer show that RoBERTa is the best model. However, when using the SpanCategorizer, MedBERT emerges as the best model for the three variants tested.

Notably, the SpanCategorizer Mix produces the best results for both RoBERTa-base and CODER models.

6.3.4 I2B2 2010

The results reaffirm the high quality of the dataset. The classification achieves a weighted average F1-score of 0.89.

The entity extraction experiments indicate differing skill requirements between entity extraction and entity classification within this dataset. The SpanCategorizer Mix was specifically developed for this purpose. It increases the weighted average F1 by 10 points.

This substantial improvement primarily indicates that separating entity extraction and entity classification can be beneficial, particularly when there is a noticeable performance disparity between models trained specifically for each task.

6.3.5 Entity classification conclusion

The conclusions from the task of entity classification are as follows:

- The SpanCategorizer and the EntityRecognizer have similar score
- It is not clear which is superior between the normal SpanCategorizer and the SpanCategorizer with SpanFinder
- The SpanCategorizer Mix can be a powerful tool if the dataset exhibits differing scores in entity extraction between model trained specifically for entity extraction and model for entity classification
- The performance of entity classification follows the results of entity extraction, i.e., the quality of the dataset or the difficuly of the task influence both entity extraction and entity classification
- Whereas for entity extraction, the scores gap of MedMentions truncated compared to the complete version is evident, for entity classification the gap is minimal

6.4 EntityRecognizer vs SpanCategorizer

Figures 6.1 and 6.3 compare the results of EntityRecognizer and SpanCategorizer. It is particularly noticeable that the results for both extraction and classification are very similar.

The final outcome indicates that the performance difference is minimal.

6.5 SpanCategorizer overlaps filter

Figure 6.4 illustrates the performance differences between models using length-based and score-based filters. The results show that the length-based filter performs slightly better, though its impact on the overall score is not significant.

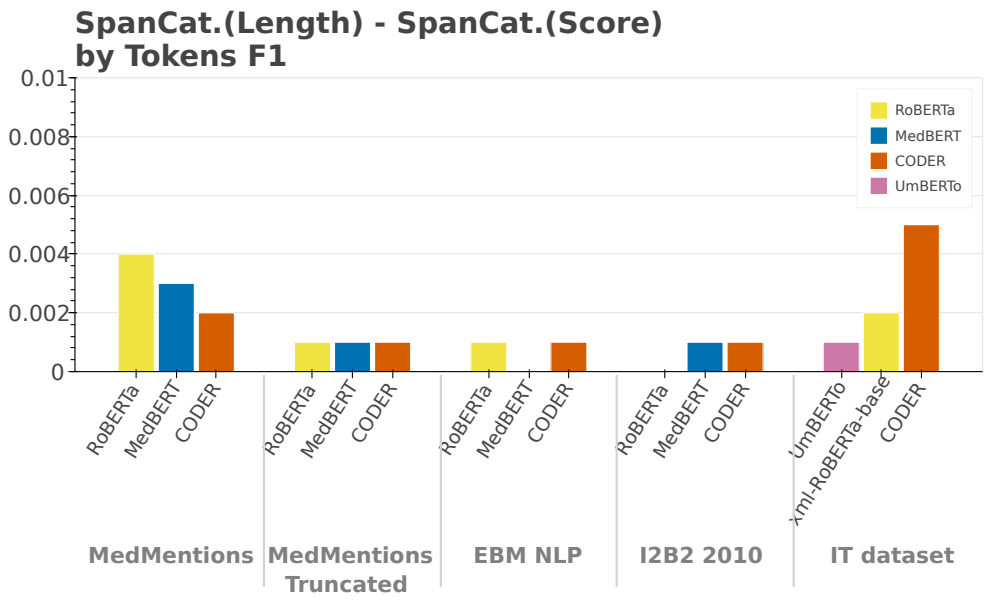

Figure 6.4: Comparison of overlap filters in SpanCategorizer across all datasets

6.6 SpanFinder as SpanSuggester

The training for the SpanCategorizer with SpanFinder requires additional vram compared to other techniques, necessitating the use of an AWS machine. To reduce the costs of this experiment, we focused solely on training models for entity classification. Consequently, SpanFinder can only be compared with other models trained specifically for entity classification.

Figure 6.5 illustrates the difference in weighted average F1 scores between the standard SpanCategorizer and the SpanCategorizer integrated with SpanFinder. The final outcome indicates that there is no clear evidence to determine which technique is superior.

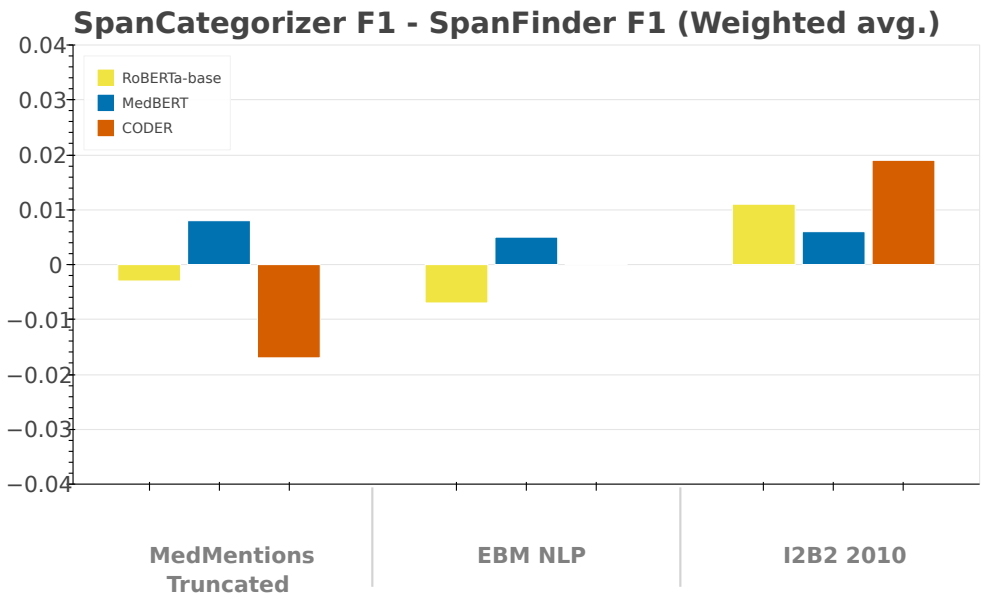

Figure 6.5: Mathematical subtraction of SpanCategorizer with Ngram suggester and SpanCategorizer with SpanFinder

6.7 Mix entity extraction and entity classification

This experiment leverage the distinct skills required for extracting and classifying entities. Therefore, an improvement in performance is expected only when this experiment is applied to datasets exhibiting these characteristics. I2B2 2010 dataset shows a marked improvement of the performance.

Another important aspect is that while with other techniques the best model is most of the times RoBERTa-base, with this technique the best model is always MedBERT. This can be intuitively attributed to the background biomedical knowledge, which significantly influences classification ability.

The final outcome indicates that this technique proves advantageous in scenarios where there is a discernible difference in F1 scores between models trained specifically for extraction and those trained for classification.

6.8 Generative approach

The results of the experiments using generative approaches are reported here since the outcomes are applicable across all three tested datasets.

6.8.1 UniNER

This model yielded no usable results. The performance of the 7B version was too low to be useful. The results are reported in Appendix B.

6.8.2 All-in-one

The performance of this technique, reported in Figures 6.6 and 6.7, was significantly lower compared to other tasks. Consequently, it was only tested on MedMentions truncated and then discarded.

The All-in-one approach was designed also to assess the model's comprehension of UMLS, but this experiment yielded no useful results. For this reason, our attention shifted to more promising tasks.

For entity classification, All-in-one achieved bad results, similar to UniNER and scored twenty points lower than other methods.

6.8.3 Entity extraction

The findings show that traditional techniques outperform the generative approach in entity extraction. However, fine-tuning can establish a competitive baseline, with relatively small datasets.

Performance improves in few-shot scenarios, particularly when examples are carefully chosen to minimize token consumption. Our research into optimal prompting revealed that prompts should be kept below 4000 tokens, incorporating two or three examples that are concise yet comprehensive.

Fine-tuning consistently enhances performance. For instance, a fine-tuned GPT-3.5 outperforms GPT-4 in a one-shot scenario. However, in the Italian

language, GPT-4 in a one-shot setting achieves similar results to that of a fine-tuned GPT-3.5

Performance is further enhanced when fine-tuning is combined with few-shots, leveraging the strengths of both approaches.

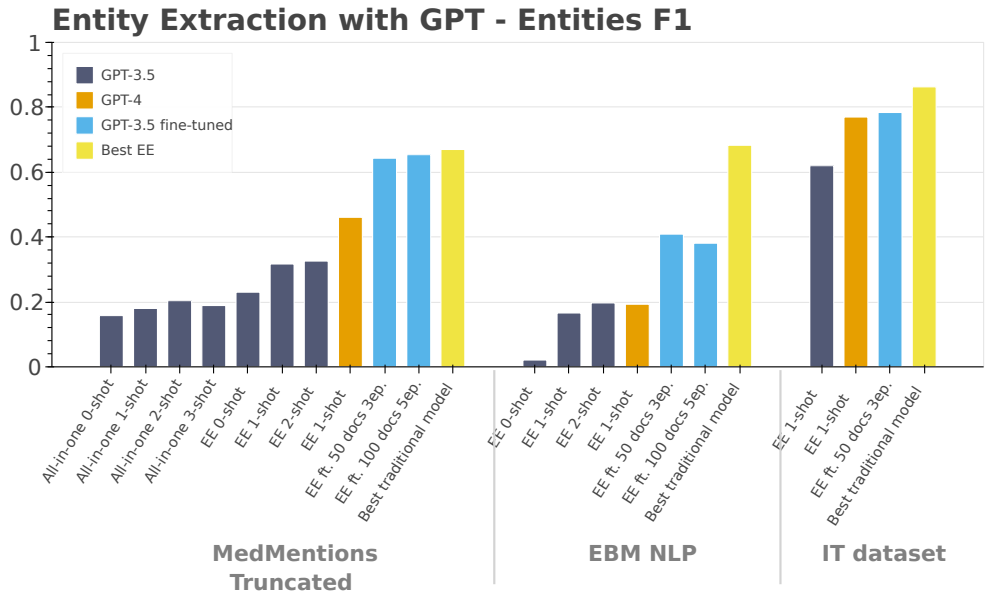

Figure 6.6: Results of GPT prompts for entity extraction. For comparison, the figure includes the dataset's top-performing model for entity extraction

6.8.4 Entity classification

During the development of this experiment, we attempted to include several candidates in the prompt, but the model sometimes overlooked the entities and produced mixed results. Consequently, the final prompt asks for only one candidate at a time, even if this increases the cost.

For the evaluation of this task, we utilized the best-performing model for entity extraction for each dataset. This approach allows us to isolate and compare the classification task more effectively.

Figure 6.7 shows the result of this task using generative approaches.

The Chain-of-Thought prompt performs worse than the simple prompt,

a phenomenon for which we currently lack an explanation. This may be attributable to the prompt's design, which excludes the document from the reasoning context, causing GPT-3.5 to underperform when relying solely on the definition. However, this theory cannot be verified.

The scores of MedMentions truncated are similar to EBM NLP. However, considering that MedMentions entities are presumably easier to classify according to traditional techniques, the results suggest that a greater number of classes may adversely affect performance. Again, while this cannot be conclusively proven, it stands as the most plausible explanation available.

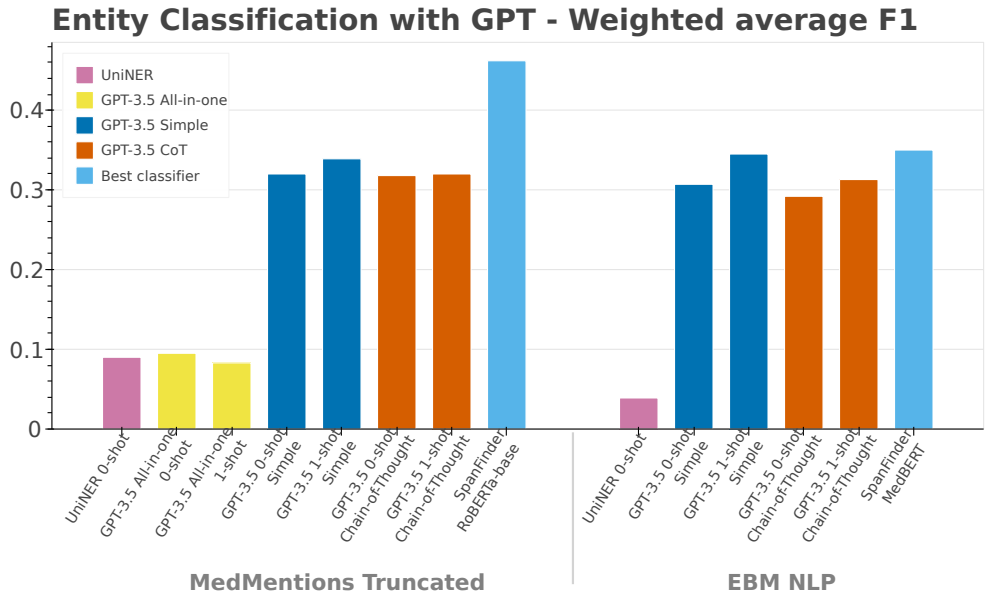

Figure 6.7: Results of GPT prompt with MedMentions truncated and EBM NLP, as a reference is used the best model for classification

6.9 Dataset scale

Measuring the model's performance when trained with identical hyperparameters on different subsets of the dataset can clarify the influence of randomness during the training phase. However, the primary focus of this experiment is not the impact of randomness but rather the scalability of models relative to document .

This experiment is considered a success if the results show that GPT-3.5, fine-tuned with fewer than 100 documents, performs better than the traditional techniques trained with more than 100 documents.

The scaling has been tested for entity extraction and entity classification with a total of 64 different models.

This experiment reveals interesting findings on MedMentions Truncated and less encouraging results on EBM NLP.

6.9.1 MedMentions truncated

Figure 6.8 shows the results of this experiment.

Comparing the models trained for entity extraction with those trained for entity classification indicates fluctuating performance levels.

The performance scaling does not consistently increase, possibly because some documents in the training subset adversely affect the overall performance. With a dataset size of 1300 documents, the model surpasses the F1 score threshold for entity recognition. Giving more documents seems to worsen the performances, but using the full dataset ultimately delivers the most effective results.

6.9.2 EBM NLP

GPT-3.5, when fine-tuned for EBM NLP, demonstrates a significantly lower score compared to the best-performing model. The model surpasses the threshold F1-score over the entities with a dataset of only 150 documents. Although the experiment yielded positive results, the marginal difference does not clearly indicate the superiority of one technique over the other.

Performance oscillation is more pronounced in this dataset, particularly for the Tokens F1 metric. This may demonstrate that the dataset's quality indeed influences the impact of randomness during training.

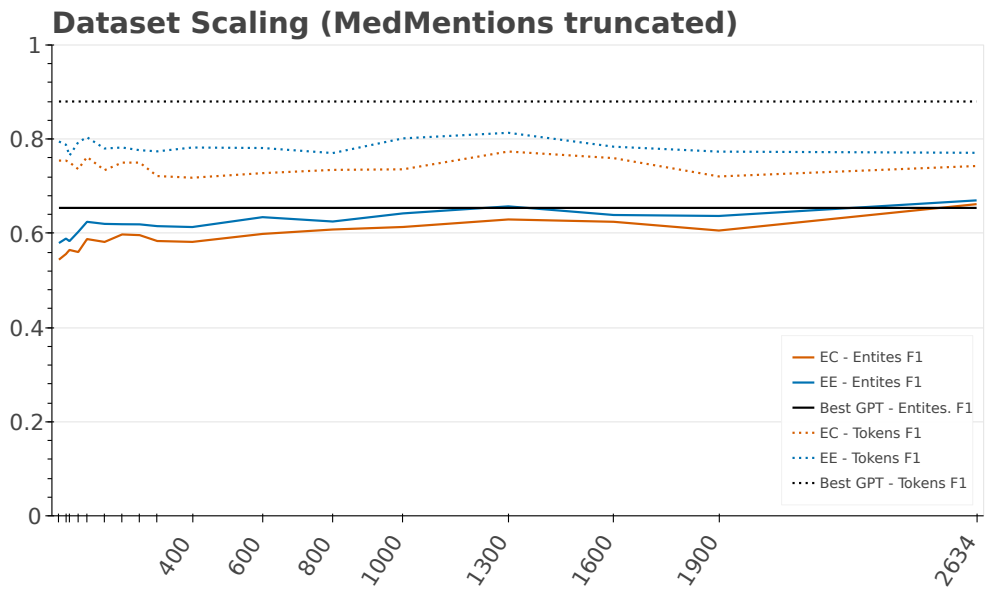

Figure 6.8: Scaling results of MedMentions truncated using EntityRecognizer with RoBERTa-base

6.9.3 Dataset scale conclusions

The outcomes of this experiment indicate that in the case of small datasets (fewer than 100 documents), generative approaches offer a viable alternative to traditional techniques.

The effectiveness of this technique cannot be determined a priori, as it significantly depends on the characteristics of the dataset.

It remains unclear how GPT-3.5 will scale with an increased volume of fine-tuning data, for example using a thousand documents.

While the use of Large Language Models in this approach is limited by monetary costs, it serves as a viable substitute when other approaches fail due to a small dataset.

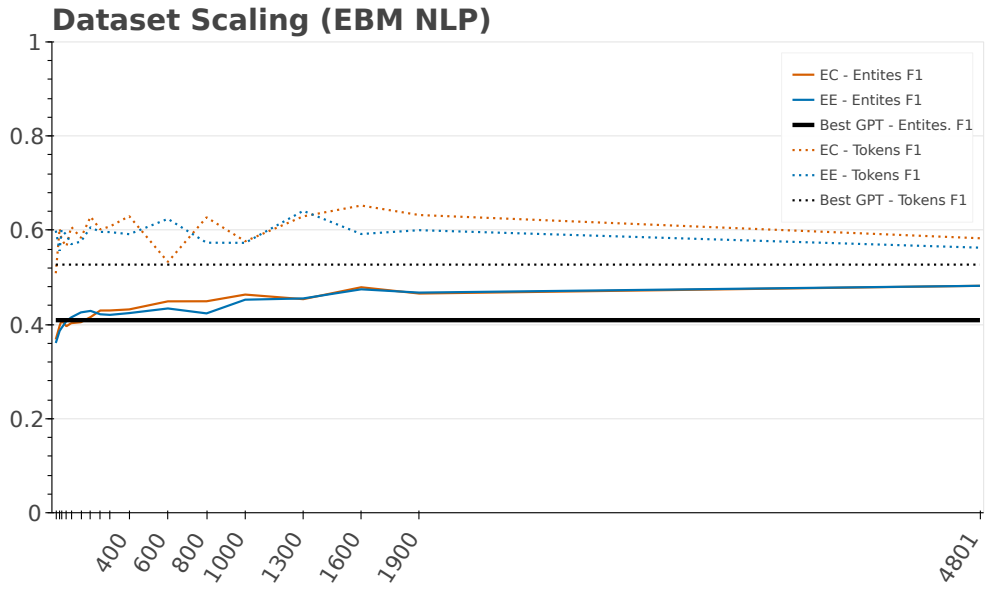

Figure 6.9: Scaling results of EBM NLP using EntityRecognizer with RoBERTa-base

Chapter 7

Conclusions

This thesis reports the techniques, with their performance, for entity extraction and classification of biomedical entities. Alongside the evaluation of the techniques, we also conducted several experiments designed to test the models in various contexts, and to search for characteristics that help understand the success of certain techniques over others.

We also explored how Large Language Models can be effectively used for entity extraction and classification, as seen in Chapter 6, Section 6.8. Our findings suggest they are best used off-the-shelf when data limitations preclude the training of traditional techniques. In scenarios where data is sufficient for fine-tuning Large Language Models, but inadequate for traditional methods, these models exhibit promising results and can serve as an effective alternative, as seen in Chapter 6, Section 6.9. In all other cases, where the dataset is sufficiently large, traditional methods are the best choice.

Our experiments demonstrated that the quality and type of datasets has a significant impact on performance, thus complicating the prediction of outcomes.

Some datasets require different skills for entity extraction and entity classification, and learning one compromises the other. In such cases, disentangling the two tasks can be achieved by training separate models exclusively for entity extraction and entity classification, thereby harnessing the strengths

of both models.

7.1 Future works

In this research, we concentrated on applying Large Language Models exclusively to tasks traditionally handled by other techniques, specifically entity extraction and entity classification.

They can be used for many different tasks, for example translating the IT dataset in English, using Large Language Models for navigating and potentially enhancing the UMLS graph, paraphrasing a document, solving abbreviations, and more. However, a significant limitation in exploring these possibilities is the challenge of assessing performance due to the scarcity of available data in the biomedical field. This underscores the need for expanded data collection and analysis to fully leverage the potential of Large Language Models in biomedical research.

Acknowledgements

I would like to extend my sincere gratitude to Vieri Emiliani for the extraordinary support, and patience, showed during my internship, which culminated in this project.

I am also thankful to my fellow interns at MAPS S.P.A., Massimiliano Barbieri, Lorenzo Niccolai and Emanuele Bollino, for their collaboration.

Precautionally, I am also thankful to ChatGPT for its rapid and for, more often than not, accurate responses to my inquiries during my studies.

Finally, but certainly not least, I am deeply grateful to my family, who have supported my studies and encouraged me to persevere through challenges.

Bibliography

[1] Amazon mechanical turk. https://www.mturk.com/. (Accessed on 11/09/2023).

[2] Art. 4 gdpr – definitions - general data protection regulation (gdpr). https://gdpr-info.eu/art-4-gdpr/. (Accessed on 11/09/2023).

[3] K. Bhatia, A. Narayan, C. D. Sa, and C. Ré. Tart: a plug-and-play transformer module for task-agnostic reasoning, 2023. eprint: arXiv:2306.07536.

[4] Bodenreider o. the unified medical language system (umls): integrating biomedical terminology. nucleic acids res. 2004 jan 1;32(database issue):d267-70. doi: 10.1093/nar/gkh061. pubmed pmid: 14681409; pubmed central pmcid: pmc308795.

[5] E. Bollino, V. Emiliani, P. Torroni, and A. Galassi. *Automatic Terminology Coding for the Biomedical Domain*. Master's thesis, University of Bologna, Artificial Intelligence, 2023.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Curran Associates, Inc., 2020. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[7] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: an open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL: https://lmsys.org/blog/2023-03-30-vicuna/.

[8] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: scaling language modeling with pathways, 2022. eprint: arXiv:2204.02311.

[9] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020. URL: https://openreview.net/forum?id=r1xMH1BtvB.

[10] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the*

58th Annual Meeting of the Association for Computational Linguistics, pages 8440–8451, Online. Association for Computational Linguistics, July 2020. DOI: 10.18653/v1/2020.acl-main.747. URL: https://aclanthology.org/2020.acl-main.747.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics, June 2019. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[12] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks, 2013. eprint: arXiv:1302.4389.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. DOI: 10.1109/CVPR.2016.90.

[14] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models, 2022. eprint: arXiv:2203.15556.

[15] M. HONNIBAL. Embed, encode, attend, predict: the new deep learning formula for state-of-the-art nlp models · explosion. https://explosion.ai/blog/deep-learning-formula-nlp, 2016. (Accessed on 11/20/2023).

[16] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. In K. Knight, A.

Nenkova, and O. Rambow, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics, June 2016. DOI: 10.18653/v1/N16-1030. URL: https://aclanthology.org/N16-1030.

[17] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: a lite BERT for self-supervised learning of language representations, September 2019. arXiv: 1909.11942 [cs.CL].

[18] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, September 2019. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz682. eprint: https://academic.oup.com/bioinformatics/article-pdf/36/4/1234/48983216/bioinformatics_36_4_1234.pdf. URL: https://doi.org/10.1093/bioinformatics/btz682.

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Ro{bert}a: a robustly optimized {bert} pretraining approach, 2020. URL: https://openreview.net/forum?id=SyxS0T4tvS.

[20] B. McCann, N. S. Keskar, C. Xiong, and R. Socher. The natural language decathlon: multitask learning as question answering, 2018. eprint: arXiv:1806.08730.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013, January 2013.

[22] S. Mohan and D. Li. Medmentions: a large biomedical corpus annotated with umls concepts, 2019. eprint: arXiv:1902.09476.

[23] OpenAI. Gpt-4 technical report, 2023. eprint: arXiv:2303.08774.

[24] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

[25] L. Parisi, S. Francia, and P. Magnani. Umberto: an italian language model trained with whole word masking. https://github.com/musixmatchresearch/umberto, 2020.

[26] J. Pennington, R. Socher, and C. Manning. GloVe: global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics, October 2014. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[27] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In M. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics, June 2018. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

[28] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018.

[29] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners, 2019.

[30] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003. URL: https://aclanthology.org/W03-0419.

[31] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: open and efficient foundation language models, 2023. eprint: arXiv:2302.13971.

[32] Ö. Uzuner, B. R. South, S. Shen, and S. L. DuVall. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *J Am Med Inform Assoc*, 18(5):552–556, 2011. [PubMed Central:PMC2243666] [DOI:10.1136/amiajnl-2011-000203] [PubMed:11825149].

[33] C. Vasantharajan, K. Z. Tun, H. Thi-Nga, S. Jain, T. Rong, and C. E. Siong. Medbert: a pre-trained language model for biomedical named entity recognition. In *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1482–1488, 2022. DOI: 10.23919/APSIPAASC55919.2022.9980157.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[35] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022. eprint: arXiv:2201.11903.

[36] J. Weston. Dialog-based language learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 829–837, Barcelona, Spain. Curran Associates Inc., 2016. ISBN: 9781510838819.

[37] B. Wong. Points of view: color blindness. *Nature Methods*, 8(6):441–441, 2011. ISSN: 1548-7105. DOI: 10.1038/nmeth.1618. URL: https://doi.org/10.1038/nmeth.1618.

[38] Z. Yuan, Z. Zhao, H. Sun, J. Li, F. Wang, and S. Yu. CODER: knowledge-infused cross-lingual medical term embedding for term normalization. en. *J. Biomed. Inform.*, 126(103983):103983, February 2022.

[39] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu, and G. Wang. Instruction tuning for large language models: a survey, 2023. eprint: arXiv:2308.10792.

[40] W. Zhou, S. Zhang, Y. Gu, M. Chen, and H. Poon. Universalner: targeted distillation from large language models for open named entity recognition, 2023. eprint: arXiv:2308.03279.

Appendix A

Dataset additional data

A.1 Dataset examples

Datasets

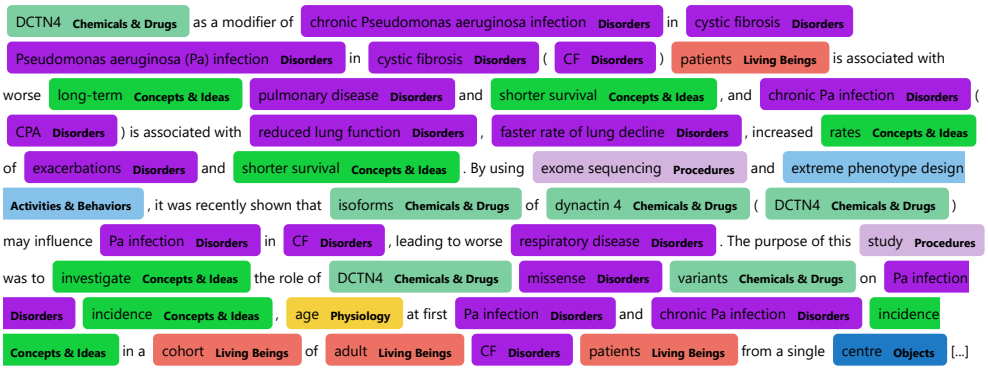

Figure A.1: Example of document from MedMentions

Figure A.2: Example of document from MedMentions Truncated

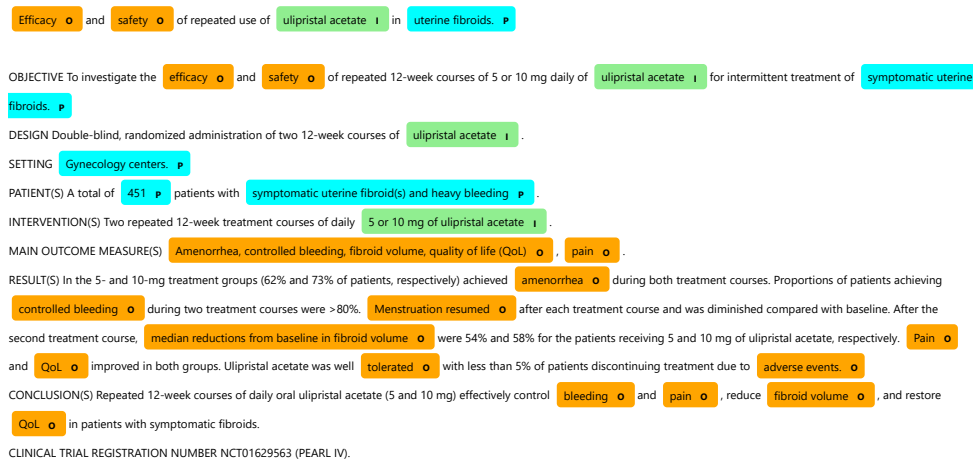

Figure A.3: Example of document from EBM NLP

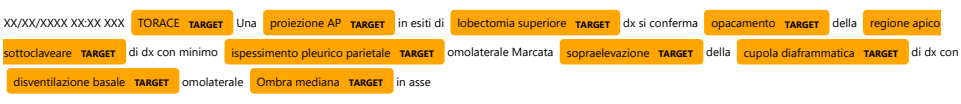

Figure A.4: Example of document from IT Target Guesser

Figure A.5: Example of document from I2B2 2010.

A.2 Dataset class statistics

Semantic group	Count	Count truncated	%	Truncated %
Chemicals & Drugs	30349	22047	10.89 %	-27.36 %
Disorders	35067	26972	12.58 %	-23.08 %
Living Beings	24855	20904	8.91 %	-15.90 %
Concepts & Ideas	92471	58905	33.17 %	-36.30 %
Procedures	29120	19508	10.44 %	-33.01 %
Activities & Behaviors	9598	4056	3.44 %	-57.74 %
Physiology	19603	12793	7.03 %	-34.74 %
Objects	5412	4353	1.94 %	-19.57 %
Anatomy	15538	13369	5.57 %	-13.96 %
Genes & Molecular Sequences	4427	2194	1.59 %	-50.44 %
Devices	2264	1607	0.81 %	-29.02 %
Phenomena	5829	3697	2.09 %	-36.58 %
Geographic Areas	2398	2293	0.86 %	-4.38 %
Organizations	732	283	0.26 %	-61.34 %
Occupations	1143	989	0.41 %	-13.47 %

Table A.1: Class Count of MedMentions

Super class	Class	Count	%
Participants	Age	2536	2.89%
Participants	Condition	10077	11.49%
Participants	Sample size	4598	5.24%
Participants	Sex	1033	1.18%
Interventions	Physical	5391	6.15%
Interventions	Drug	19771	22.54%
Interventions	Control	3041	3.47%
Interventions	Surgical	1806	2.06%
Interventions	Educational	2834	3.23%
Interventions	Psychological	469	0.53%
Interventions	Other	1036	1.18%
Outcomes	Physical	19242	21.94%
Outcomes	Other	7238	8.25%
Outcomes	Pain	981	1.12%
Outcomes	Mental	4322	4.93%
Outcomes	Mortality	1361	1.55%
Participants	-	18244	21%
Interventions	-	34348	40%
Outcomes	-	33144	39%

Table A.2: Class Count of EBM-NLP

Super class	Count	%
Problem	7056	43%
Test	4605	25%
Treatment	4839	29%

Table A.3: Class Count of I2B2 2010

Appendix B

Experimental Results

All the experiments results are reported here.

B.1 Entity extraction

B.1.1 MedMentions

			Entites			Tokens		
Embedding	**Component**	**Task**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Tok2Vec	EntityRec.	EE	0.652	0.706	0.678	0.877	0.870	0.874
Tok2Vec	EntityRec.	EC	0.621	0.645	0.633	0.861	0.787	0.822
RoBERTa	EntityRec.	EE	0.699	0.770	**0.733**	0.874	0.942	**0.907**
RoBERTa	EntityRec.	EC	0.696	0.758	**0.726**	0.872	0.924	**0.897**
MedBERT	EntityRec.	EE	0.702	0.718	0.710	0.892	0.896	0.894
MedBERT	EntityRec.	EC	0.688	0.725	0.706	0.881	0.899	0.889
CODER	EntityRec.	EE	0.691	0.714	0.702	0.891	0.891	0.891
CODER	EntityRec.	EC	0.674	0.717	0.695	0.874	0.894	0.884

Table B.1: Results of entity extraction for EntityRecognizer on MedMentions

			Entites			Tokens		
Embedding	**Component**	**Task**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Tok2Vec	SpanCat.(Score)	EE	0.697	0.643	0.668	0.887	0.748	0.811
Tok2Vec	SpanCat.(Score)	EC	0.707	0.498	0.584	0.895	0.538	0.672
Tok2Vec	SpanCat.(Length)	EE	0.696	0.642	0.668	0.887	0.750	0.813
Tok2Vec	SpanCat.(Length)	EC	0.709	0.499	0.586	0.895	0.544	0.677
RoBERTa	SpanCat.(Score)	EE	0.711	0.754	0.732	0.889	0.866	0.877
RoBERTa	SpanCat.(Score)	EC	0.759	0.653	0.702	0.911	0.732	0.812
RoBERTa	SpanCat.(Length)	EE	0.712	0.755	**0.733**	0.888	0.873	**0.881**
RoBERTa	SpanCat.(Length)	EC	0.759	0.653	**0.702**	0.911	0.735	**0.814**
MedBERT	SpanCat.(Score)	EE	0.715	0.710	0.713	0.898	0.805	0.849
MedBERT	SpanCat.(Score)	EC	0.716	0.635	0.673	0.887	0.721	0.796
MedBERT	SpanCat.(Length)	EE	0.716	0.711	0.714	0.898	0.811	0.852
MedBERT	SpanCat.(Length)	EC	0.716	0.636	0.674	0.887	0.726	0.798
CODER	SpanCat.(Score)	EE	0.699	0.698	0.699	0.886	0.837	0.861
CODER	SpanCat.(Score)	EC	0.695	0.653	0.673	0.878	0.751	0.809
CODER	SpanCat.(Length)	EE	0.699	0.698	0.699	0.885	0.841	0.863
CODER	SpanCat.(Length)	EC	0.695	0.653	0.674	0.877	0.755	0.811

Table B.2: Results of entity extraction for SpanCategorizer on MedMentions

B.1.2 MedMentions truncated

			Entites			Tokens		
Embedding	**Component**	**Task**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Tok2Vec	EntityRec.	EE	0.570	0.631	0.599	0.711	0.708	0.709
Tok2Vec	EntityRec.	EC	0.608	0.591	0.600	0.744	0.663	0.701
RoBERTa	EntityRec.	EE	0.629	0.716	**0.670**	0.736	0.810	**0.771**
RoBERTa	EntityRec.	EC	0.658	0.665	**0.662**	0.754	0.733	0.743
MedBERT	EntityRec.	EE	0.652	0.632	0.642	0.765	0.707	0.735
MedBERT	EntityRec.	EC	0.651	0.613	0.631	0.759	0.697	0.726
CODER	EntityRec.	EE	0.600	0.684	0.639	0.718	0.829	0.769
CODER	EntityRec.	EC	0.618	0.661	0.639	0.731	0.768	**0.749**

Table B.3: Results of entity extraction for EntityRecognizer on MedMentions truncated

			Entites			Tokens		
Embedding	**Component**	**Task**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Tok2Vec	SpanCat.(Score)	EE	0.662	0.489	0.562	0.774	0.493	0.602
Tok2Vec	SpanCat.(Score)	EC	0.656	0.527	0.585	0.779	0.560	0.652
Tok2Vec	SpanCat.(Length)	EE	0.662	0.489	0.562	0.774	0.494	0.603
Tok2Vec	SpanCat.(Length)	EC	0.656	0.527	0.585	0.778	0.563	0.653
RoBERTa	SpanCat.(Score)	EE	0.675	0.661	**0.668**	0.775	0.687	0.728
RoBERTa	SpanCat.(Score)	EC	0.667	0.626	0.646	0.763	0.658	0.707
RoBERTa	SpanCat.(Length)	EE	0.674	0.661	0.667	0.774	0.688	0.729
RoBERTa	SpanCat.(Length)	EC	0.667	0.626	0.646	0.762	0.660	0.707
MedBERT	SpanCat.(Score)	EE	0.645	0.650	0.648	0.751	0.701	0.725
MedBERT	SpanCat.(Score)	EC	0.671	0.570	0.616	0.774	0.613	0.684
MedBERT	SpanCat.(Length)	EE	0.645	0.650	0.647	0.751	0.703	0.726
MedBERT	SpanCat.(Length)	EC	0.670	0.570	0.616	0.773	0.616	0.685
CODER	SpanCat.(Score)	EE	0.613	0.669	0.639	0.734	0.748	0.741
CODER	SpanCat.(Score)	EC	0.645	0.589	0.616	0.764	0.634	0.693
CODER	SpanCat.(Length)	EE	0.612	0.669	0.639	0.734	0.751	**0.742**
CODER	SpanCat.(Length)	EC	0.646	0.590	0.616	0.763	0.636	0.694
RoBERTa	SpanFinder(Score)	EC	0.675	0.633	**0.653**	0.778	0.655	0.711
RoBERTa	SpanFinder(Length)	EC	0.674	0.633	0.653	0.777	0.659	0.713
MedBERT	SpanFinder(Score)	EC	0.645	0.625	0.635	0.760	0.669	0.711
MedBERT	SpanFinder(Length)	EC	0.644	0.624	0.634	0.758	0.675	**0.714**
CODER	SpanFinder(Score)	EC	0.671	0.575	0.619	0.781	0.624	0.694
CODER	SpanFinder(Length)	EC	0.670	0.575	0.619	0.780	0.628	0.696

Table B.4: Results of entity extraction for SpanCategorizer on MedMentions truncated

			Entites			Tokens		
Model	**Technique**	**Task**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
UniNER	0-shot	EE	0.484	0.010	0.019	0.648	0.021	0.041
UniNER	0-shot	EC	0.468	0.086	0.146	0.657	0.183	0.286
GPT-3.5	0-shot	All	0.648	0.090	0.158	0.885	0.208	0.336
GPT-3.5	1-shot	All	0.637	0.105	0.180	0.878	0.243	0.380
GPT-3.5	2-shot	All	0.631	0.122	0.204	0.870	0.284	0.428
GPT-3.5	3-shot	All	0.586	0.113	0.189	0.874	0.289	0.434
GPT-3.5	0-shot	EE	0.472	0.152	0.230	0.624	0.313	0.417
GPT-3.5	1-shot	EE	0.457	0.243	0.317	0.629	0.499	0.557
GPT-3.5	2-shot	EE	0.449	0.255	0.326	0.618	0.510	0.559
GPT-4	1-shot	EE	0.449	0.473	0.461	0.600	0.853	0.704
GPT-3.5	ft. 100 doc - 5 ep.	EE	0.692	0.621	**0.654**	0.876	0.880	0.878
GPT-3.5	ft. 50 doc - 3 ep.	EE	0.671	0.618	0.643	0.874	0.885	**0.880**

Table B.5: Results of entity extraction for GPT-based prompt on MedMentions Truncated. The evaluation is performed on a subset of 50 documents.

B.1.3 EBM NLP

			Entites			Tokens		
Embedding	**Component**	**Task**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Tok2Vec	EntityRec.	EE	0.516	0.311	0.388	0.663	0.279	0.393
Tok2Vec	EntityRec.	EC	0.532	0.227	0.318	0.683	0.229	0.343
RoBERTa	EntityRec.	EE	0.548	0.431	**0.482**	0.740	0.454	0.563
RoBERTa	EntityRec.	EC	0.536	0.437	**0.482**	0.719	0.490	0.583
MedBERT	EntityRec.	EE	0.566	0.393	0.464	0.765	0.428	0.549
MedBERT	EntityRec.	EC	0.483	0.447	0.464	0.673	0.573	**0.619**
CODER	EntityRec.	EE	0.495	0.428	0.459	0.690	0.499	**0.579**
CODER	EntityRec.	EC	0.462	0.437	0.449	0.654	0.568	0.608

Table B.6: Results of entity extraction for EntityRecognizer on EBM NLP

			Entites			Tokens		
Embedding	**Component**	**Task**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Tok2Vec	SpanCat.(Score)	EE	0.599	0.268	0.370	0.719	0.164	0.267
Tok2Vec	SpanCat.(Score)	EC	0.625	0.212	0.317	0.699	0.193	0.303
Tok2Vec	SpanCat.(Length)	EE	0.597	0.267	0.369	0.718	0.164	0.267
Tok2Vec	SpanCat.(Length)	EC	0.623	0.211	0.315	0.697	0.195	0.304
RoBERTa	SpanCat.(Score)	EE	0.521	0.438	**0.475**	0.717	0.481	0.575
RoBERTa	SpanCat.(Score)	EC	0.528	0.439	0.480	0.680	0.489	**0.569**
RoBERTa	SpanCat.(Length)	EE	0.520	0.437	0.475	0.715	0.482	**0.576**
RoBERTa	SpanCat.(Length)	EC	0.523	0.434	0.474	0.675	0.491	0.569
MedBERT	SpanCat.(Score)	EE	0.479	0.446	0.462	0.680	0.490	0.570
MedBERT	SpanCat.(Score)	EC	0.563	0.423	**0.483**	0.726	0.440	0.548
MedBERT	SpanCat.(Length)	EE	0.476	0.442	0.459	0.676	0.492	0.570
MedBERT	SpanCat.(Length)	EC	0.560	0.420	0.480	0.723	0.444	0.550
CODER	SpanCat.(Score)	EE	0.544	0.378	0.446	0.715	0.374	0.491
CODER	SpanCat.(Score)	EC	0.514	0.421	0.463	0.683	0.452	0.544
CODER	SpanCat.(Length)	EE	0.544	0.379	0.447	0.714	0.375	0.492
CODER	SpanCat.(Length)	EC	0.511	0.419	0.461	0.681	0.454	0.545
RoBERTa	SpanFinder(Score)	EC	0.540	0.382	0.448	0.725	0.336	0.460
RoBERTa	SpanFinder(Length)	EC	0.533	0.378	0.442	0.721	0.338	0.460
MedBERT	SpanFinder(Score)	EC	0.576	0.373	0.453	0.747	0.313	0.441
MedBERT	SpanFinder(Length)	EC	0.568	0.369	0.447	0.742	0.314	0.441
CODER	SpanFinder(Score)	EC	0.508	0.388	0.440	0.710	0.343	0.462
CODER	SpanFinder(Length)	EC	0.506	0.386	0.438	0.707	0.345	0.463

Table B.7: Results of entity extraction for SpanCategorizer on EBM NLP

		Entites			Tokens		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
UniNER	EE 0-shot	0.176	0.016	0.030	0.392	0.067	0.114
UniNER	EC 0-shot	0.193	0.044	0.071	0.364	0.056	0.096
GPT-3.5	0-shot	0.102	0.012	0.021	0.283	0.031	0.057
GPT-3.5	1-shot	0.164	0.167	0.166	0.334	0.334	0.334
GPT-3.5	2-shot	0.180	0.217	0.197	0.353	0.404	0.377
GPT-4	1-shot	0.193	0.192	0.193	0.308	0.542	0.393
GPT-3.5	ft. 100 doc 5 ep. 1-shot	0.338	0.438	0.381	0.451	0.620	0.522
GPT-3.5	ft. 50 doc 3 ep. 1-shot	0.370	0.457	**0.409**	0.483	0.580	**0.527**

Table B.8: Results of entity extraction for GPT-based prompt on EBM NLP. The evaluation is performed on a subset of 50 documents.

B.1.4 I2B2 2010

			Entites			Tokens		
Embedding	**Component**	**Task**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Tok2Vec	EntityRec.	EE	0.933	0.926	0.930	0.957	0.942	0.950
Tok2Vec	EntityRec.	EC	0.853	0.829	0.840	0.905	0.859	0.881
RoBERTa	EntityRec.	EE	0.958	0.954	**0.956**	0.971	0.969	0.970
RoBERTa	EntityRec.	EC	0.914	0.867	**0.890**	0.950	0.876	0.912
MedBERT	EntityRec.	EE	0.956	0.954	0.955	0.973	0.968	**0.971**
MedBERT	EntityRec.	EC	0.900	0.870	0.885	0.935	0.897	**0.916**
CODER	EntityRec.	EE	0.946	0.942	0.944	0.963	0.963	0.963
CODER	EntityRec.	EC	0.869	0.847	0.858	0.920	0.886	0.903

Table B.9: Results of entity extraction for EntityRecognizer on I2B2 2010

			Entites			Tokens		
Embedding	**Component**	**Task**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Tok2Vec	SpanCat.(Score)	EE	0.947	0.882	0.913	0.963	0.840	0.898
Tok2Vec	SpanCat.(Score)	EC	0.877	0.792	0.832	0.925	0.761	0.835
Tok2Vec	SpanCat.(Length)	EE	0.947	0.882	0.913	0.962	0.841	0.897
Tok2Vec	SpanCat.(Length)	EC	0.877	0.792	0.833	0.923	0.763	0.836
RoBERTa	SpanCat.(Score)	EE	0.963	0.927	**0.945**	0.973	0.879	**0.924**
RoBERTa	SpanCat.(Score)	EC	0.913	0.848	0.879	0.953	0.814	0.878
RoBERTa	SpanCat.(Length)	EE	0.962	0.927	0.944	0.973	0.880	0.924
RoBERTa	SpanCat.(Length)	EC	0.913	0.848	0.879	0.952	0.815	0.878
MedBERT	SpanCat.(Score)	EE	0.952	0.928	0.940	0.973	0.877	0.923
MedBERT	SpanCat.(Score)	EC	0.908	0.858	**0.883**	0.953	0.827	**0.886**
MedBERT	SpanCat.(Length)	EE	0.955	0.931	0.943	0.972	0.880	0.924
MedBERT	SpanCat.(Length)	EC	0.906	0.856	0.880	0.951	0.829	0.886
CODER	SpanCat.(Score)	EE	0.954	0.905	0.929	0.971	0.862	0.913
CODER	SpanCat.(Score)	EC	0.894	0.813	0.852	0.942	0.794	0.862
CODER	SpanCat.(Length)	EE	0.955	0.905	0.929	0.970	0.864	0.914
CODER	SpanCat.(Length)	EC	0.897	0.816	0.854	0.941	0.798	0.864
RoBERTa	SpanFinder(Score)	EC	0.908	0.839	0.872	0.945	0.750	0.836
RoBERTa	SpanFinder(Length)	EC	0.907	0.838	0.871	0.944	0.751	0.837
MedBERT	SpanFinder(Score)	EC	0.911	0.836	0.872	0.945	0.751	0.837
MedBERT	SpanFinder(Length)	EC	0.909	0.834	0.870	0.943	0.754	0.838
CODER	SpanFinder(Score)	EC	0.885	0.794	0.837	0.936	0.727	0.818
CODER	SpanFinder(Length)	EC	0.887	0.796	0.839	0.935	0.730	0.820

Table B.10: Results of entity extraction for SpanCategorizer on I2B2 2010

B.1.5 IT dataset

		Entites			Tokens		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Tok2Vec	EntityRec.	0.860	0.867	**0.863**	0.952	0.931	**0.941**
xlm-RoBERTa-base	EntityRec.	0.672	0.672	0.672	0.808	0.817	0.813
UmBERTo	EntityRec.	0.672	0.657	0.664	0.815	0.795	0.805
CODER	EntityRec.	0.668	0.668	0.668	0.799	0.807	0.803

Table B.11: Results of entity extraction for EntityRecognizer on IT dataset

		Entites			Tokens		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Tok2Vec	SpanCat.(Score)	0.876	0.843	0.859	0.960	0.863	0.909
Tok2Vec	SpanCat.(Length)	0.877	0.844	0.860	0.957	0.873	0.913
xlm-RoBERTa-base	SpanCat.(Score)	0.864	0.858	**0.861**	0.954	0.888	0.920
xlm-RoBERTa-base	SpanCat.(Length)	0.862	0.857	0.859	0.949	0.895	**0.921**
UmBERTo	SpanCat.(Score)	0.868	0.719	0.787	0.955	0.775	0.856
UmBERTo	SpanCat.(Length)	0.872	0.723	0.791	0.952	0.785	0.861
CODER	SpanCat.(Score)	0.871	0.843	0.857	0.956	0.863	0.907
CODER	SpanCat.(Length)	0.874	0.846	0.860	0.952	0.870	0.909

Table B.12: Results of entity extraction for SpanCategorizer on IT dataset

		Entites			Tokens		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
UniNER	0-shot	0.667	0.037	0.069	0.917	0.048	0.091
GPT-3.5	1-shot	0.722	0.544	0.620	0.891	0.718	0.795
GPT-4	1-shot	0.790	0.750	0.770	0.858	0.916	0.886
GPT-3.5	ft. 50 doc 3 ep. 0-shot	0.761	0.730	0.745	0.884	0.903	0.893
GPT-3.5	ft. 50 doc 3 ep. 1-shot	0.796	0.772	**0.784**	0.902	0.913	**0.908**

Table B.13: Results of entity extraction for GPT-based on IT dataset.

B.2 Entity Classification

B.2.1 MedMentions

		Weighted avg.		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**
Tok2Vec	EntityRec.	0.361	0.369	0.354
RoBERTa	EntityRec.	0.451	0.485	**0.466**
MedBERT	EntityRec.	0.440	0.463	0.451
CODER	EntityRec.	0.428	0.452	0.438

Table B.14: Results of entity classification for MedMentions using EntityRecognizer

		Weighted avg.		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**
Tok2Vec	SpanCat.(Score)	0.496	0.364	0.417
Tok2Vec	SpanCat.(Length)	0.497	0.365	0.419
RoBERTa	SpanCat.(Score)	0.551	0.478	0.511
RoBERTa	SpanCat.(Length)	0.551	0.478	**0.511**
MedBERT	SpanCat.(Score)	0.505	0.448	0.473
MedBERT	SpanCat.(Length)	0.506	0.449	0.474
CODER	SpanCat.(Score)	0.477	0.444	0.458
CODER	SpanCat.(Length)	0.477	0.445	0.458
Tok2Vec	SpanCat-mix	0.428	0.466	0.444
RoBERTa	SpanCat-mix	0.449	0.493	0.469
MedBERT	SpanCat-mix	0.453	0.493	**0.471**
CODER	SpanCat-mix	0.445	0.487	0.464

Table B.15: Results of entity classification for MedMentions using SpanCategorizer

B.2.2 MedMentions Truncated

		Weighted avg.		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**
Tok2Vec	EntityRec.	0.395	0.382	0.386
RoBERTa	EntityRec.	0.448	0.453	**0.449**
MedBERT	EntityRec.	0.452	0.426	0.438
CODER	EntityRec.	0.396	0.423	0.408

Table B.16: Results of entity classification for MedMentions truncated using EntityRecognizer

		Weighted avg.		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**
Tok2Vec	SpanCat.(Score)	0.473	0.383	0.420
Tok2Vec	SpanCat.(Length)	0.473	0.384	0.420
RoBERTa	SpanCat.(Score)	0.477	0.446	**0.459**
RoBERTa	SpanCat.(Length)	0.476	0.446	0.459
MedBERT	SpanCat.(Score)	0.484	0.418	0.446
MedBERT	SpanCat.(Length)	0.484	0.418	0.446
CODER	SpanCat.(Score)	0.450	0.414	0.429
CODER	SpanCat.(Length)	0.450	0.414	0.429
RoBERTa	SpanFinder(Score)	0.479	0.450	**0.462**
RoBERTa	SpanFinder(Length)	0.479	0.449	0.462
MedBERT	SpanFinder(Score)	0.446	0.433	0.438
MedBERT	SpanFinder(Length)	0.445	0.432	0.437
CODER	SpanFinder(Score)	0.482	0.417	0.446
CODER	SpanFinder(Length)	0.481	0.416	0.445
Tok2Vec	SpanCat-mix	0.349	0.388	0.359
RoBERTa	SpanCat-mix	0.392	0.442	0.414
MedBERT	SpanCat-mix	0.400	0.453	**0.424**
CODER	SpanCat-mix	0.395	0.447	0.419

Table B.17: Results of entity classification for MedMentions truncated using SpanCategorizer

		Weighted avg.		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**
UniNER	0-shot	0.343	0.052	0.090
GPT-3.5	All-in-one 0-shot simple	0.382	0.054	0.095
GPT-3.5	All-in-one 1-shot simple	0.351	0.047	0.083
GPT-3.5	All-in-one 2-shot simple	0.540	0.045	0.083
GPT-3.5	All-in-one 3-shot simple	0.540	0.038	0.071
GPT-3.5	Best EE + 0-shot simple	0.332	0.331	0.320
GPT-3.5	Best EE + 1-shot simple	0.347	0.350	0.339
GPT-3.5	Best EE + 0-shot CoT	0.354	0.305	0.318
GPT-3.5	Best EE + 1-shot CoT	0.335	0.328	0.320

Table B.18: Results of entity classification for MedMentions Truncated. The evaluation is performed on a subset of 50 documents.

B.2.3 EBM

		Weighted avg.		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**
Tok2Vec	EntityRec.	0.425	0.186	0.259
RoBERTa	EntityRec.	0.369	0.316	**0.337**
MedBERT	EntityRec.	0.321	0.301	0.310
CODER	EntityRec.	0.302	0.288	0.294

Table B.19: Results of entity classification for EBM NLP using EntityRecognizer

		Weighted avg.		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**
Tok2Vec	SpanCat.(Score)	0.433	0.187	0.250
Tok2Vec	SpanCat.(Length)	0.431	0.186	0.249
RoBERTa	SpanCat.(Score)	0.353	0.315	0.326
RoBERTa	SpanCat.(Length)	0.349	0.310	0.321
MedBERT	SpanCat.(Score)	0.406	0.317	**0.355**
MedBERT	SpanCat.(Length)	0.404	0.315	0.352
CODER	SpanCat.(Score)	0.348	0.299	0.317
CODER	SpanCat.(Length)	0.346	0.297	0.315
RoBERTa	SpanFinder(Score)	0.396	0.287	0.333
RoBERTa	SpanFinder(Length)	0.391	0.283	0.328
MedBERT	SpanFinder(Score)	0.439	0.292	**0.350**
MedBERT	SpanFinder(Length)	0.432	0.287	0.344
CODER	SpanFinder(Score)	0.363	0.281	0.317
CODER	SpanFinder(Length)	0.361	0.280	0.315
Tok2Vec	SpanCat-mix	0.383	0.308	0.339
RoBERTa	SpanCat-mix	0.390	0.316	0.346
MedBERT	SpanCat-mix	0.390	0.316	**0.346**
CODER	SpanCat-mix	0.389	0.314	0.344

Table B.20: Results of entity classification for EBM NLP using SpanCategorizer

		Weighted avg.		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**
UniNER	0-shot	0.062	0.028	0.039
GPT-3.5	0-shot simple	0.399	0.255	0.307
GPT-3.5	1-shot simple	0.414	0.304	0.345
GPT-3.5	0-shot CoT	0.391	0.246	0.292
GPT-3.5	1-shot CoT	0.399	0.273	0.313

Table B.21: Results of entity classification for EBM NLP. The evaluation is performed on a subset of 50 documents.

B.2.4 I2B2

		Weighted avg.		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**
Tok2Vec	EntityRec.	0.724	0.704	0.714
RoBERTa	EntityRec.	0.825	0.783	**0.804**
MedBERT	EntityRec.	0.807	0.779	0.793
CODER	EntityRec.	0.745	0.726	0.736

Table B.22: Results of entity classification for I2B2 2010 using EntityRecognizer

		Weighted avg.		
Embedding	**Component**	**Prec.**	**Rec.**	**F1**
Tok2Vec	SpanCat.(Score)	0.773	0.698	0.733
Tok2Vec	SpanCat.(Length)	0.774	0.699	0.734
RoBERTa	SpanCat.(Score)	0.829	0.769	0.798
RoBERTa	SpanCat.(Length)	0.828	0.769	0.797
MedBERT	SpanCat.(Score)	0.823	0.779	0.800
MedBERT	SpanCat.(Length)	0.820	0.776	0.797
CODER	SpanCat.(Score)	0.798	0.725	0.760
CODER	SpanCat.(Length)	0.802	0.729	0.764
RoBERTa	SpanFinder(Score)	0.820	0.757	0.787
RoBERTa	SpanFinder(Length)	0.818	0.755	0.785
MedBERT	SpanFinder(Score)	0.829	0.761	0.794
MedBERT	SpanFinder(Length)	0.825	0.758	0.790
CODER	SpanFinder(Score)	0.784	0.703	0.741
CODER	SpanFinder(Length)	0.787	0.705	0.744
Tok2Vec	SpanCat-mix	0.876	0.873	0.874
RoBERTa	SpanCat-mix	0.878	0.877	0.878
MedBERT	SpanCat-mix	0.891	0.890	**0.891**
CODER	SpanCat-mix	0.877	0.876	0.877

Table B.23: Results of entity classification for I2B2 2010 using SpanCategorizer

B.3 Dataset Scaling

B.3.1 MedMentions Truncated

Embedding	DS size	Task	Entities F1	Tokens F1
RoBERTa-base	20	EE	0.579	0.795
RoBERTa-base	40	EE	0.589	0.788
RoBERTa-base	50	EE	0.583	0.765
RoBERTa-base	75	EE	0.603	0.794
RoBERTa-base	100	EE	0.624	0.804
RoBERTa-base	150	EE	0.620	0.780
RoBERTa-base	200	EE	0.619	0.782
RoBERTa-base	250	EE	0.619	0.777
RoBERTa-base	300	EE	0.615	0.774
RoBERTa-base	400	EE	0.613	0.782
RoBERTa-base	600	EE	0.634	0.781
RoBERTa-base	800	EE	0.625	0.770
RoBERTa-base	1000	EE	0.642	0.802
RoBERTa-base	1300	EE	0.657	0.814
RoBERTa-base	1600	EE	0.639	0.784
RoBERTa-base	1900	EE	0.637	0.774

Table B.24: Results of entity extraction for EntityRecognizer trained for entity extraction on MedMentions truncated and cutted

Embedding	DS size	Task	Entities F1	Tokens F1	Weighted avg.
RoBERTa-base	20	EC	0.544	0.754	0.262
RoBERTa-base	40	EC	0.556	0.754	0.314
RoBERTa-base	50	EC	0.565	0.751	0.327
RoBERTa-base	75	EC	0.560	0.736	0.348
RoBERTa-base	100	EC	0.588	0.762	0.362
RoBERTa-base	150	EC	0.582	0.734	0.379
RoBERTa-base	200	EC	0.598	0.750	0.397
RoBERTa-base	250	EC	0.596	0.750	0.393
RoBERTa-base	300	EC	0.584	0.722	0.400
RoBERTa-base	400	EC	0.582	0.718	0.409
RoBERTa-base	600	EC	0.599	0.728	0.420
RoBERTa-base	800	EC	0.608	0.735	0.428
RoBERTa-base	1000	EC	0.613	0.736	0.432
RoBERTa-base	1300	EC	0.629	0.774	0.434
RoBERTa-base	1600	EC	0.624	0.760	0.438
RoBERTa-base	1900	EC	0.606	0.721	0.449

Table B.25: Results of entity extraction for EntityRecognizer trained for entity classification on MedMentions truncated and cutted

B.3.2 EBM NLP

Embedding	DS size	Task	Entities F1	Tokens F1
RoBERTa-base	20	EE	0.361	0.599
RoBERTa-base	40	EE	0.386	0.556
RoBERTa-base	50	EE	0.393	0.602
RoBERTa-base	75	EE	0.407	0.590
RoBERTa-base	100	EE	0.415	0.570
RoBERTa-base	150	EE	0.426	0.576
RoBERTa-base	200	EE	0.429	0.606
RoBERTa-base	250	EE	0.422	0.597
RoBERTa-base	300	EE	0.420	0.596
RoBERTa-base	400	EE	0.424	0.592
RoBERTa-base	600	EE	0.434	0.624
RoBERTa-base	800	EE	0.423	0.574
RoBERTa-base	1000	EE	0.453	0.573
RoBERTa-base	1300	EE	0.455	0.641
RoBERTa-base	1600	EE	0.475	0.592
RoBERTa-base	1900	EE	0.468	0.600

Table B.26: Results of entity extraction for EntityRecognizer trained for entity extraction on EBM NLP cutted

Embedding	DS size	Task	Entities F1	Tokens F1	Weighted avg.
RoBERTa-base	20	EC	0.368	0.509	0.236
RoBERTa-base	40	EC	0.397	0.604	0.221
RoBERTa-base	50	EC	0.408	0.580	0.259
RoBERTa-base	75	EC	0.396	0.566	0.244
RoBERTa-base	100	EC	0.403	0.607	0.234
RoBERTa-base	150	EC	0.405	0.584	0.260
RoBERTa-base	200	EC	0.416	0.629	0.245
RoBERTa-base	250	EC	0.430	0.602	0.273
RoBERTa-base	300	EC	0.430	0.608	0.270
RoBERTa-base	400	EC	0.432	0.630	0.266
RoBERTa-base	600	EC	0.449	0.533	0.321
RoBERTa-base	800	EC	0.449	0.627	0.270
RoBERTa-base	1000	EC	0.463	0.577	0.323
RoBERTa-base	1300	EC	0.454	0.629	0.282
RoBERTa-base	1600	EC	0.479	0.653	0.307
RoBERTa-base	1900	EC	0.466	0.632	0.306

Table B.27: Results of entity extraction for EntityRecognizer trained for entity classification on EBM NLP cutted

Appendix C

Others

C.1 Color palette

In Chapter 6 the colors for the graph are for colorblind. The colors are picked from a paper by Bang Wong[37].

The elements in the graph are mapped with these colors:

Element	Color	HEX
GPT-3.5	dark grey	#525975
GPT-4	orange	#E69F00
GPT-3.5 fine-tuned	bright azure	#56B4E9
UniNER	pink	#CC79A7
GPT-3.5 EC simple	blue	#0072B2
GPT-3.5 EC CoT	bright orange	#D55E00
GPT-3.5 All-in-one	yellow	#F0E442
EntityRecognizer	yellow	#F0E442
SpanCategorizer	blue	#0072B2
SpanFinder	pink	#CC79A7
SpanCategorizer Mix	bright orange	#D55E00
RoBERTa-base and xml-RoBERTa-base	yellow	#F0E442
MedBERT	blue	#0072B2
CODER	bright orange	#D55E00
UmBERTo	pink	#CC79A7
Entities	yellow	#F0E442
Tokens	blue	#0072B2
Training loss	yellow	#F0E442
Training accuracy	bright orange	#D55E00
Validation loss	blue	#0072B2
Validation token accuracy	pink	#CC79A7

Table C.1: Map of elements in the graph to color

C.2 GPT-3.5 fine-tuning loss

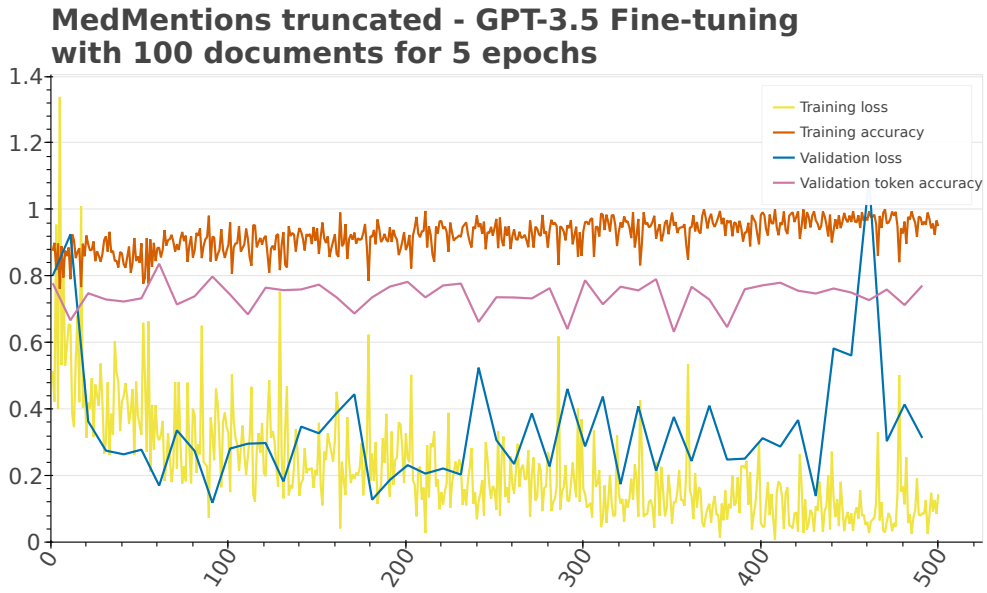

Figure C.1: MedMentions truncated fine-tuning loss and accuracy using 100 documents for 5 epochs

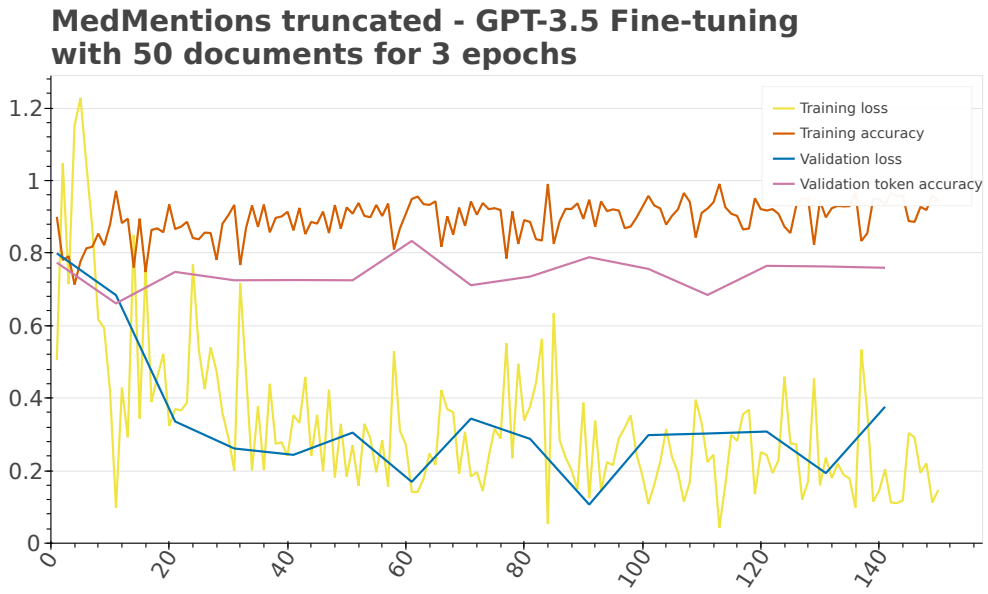

Figure C.2: MedMentions truncated fine-tuning loss and accuracy using 50 documents for 3 epochs

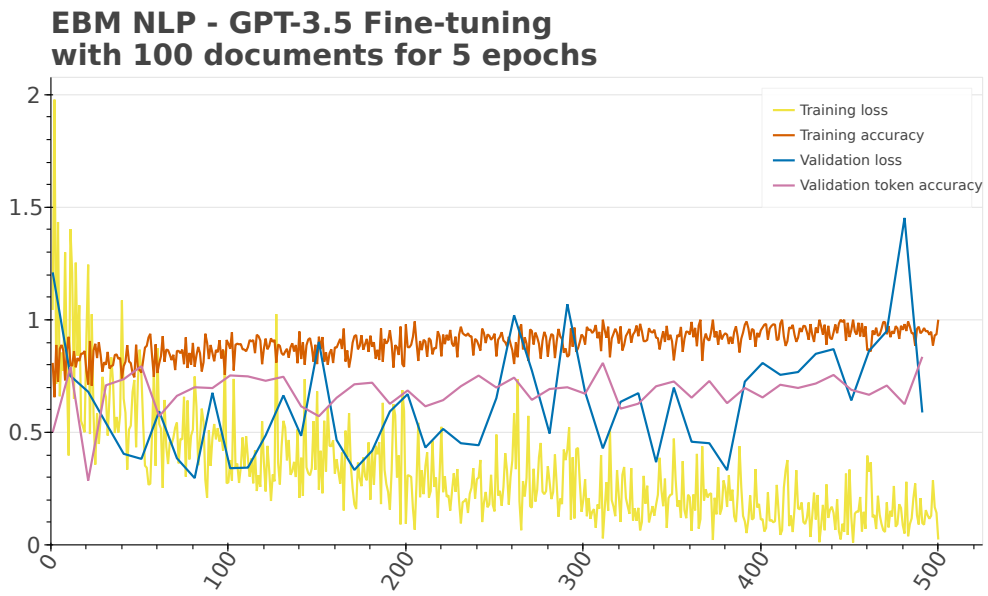

Figure C.3: EBM NLP fine-tuning loss and accuracy using 100 documents for 5 epochs

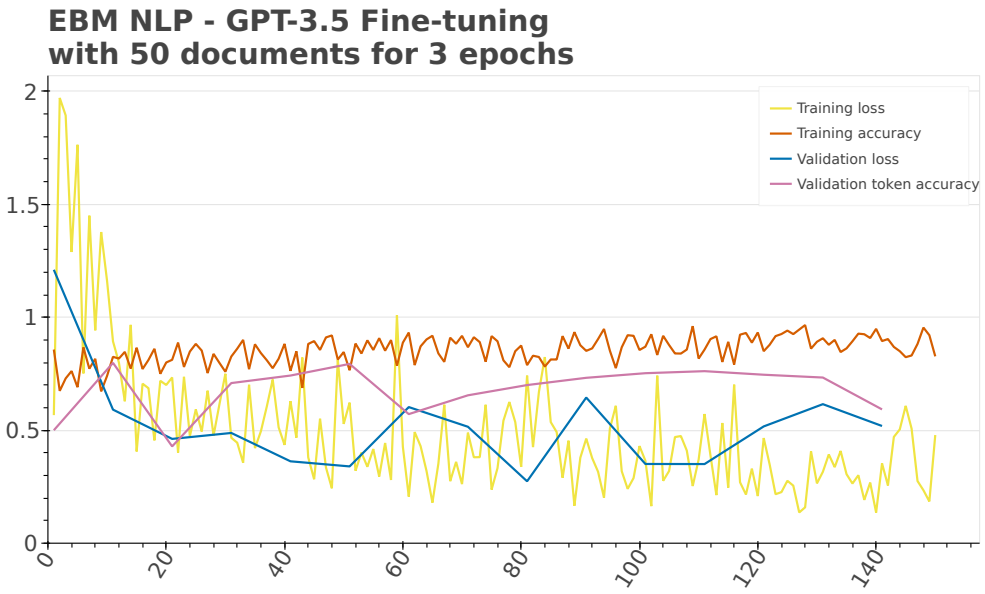

Figure C.4: EBM NLP fine-tuning loss and accuracy using 50 documents for 3 epochs

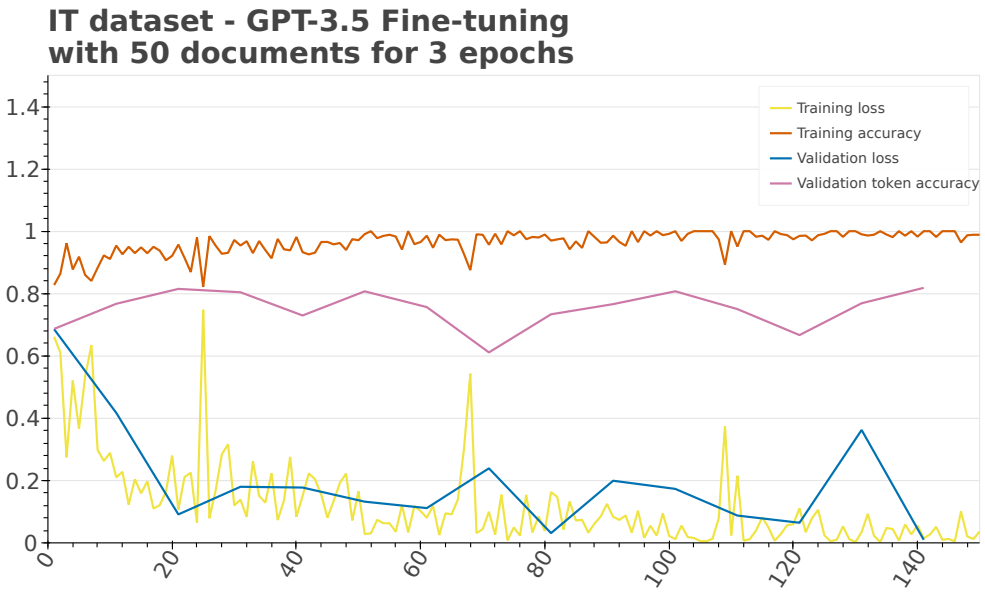

Figure C.5: IT dataset fine-tuning loss and accuracy using 50 documents for 3 epochs

C.3 Machine specification

The majority of training has been made on a single machine offered by MAPS. S.P.A.

In Table C.3 are reported the main characteristic of the machine.

Specification	**Value**
Operating System	Ubuntu 16.04
CPU	Intel(R) Xeon(R) CPU E5-2620 v4
Number of Cores	8
Clock Speed (GHz)	2.10GHz
Installed RAM	94GB - 2133 MHz
Graphics Card	NVIDIA TITAN Xp
CUDA Version	11.3

Table C.2: Machine Specifications