



JOSÉ MANUEL MARQUES FERRÃO

Master in Management
Bachelor in Law

DEMOCRATIZING LEGAL UNDERSTANDING

A MODEL FOR SIMPLIFYING JUDICIAL DECISIONS

MASTER IN ANALYSIS AND ENGINEERING OF BIG DATA

NOVA University Lisbon
September, 2024



DEMOCRATIZING LEGAL UNDERSTANDING

A MODEL FOR SIMPLIFYING JUDICIAL DECISIONS

JOSÉ MANUEL MARQUES FERRÃO

Master in Management
Bachelor in Law

Adviser: João Magalhães
Full Professor, NOVA University Lisbon

Co-adviser: David Semedo
Assistant Professor, NOVA University Lisbon

Democratizing Legal Understanding

A Model for Simplifying Judicial Decisions

Copyright © José Manuel Marques Ferrão, NOVA School of Science and Technology, NOVA University Lisbon.

The NOVA School of Science and Technology and the NOVA University Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

ACKNOWLEDGEMENTS

I would like to extend my deepest gratitude to my supervisors, Professor João Magalhães and Professor David Semedo, for their unwavering guidance, support, and expertise throughout the course of my research. Their insightful feedback, encouragement, and commitment to excellence have been instrumental in the completion of this thesis.

I would also like to express my sincere thanks to my family for their support and encouragement throughout this journey. Your understanding and patience have been greatly appreciated and have provided me with the strength to see this work through to completion.

ABSTRACT

This thesis presents the development of an automated model designed to process legal decisions and generate concise descriptors that summarize the most relevant aspects of each case. The aim is to facilitate efficient legal research by providing a set of descriptors that encapsulate the key facts, legal issues, and rulings contained within each decision. This approach addresses the growing need for automated solutions in the legal domain, where large volumes of unstructured data can be difficult to process manually. The model's performance was evaluated through qualitative assessments which demonstrate its potential utility for legal professionals and institutions seeking to optimize their decision analysis workflows. This research contributes to the intersection of the legal domain, natural language processing and big data, offering a framework for enhancing the accessibility and usability of legal information.

Keywords: Legal LLM, Legal Named Entity Recognition, Legal Literacy, Legal Interpretation, Text Extraction

RESUMO

Esta tese apresenta o desenvolvimento de um modelo automatizado concebido para processar decisões judiciais e gerar descritores concisos que resumem os aspetos mais relevantes de cada caso. O objetivo é facilitar a pesquisa jurídica eficiente, fornecendo um conjunto de descritores que encapsulam os factos chave, as questões jurídicas e as decisões contidas em cada caso. Esta abordagem responde à crescente necessidade de soluções automatizadas no domínio jurídico, onde grandes volumes de dados não estruturados podem ser difíceis de processar manualmente. O desempenho do modelo foi avaliado através de métodos qualitativos, que demonstram o seu potencial para profissionais e instituições jurídicas que procuram otimizar os seus fluxos de análise de decisões. Esta investigação pretendeu realizar contributos em diversos domínios do conhecimento, designadamente na interseção entre a área jurídica, o processamento de linguagem natural e o tratamento de grande volume de dados, oferecendo um quadro para melhorar a acessibilidade e usabilidade da informação jurídica.

Palavras-chave: LLM Legal, Reconhecimento de Entidades Legais, Literacia Jurídica, Interpretação Jurídica, Extração de Texto

CONTENTS

List of Figures	vii
Glossary	ix
Acronyms	xi
1 Introduction	1
1.1 Contextualization	1
1.2 Problem to Solve and Motivation	1
1.3 Objective and Research Challenges	3
2 Literature Review	6
2.1 The Turing Test	6
2.2 The Syntactic Era	7
2.3 The Statistical Era	8
2.4 The Rise of Machine Learning	9
2.5 The Deep Learning Revolution	9
2.5.1 The Transformer Model	11
2.5.2 Decoder	19
2.5.3 Encoder-Decoder Functioning	20
3 Related Work	22
3.1 Large Language Models	22
3.1.1 Bidirectional Encoder Representation from Transformers	23
3.1.2 Sentence-BERT	24
3.1.3 Generative Pre-Trained Transformer Decoder	24
3.1.4 The T-5 Model (Encoder-Decoder)	26
3.1.5 The Adaptability of the Transformers Model	27
3.2 SAILER: Structure-aware Pre-trained Language Model for Legal Case Retrieval	27

3.3	Named Entity Recognition in the Legal Domain	30
4	Design Specification of a Legal Search Framework	32
4.1	Task Definition	32
4.2	Architecture	33
4.3	Named Entity Recognition	33
4.4	Annotating Legal Descriptors	34
4.5	Evaluation	35
5	Generation of Legal Descriptors	36
5.1	Introduction and base model	36
5.2	Datasets and pretraining	37
5.3	Training GlorIA on the Legal Domain	37
5.3.1	Model 1	37
5.3.2	Model 2	39
5.3.3	Model 3	40
5.3.4	Model 4	41
5.4	Model Derivatives	42
5.5	Results and Discussion	43
5.6	Take away lessons	45
5.6.1	Utilizing Specialized Legal Texts	46
5.6.2	Leveraging GlorIA for Legal Language Processing	46
5.6.3	Addressing the Initial Objectives Through Alternative Means	47
5.6.4	Conclusion	47
6	Conclusions and Future Work	48
6.1	Overview of the Work Conducted	48
6.2	Reasons for Abandoning the Initial Work Proposal	48
6.2.1	Technical Challenges	49
6.2.2	Legal Data	49
6.2.3	Feasibility and Time Constraints	49
6.3	Challenges and Limitations	50
6.4	Contributions	51
6.5	Future Work	52
6.6	Final Remarks	53
	Bibliography	54

LIST OF FIGURES

2.1	The Transformer architecture	11
2.2	(left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.	16
2.3	Orchestrating the encoder-decoder model.	20
3.1	The legal structure of documents	28
4.1	Proposed Architecture	33
5.1	A sample of one of the valid decisions extracted	38
5.2	A sample of one of the invalid decisions extracted	38
5.3	Evolution of the training loss, as a function of the training steps	42

LISTINGS

5.1 A sample of the descriptors generated	43
---	----

GLOSSARY

Commented Legislation	A body of laws with annotations, explanations, or commentary provided by legal experts. These comments help to provide context and to clarify the meaning of laws. In some cases, examples or references to case law may be included. (pp. 3, 4, 32, 34, 42, 45, 46, 48–50)
Diário da República	An official online publication of legal and administrative documents in Portugal, accessible through https://diariodarepublica.pt/ . It provides up-to-date information on legislative acts, regulations, and other official announcements. (p. 42)
Feed-Forward Layer	A neural network architecture where connections between nodes do not form cycles, characterized by a sequential flow of information from the input layer, through one or more hidden layers, to the output layer without any backward or recurrent connections. (pp. 18, 19)
Neural Network	A computational model inspired by the structure of biological neural networks, consisting of interconnected nodes or "neurons" organized in layers, which processes information through adaptive connections to perform tasks like classification, regression, or pattern recognition. (pp. 12, 13)
Positional Embedding	A numerical representation that captures the position of elements in a sequence, often used in models to provide contextual information about sequence order. (p. 11)

Self-Attention	A mechanism in deep learning that weighs the relevance of different parts of a sequence to a particular element, enabling models to consider global context when processing sequential data. (pp. 10, 14–16)
Subwords	Units of language that are smaller than words but larger than individual characters, often derived by breaking down words based on morphological boundaries or frequent patterns. (p. 11)
Transformer	A deep learning architecture primarily characterized by its use of self-attention mechanisms, allowing it to handle sequential data without relying on recurrent layers. (pp. 10–12, 14–16, 19, 20, 23, 25, 26)
Word Embedding	A numerical representation of words in a continuous vector space, capturing semantic relationships and used to improve machine learning models on text data. (p. 11)

ACRONYMS

AI	Artificial Intelligence (<i>pp. 4, 5, 7, 52, 53</i>)
BERT	Bidirectional Encoder Representations from Transformers (<i>pp. 10, 23–27, 29, 31, 34, 49</i>)
CBOW	Continuous Bag of Words (<i>p. 12</i>)
CoT	Chain-of-Thought (<i>pp. 2, 25, 26</i>)
GPT	Generative Pre-trained Transformer (<i>pp. 10, 24–27</i>)
LLM	Large Language Model (<i>pp. 2, 3, 22, 23, 34, 36, 45, 47, 48, 51, 53</i>)
LSTM	Long Short-Term Memory (<i>pp. 10, 31</i>)
MLM	Mask Language Modeling (<i>p. 29</i>)
NER	Named Entity Recognition (<i>pp. 9, 31, 33, 34, 45–47, 49, 50, 52</i>)
NLP	Natural Language Processing (<i>pp. 1, 2, 6, 7, 9, 10, 12, 22, 24–27, 46, 47, 51, 53</i>)
ReLU	Rectified Linear Unit (<i>p. 19</i>)
RNNs	Recurrent Neural Networks (<i>pp. 9, 10</i>)
SAILER	Structure-aware Pre-trained Language Model for Legal Case Retrieval (<i>pp. 28–30</i>)
SBERT	Sentence-BERT (<i>p. 24</i>)
STJ	Supremo Tribunal de Justiça (the Supreme Court of Justice in Portugal) (<i>pp. 32, 33, 37, 42, 46, 48</i>)
SVMs	Support Vector Machines (<i>p. 9</i>)

INTRODUCTION

1.1 Contextualization

Human communication is extremely complex and remarkably diversified. It takes a vast amount of time to learn a single language and yet, throughout history, people have learnt to communicate in many different ways, both written and verbally. Paradoxically to this degree of complexity, the ability to communicate is an intrinsic trace of humans and thus it is one of the first things that we learn to do in life [29, 47].

Given its importance, it is no surprise that computer scientists have tried to teach machines to understand the human language for decades. As we will see later in this work, the first attempts suffered from profound limitations at different levels which severely affected their overall performance [31]. Despite some progresses made in the field, until very recently computers were not able to manipulate human language in a similar way to that of humans. Notwithstanding the above, some very recent developments have completely changed the landscape for language models. The advancements are so profound that they are set to completely transform not only the field of [Natural Language Processing \(NLP\)](#), but the entire functioning of society, with some jobs even threatened to be extinguished in the next few years [46, 2].

As a consequence of the magnitude of these advancements, people have stopped asking if language models will one day be capable of “speaking” like humans to wondering how can they be effectively useful. If we assume that language models are capable of mastering the language at the level of humans, then one obvious application they should have is to reliably develop the tasks that require trustworthy use of language. One such group of tasks can be found in the field of law. This brings us to the motivation to develop this particular project, which will be discussed in the next section.

1.2 Problem to Solve and Motivation

To this day, the practice of law is still very inefficient. When faced with a particular case, a lawyer, a judge, a public prosecutor, a solicitor, or any other professional must, first

and foremost, read the actual text of the law. On top of knowing what is actually written in the law, another essential component is to understand how the law is and has been applied by the courts. To achieve this, another stepping stone must be surpassed: the reading of case law, that is, **of previous decisions taken by the courts in an analogous situation to that under consideration**. If reading case law decisions is already a time consuming process, especially in Portugal where the length of those decisions tends to be quite long, searching for the decisions that are actually relevant is, in itself, a task that can occupy several valuable hours of work. But even after reading the law, spending hours searching for decisions similar to the case in-hands and reading those decisions, the knowledge-gaining process is still not over. In most cases, law professionals are also required to consider the so called “doctrine”. These are the texts written by, for instance, reputed law professors, often cited by the courts, which help to better understand the law and how different articles articulate amongst each other.

Interestingly, a branch of [NLP](#) has a lot of similarities with how legal decisions are written. [Chain-of-Thought \(CoT\)](#) is a technique in [NLP](#) that significantly enhances the multi-step reasoning capabilities of [Large Language Model \(LLM\)](#)s. It involves explicitly encouraging the models to generate intermediate rationales or reasoning steps while solving a problem, providing a series of logical considerations that lead to the final answer. This method has shown to dramatically improve the performance of [LLMs](#) on complex reasoning tasks by making the reasoning process more transparent and structured [65, 64]. Similarly, when a judge writes a legal decision, there is the need to spell out all the intermediate factors that led him to took the decision. This kind of relation between how decisions are written and how a given model works could potentially work in our favor when developing a [NLP](#) model for the legal domain.

Apart from these considerations, it seems undeniable that the practice of law is associated with cumbersome and time consuming processes, where the consultation of a multitude of different sources is a requirement impossible to circumvent. This has a negative impact in the productivity of professionals, diverting the attention from what should actually be the true focus of their work: gain more knowledge about the meaning of the law and how it is applied in practice.

From a second perspective, one must also consider how non-professional people interact with the law. A crucial cornerstone to be considered at this instance is that everyone is required to know the law¹. In fact, there is a very common Portuguese expression that goes as follows: “o desconhecimento da lei não aproveita a ninguém” - in a loose translation, this means that even if you do not know the law, you are still required to act as provided by the law. This is, without doubt, very paradoxical. How can people be forced to behave in a specific way if they do not even know that they should behave like that in the first place? Of course, for some actions it is intuitively obvious what is the only appropriate behaviour: a person cannot, except in some very particular situations,

¹At least in Portugal, this is an actual legal duty of citizens, as prescribed by the Portuguese Civil Code in its Article 6.

harm or kill another person, or steal from a third party. No one has doubts about this and it will be unequivocally false if someone stated otherwise. Notwithstanding, some other cases are actually not clear at all. For example, in matters related to taxes, it is common for heavy fines to be applied if someone does not submit a given tax form (even if the person did not know that the document must have been submitted in the first place). In other situations, there is the possibility that someone actually wants to know more about a specific law topic that might affect him, but the understanding of the legal framework is simply not possible to grasp for that person, given, for instance, the dense and often inaccessible legal jargon used by professionals, or the labyrinthine nature of the legal system itself, where cross referencing between different articles, regulations and statutes is a common practice.

Whether we are considering a professional that needs to get valuable insights efficiently, or simply a regular citizen who wants to understand what are the rights and obligations that affect him in a particular case, the need to improve access to legal knowledge is clear and urgent. The advent of new technologies that have enabled the rise of the most recent language models created new and unprecedented opportunities to make progress in this field. It is our intention and the goal of this project to make contributions in this space, increasing and facilitating the access to legal literacy.

1.3 Objective and Research Challenges

During this work, we had one main objective: **to train a European-Portuguese LLM in the legal domain**. Our approach met several different challenges, which we will briefly address below.

We started our work with a very clear and defined goal: to identify within the text of courts' decisions the legal knowledge which is implicitly present and cross reference that implicit knowledge with the text of the law and of other decisions to clarify the legal foundations behind the courts' rulings. In our initial approach, this knowledge will be extracted from [Commented Legislation](#).

Due to the pivotal importance of [Commented Legislation](#) in the legal world, an explanation of this concept must be given. [Commented Legislation](#), often referred to as annotated legislation, is a critical resource in the legal field that combines statutory law with expert commentary, annotations, and references to relevant case law, legal precedents, and literature articles. It focuses on providing a comprehensive approach to understanding legislation by offering a multifaceted view of the law, elucidating its meaning, application, and implications of statutory provisions. The legal texts, by their nature, are drafted in language that is meant to be precise and formal, which can sometimes render them inaccessible to non-experts and even to professionals looking for practical interpretations. [Commented Legislation](#) bridges this gap by providing in-depth explanations, historical context, and practical examples that illuminate the intent behind the law and its real-world

application. For these reasons, [Commented Legislation](#) is viewed as an indispensable tool for judges, lawyers, students and, in general, any legal practitioner.

Having a system that could cross-reference courts' decisions with the knowledge extracted from [Commented Legislation](#) would not only facilitate the work of legal professionals, as it would also contribute to educate the general public. On what concerns professionals, from Section 1.2 we have already mentioned how inefficient the search for legal insights can be. By using a model of this nature, that inefficiency could be significantly reduced by considering three sub objectives:

- first, professionals could more easily identify the relevant articles and legal statutes that the court took into consideration, which would reduce the time they take to search for that information, in the cases where it is not expressly mentioned;
- second, having a system which could pinpoint professionals to other relevant decisions as the text unfolds would allow them to read only the case law that is actually relevant to them, which could notably undercut the time they take to search for analogous decisions;
- finally, the knowledge gathered from [Commented Legislation](#) would allow for the legal professionals to actually know more about the law, their evolution, their interpretation and how it has been applied before

Also, for non-professional users, we believe it would be of value to benefit from similar features, especially in the cases where the court ruling actually affects them, as they would be able to more easily understand why the decision was taken in a certain way, even without the help of a lawyer.

However, we soon came to the conclusion that this objective was too ambitious to be effectively achieved within the time period available to perform this work. Instead, we realized that several intermediate steps needed to be completed before a model could actually be able to grasp the legal knowledge with the depth required to satisfy our first objective. Once this was apparent, we tried to break our problem into smaller pieces and re-defined the primary goal of this work: **to develop a model that could generate descriptors for a given decision which would encapsulate the meaning of the subjects discussed therein**. Other experimental tasks were also performed that shall be discussed in more detail in Chapter 5.

Having clarified the objective of this work, we will now address some of the challenges already identified in the literature. The idea of using [Artificial Intelligence \(AI\)](#) to help lawyers or lay people to better understand the law is not a new one. In [24] a semi-automatic system of identification and application of amendments to Greek legislative documents is proposed. In [36], a collection for automatic mining of arguments was created, based on German legal case documents. Similarly, in [37] a system is proposed for answering legal research questions based on Dutch cases, modeled through graphs. To

improve the automatic extraction of information from legal documents, in [13] an active learning approach is used to deal with the scarcity of annotated data. In [14], a canonical modeling of question-answer pairs from testimonial transcripts between a lawyer and deponent is proposed, with the objective of summarizing the most important facts and answers.

However, case law corresponds to raw texts which are inherently noisy. On what specifically concerns the field of law, the need to have factual consistency and high levels of accuracy when performing cross-referencing tasks with the articles and legislation that grounds decisions is a notably sensible topic. For instance, if a system would indicate as the basis of a relevant part of a decision an article or statute which is outdated and has actually been revoked, this could have devastating consequences if an appeal is to be made by a lawyer taking into account this wrong information provided the AI agent.

It is our ambition to innovate the computational capacity of understanding legislative texts with the latest deep neural network techniques [60], [1]. Specifically, the Transformer architecture [21], which addressed the limitations of previously existing models when it comes to modeling complex text sequences, such as a dialogue, has enabled the learning of highly capable language models. Based on Transformers, the BERT architecture [4] set a new state of the art in standard text comprehension tasks (e.g., question answering, sentiment analysis and others), and was quickly adopted for text comprehension in other domains, such as scientific and biomedical [39] documents, showing promise in the legal domain as well. Specifically, in [23] the authors note that the BERT model fine-tuned on a collection of legal documents results in a highly effective model. Also in [55], the BERT model is used for text modeling at the paragraph level and it sets the state of the art on the COLIEE 19 dataset for the task of searching for legal documents from the federal court of Canada. In [48], the authors surpass the state of the art using Transformers for the task of detecting implications in legal texts. Thus, we believe that, on what concerns analysis of case law, a transformer architecture can be more successful than previous models.

LITERATURE REVIEW

In this chapter, we aim at providing an historical overview of the works that have been conducted in the area of [NLP](#), from the foundation of the field to the most recent advancements in this domain.

2.1 The Turing Test

“I propose to consider the question, "Can machines think?"”[61]. This was the famous opening written by Alan Touring in 1950, on a paper that can be thought of as the foundation of [NLP](#). In [61], Turing contemplated how to effectively address the question. He posited that if the response depended purely on popular understanding, the optimal approach would be a statistical survey, soliciting opinions from the general public, which the author considered to be absurd.

To circumvent this, Touring proposed an alternative, unambiguous way that could rigorous help to conclude if a machine could think, which he named famously named as the imitation game. In this game, three persons were involved, a man, a woman and an interrogator, which could be of either sex, and was physically separated from the man and the woman. The objective of the game would be for the interrogator to ask questions to the man and the woman in order to determine which was the man and which was the women. The questions should be asked and answered in a written way, so that the interrogator did not get any clues from the tone of the voices of the person who was answering (or, alternatively, an intermediary would move between the two rooms). The role of the man was to deceive the interrogator, leading him to think that he was the woman. The role of the woman was to help the interrogator.

With this set-up in mind, suppose now that the man was replaced by a machine. In this circumstance, the original question “Can machines think?” could be replaced by an objective, unambiguous question “Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman?”[61].

The imitation game was not actually performed by Touring. It was a proposal of

a conceptual framework that could steer the discussion from ambiguous definition of “machines” and “think” towards a concrete, measurable criterion for machine intelligence. It also represented a milestone for NLP as it defined machine intelligence through the use of language, setting the stage for subsequent developments in conversational AI and language understanding algorithms. At the same time, it also created a benchmark for progress, as passing the Turing Test was seen as a kind of holy grail for researchers and scientists.

2.2 The Syntactic Era

Spanning in the 1970s, the first approaches to NLP are usually grouped in what is known as the syntactic era, where models were heavily rule-based. This means that computers were programmed to follow explicit instructions, and there they did not actually learn anything from data, as it happens in modern machine learning and deep learning approaches.

In 1957, the work of Noam Chomsky “Syntactic Structures” [17] introduced the transformational-generative grammar concept. Though earlier than the syntactic era *per se*, Chomsky’s framework would cast a long shadow, influencing many subsequent parsing systems and setting a foundation for the study of formal grammars.

In the 1960s, the field began to see computational applications of these linguistic theories. For instance, in [35] the authors ventured into automating the morphological analysis of English, utilizing rule-based methods to understand the internal structure of words. Three years later, in 1966, Tadao Kasami paper brought forth the Kasami algorithm, an important parsing method for context-free grammars, which he proposed in [34].

By 1970, new parsing techniques were emerging that could tackle more complex language structures. In that year, Jay Earley published “An efficient context-free parsing algorithm” [22] that described the Earley parser, which was groundbreaking for being able to handle any context-free grammar. Also in 1970, a paper from William A. Woods “Transition network grammars for natural language analysis” [68] introduced the idea of Augmented Transition Networks, serving as a type of state machine, quickly became popular tools for parsing natural languages.

In 1972, Terry Winograd “Understanding Natural Language” [67] took the field in a slightly different direction, focusing on the practical application of these theories. In it, he described the SHRDLU system, which simulated a robot embedded in a world of toy blocks. As described in [32] “The program was able to accept natural-language text commands (*Move the red block on top of the smaller green one*) of an unseen complexity and sophistication. His system was also the first attempt to build an extensive (for the time) grammar of English, based on Halliday’s systemic grammar.” This piece highlighted the potential of rule-based NLP, while also pointing out its limitations.

By the end of the decade, in 1978, by Mitchell P. Marcus published his PhD thesis, “A Theory of Syntactic Recognition for Natural Language” [44] which made significant strides by advocating for the use of feature structures’ constraints in parsing to represent

both syntactic and semantic information. This approach hinted at the direction the field would take in the coming years, leaning more towards feature-based grammars.

2.3 The Statistical Era

Emerging in the 1980s and solidifying its dominance through the 1990s, the Statistical Era was characterized by the shift from rule-based systems to probabilistic models. The evolution was propelled by increased computational power, growing data availability, and the realization that capturing the nuances of human language required more than deterministic rules.

At the forefront of this era was the innovative use of n-grams in language modeling. Pioneering works, such as those by Church and Gale. In [18], the authors highlighted the critical role of n-grams in predicting word sequences. In [15], a language model is described as “a probability distribution $p(s)$ over strings s that describes how often the string s occurs as a sentence in some domain of interest”. For example, in the sentence *The weather today is very...*, one might say that is probable for the next word in this sequence to be *good* or *bad* but not *car* or *fridge*. Given the computational expense of this task, it was common to approximate the most probable next word taking into consideration only two or three words before the word of interest. The authors of [15] explicitly reference this: “The central goal of the most commonly used language models, trigram models, is to determine the probability of a word given the previous two words: $p(w_i|w_{i-2}w_{i-1})$. The simplest way to approximate this probability is to compute the number of times the word sequence $w_{i-2}w_{i-1}w_i$ occurs in some corpus of training data divided by the number of times the word sequence $w_{i-2}w_{i-1}$ occurs. This value is called the maximum likelihood (ML) estimate”.

As the era progressed, statistical techniques permeated other areas of NLP. Part-of-speech tagging saw improvements with approaches like the Maximum Entropy models introduced by Ratnaparkhi in [51]. Such models, relying on the principle of making minimal assumptions, showcased remarkable precision in linguistic annotation tasks.

In parallel, the quest to decode the semantic essence of language, a challenge deemed intricate due to its subjective and context-dependent nature, saw innovative endeavours during this period. Traditional models, grounded in syntactic analyses, often fell short in capturing the nuanced semantic relationships between words. Recognizing this limitation, in [19] Pereira, Tishby, and Lee (1993) explored distributional clustering, a novel technique premised on the hypothesis that words sharing similar contexts bear semantic resemblance. Their pioneering research underscored the significance of analyzing word co-occurrences and distributions within corpora to infer semantic affinities. This insight not only facilitated a deeper understanding of word semantics but also sowed the seeds for future methodologies, notably word embeddings, that would leverage distributional properties to generate dense, continuous vector representations of words, capturing semantic and syntactic nuances with enhanced finesse.

2.4 The Rise of Machine Learning

From 2000 to 2010, during the transition into the 21st century, a significant paradigm shift for [NLP](#) occurred. In substitution of hard coded rules or more simplistic statistical approaches, the algorithms starting to learn from the data itself. One of the noticeable shifts was in sentiment analysis. Instead of relying on pre-established lists of words with their sentiment values, models like [Support Vector Machines \(SVMs\)](#) became prevalent, offering a more dynamic approach by learning sentiment from diverse textual examples. A pivotal example was the work of [30]. In this paper, Joachims presented a method for text categorization using [SVMs](#) and emphasized how high-dimensional feature spaces, characteristic of text data, can be effectively handled by [SVMs](#). He demonstrated that [SVMs](#), even with the standard settings, can outperform previously established methods on a variety of datasets.

In 2001, probabilistic graphical models, such as Conditional Random Fields introduced by Lafferty et al. in [38], became popular for structured prediction tasks in [NLP](#), like part-of-speech tagging and [Named Entity Recognition \(NER\)](#). Unlike earlier models, Conditional Random Fields could take the entire sequence into account, leading to more accurate predictions.

Another major development was the shift towards unsupervised and semi-supervised learning techniques, given the high cost of obtaining labeled data. Algorithms that could leverage unlabeled data, like the co-training approach and bootstrapping methods, became essential.

Topic modeling also gained prominence, with Latent Dirichlet Allocation by Blei et al., presented in [9], which emerged as a go-to method for extracting topics from large text corpora. The ability of the algorithm to uncover hidden thematic structures in texts made it invaluable for content summarizing and organization and represented a significant improvement compared to the previously available techniques.

2.5 The Deep Learning Revolution

From the 2010 onwards, the deep learning revolution, which began to shape various fields of artificial intelligence, also had a profound impact on [NLP](#). One of the primary contributors to this transition was the adoption and advancement of [Recurrent Neural Networks \(RNNs\)](#). [RNNs](#), given their ability to process sequences, were aptly suited for tasks that involved a temporal dimension, like time series analysis, but more pertinently for [NLP](#) tasks such as machine translation, speech recognition, and text generation. A pivotal work in the domain of machine translation using [RNNs](#) is by Sutskever et al. (2014), titled “Sequence to Sequence Learning with Neural Networks”. In [58], the authors of this paper presented a model that uses [RNNs](#) to map input sequences to output sequences. The framework was shown to perform well on an English-to-French translation task, signaling the potential for deep learning methods in replacing traditional statistical machine translation systems.

However, while [RNNs](#) held promise, they suffered from problems like vanishing and exploding gradients, especially when dealing with long sequences. This shortcoming led to the development of [Long Short-Term Memory \(LSTM\)](#) networks by Hochreiter and Schmidhuber. [LSTM](#) models were designed with memory cells that regulated the flow of information, allowing them to retain or forget data over long sequences, thereby alleviating some of the challenges faced by standard [RNNs](#). The potential of [LSTM](#) networks in speech recognition was underscored by Graves et al. in [26] with their paper “Speech Recognition with Deep Recurrent Neural Networks”. Here, the authors demonstrated that deep bidirectional [LSTM](#) models, when combined with traditional [RNNs](#), significantly reduced word error rates in speech recognition tasks. Another seminal contribution in the domain of sequence processing using the [LSTM](#) architecture was the paper “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation” by Cho et al., published in (2014). In [16], the authors laid the foundation for sequence-to-sequence architectures, showing that [LSTM](#) could encode a source sentence into a fixed-length vector and decode it into a target sentence, offering a novel approach to machine translation.

The year of 2013 marked the advent of a paradigm-shifting approach to word representation: word embeddings. Central to this revolution was the development of Word2Vec by Mikolov et al, presented in [45], with a paper titled “Efficient estimation of word representations in vector space”. Word2Vec was innovative in its ability to map words into a continuous vector space, where the positioning of words in this space intuitively encapsulated semantic and syntactic relationships. For instance, vector operations could capture relationships like *King - Man + Woman $\approx$ Queen*. This was a radical departure from previous techniques that largely represented words as isolated entities without considering their contextual relationships. Mikolov methods, through efficient algorithms such as skip-gram and continuous bag of words, not only improved performance in various tasks but also reduced computational overhead.

Building on the foundational advancements in word embeddings and deep learning architectures, the latter part of the decade observed the rise of holistic models tailored to handle diverse language tasks without the necessity of extensive task-specific architectures. A monumental stride in this direction was the introduction of the [Transformer](#) architecture in [63] by Vaswani et al. in 2017 where they presented the landmark study “Attention Is All You Need”. The [Transformer](#) model discarded recurrence entirely and instead utilized a [Self-Attention](#) mechanism to weigh input tokens differently, enabling it to capture long-range dependencies and intricate relationships in textual data. Its scalability and flexibility led to the birth of models like [Bidirectional Encoder Representations from Transformers \(BERT\)](#), which fine-tuned pre-trained representations for various tasks, and [Generative Pre-trained Transformer \(GPT\)](#), which showcased the power of unsupervised learning and transfer learning in [NLP](#). Both models, inspired by the architecture of the [Transformer](#), set new standards in a range of [NLP](#) benchmarks, from question-answering to sentiment analysis.

Given the pivotal importance of deep learning techniques in the context of this work,

we will now go over them in more detail, in particular focusing on the [Transformers](#) architecture. We will also address the foundational pillars on which this model is based upon, notably the idea of [Word Embedding](#) and [Positional Embedding](#)

2.5.1 The Transformer Model

To better understand how a [Transformer](#) works, we replicate from [63] an image which represents how the model is constructed:

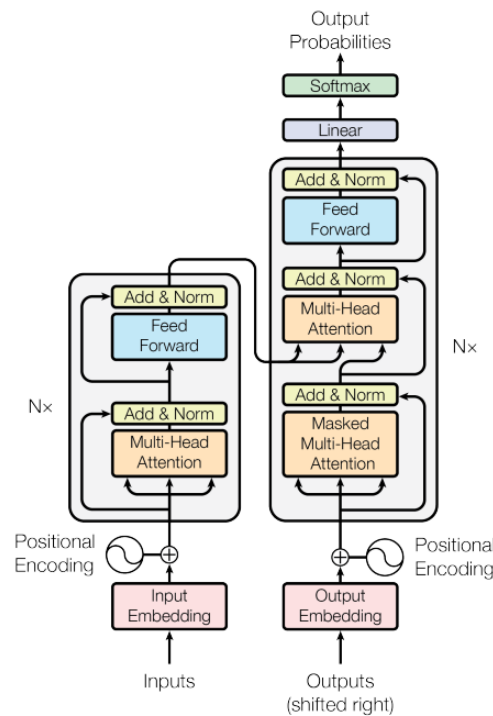


Figure 2.1: The Transformer architecture

At its core, the [Transformer](#) comprises two main structures: an encoder (shown in the left part of Figure 2.1) and a decoder (shown in the right part of Figure 2.1). Both of these structures are formed by multiple layers of the same design, with the original [Transformer](#) model utilizing six layers in each segment, making a total of twelve layers.

To understand how the functioning of the [Transformer](#), it's essential to break down its primary components and their roles. As shown in Figure 2.1, the model receives input embeddings, which correspond to [Word Embeddings](#). Therefore, before looking at the actual functioning of the model itself, we will describe the need to use a vector representation of words.

A note of clarification is due at this stage. In order to give make our work more easily understood, the examples given in the next sections will often be based on words. However, it is important to note that the inputs received by the [Transformer](#) model do not have to be words, as the primary inputs that are fed into the embedding layers are the tokenized sequences. These sequences can be words, [Subwords](#), or even characters, depending on

the granularity of the tokenizer used. Thus, although we will utilize words as the basis of our examples, it is important to keep in mind that the [Transformer](#) can receive other units of language.

2.5.1.1 Word Embeddings

As computers are much more capable to process numbers than words, the idea to convert words into numbers was always present in the field of [NLP](#), which led to the creation of the concept of vector semantics. Vector semantics refers to the idea of representing words in a multi-dimensional space. Each word is represented as a point or vector in this space. The positioning of the word vectors is determined by the contextual similarities of words and their differences. In simple terms, words with similar meanings or contexts will be close in this space, whereas words with different meanings or contexts will be far apart.

Representing words as vectors was helpful in terms of computational efficiency, but it created another problem: how can one convert a word into a numerical representation, taking into account the need to account for contextual similarities? The initial approaches relied on count-based methods [20, 62, 27]. This involves analyzing how often words co-occur with other words in large texts. The representation of a word is then based on these co-occurrence counts with other words. For example, in a given corpus, one might count how often the word “apple” appears near the word “fruit” versus how often it appears near the word “car”. The same count-based principle was applied to documents. A document was represented by a vector that recorded the counts of words in it. So, two documents talking about similar topics would have similar vector representations because they would likely use similar words.

This approach had several problems. First, creating vector representation of words by counting co-occurrences with other words is extremely inefficient. Second, words that have the same meaning can be used interchangeably. For example, if one decides to find similar document by counting the words that appear in those documents, the expression “President of the United States” can be used in one document and the expression “Joe Biden” can be used in another document. Despite the fact that these documents are most likely related, their vector representation can be far apart. Another cause for the problems of count-based approaches is the fact that they generate very long, sparse vectors. This is so because most words will not be neighbours of most other words.

To circumvent the above-mentioned limitations, other approaches emerged, most notably Word2Vec, presented in [45]. The objective of Word2Vec was to map word to dense vector spaces by making use of shallow [Neural Networks](#). The first step was to represent words as one-hot vectors. For example, in the sentence *The red Ferrari is fast*, the word *Ferrari* would be represented as [0,0,1,0,0]. To train the [Neural Network](#), two approaches could be used: [Continuous Bag of Words \(CBOW\)](#) and the skip-gram. The [CBOW](#) consisted of considering the surrounding words in a sentence and, given the vectors for the words, the objective was to predict the center words. Considering a simple

example, in which we only take into account the words at position $t - 1$ and $t + 1$, we could use the one-hot vector of *red* and *fast* as the inputs of the [Neural Network](#). These inputs would then be used in a dense projection layer. For example, the initial one-hot vectors of dimension 5 could be mapped to a projection layer of size 3 with weights randomly initialized. From this projection layer to the output layer, the dense vector would then be transformed again into the original space of dimension 5, by using another set of randomly initialized weights. We would then see how the this output compared to the one-hot vector of the word *Ferrari* and use back propagation to optimize the weights. The skip-gram model works in a similar manner, but the other way around. For instance, in our example from above, the input to the [Neural Network](#) would be the word *Ferrari* and the objective would be to predict the surrounding words, in this case *red* and *fast*.

In more rigorous terms, as described in [33], the goal of Word2Vec is to train a classifier such that, given a tuple (w, c) of a target word w paired with a candidate context word c (for example (Ferrari, red), or perhaps (Ferrari, slow)) it will return the probability that c is a real context word (true for red, false for slow):

$$P(+|w, c) \tag{2.1}$$

The probability that word c is not a real context word for w is just 1 minus Eq. 2.1:

$$P(-|w, c) = 1 - P(+|w, c) \tag{2.2}$$

To compute the probability P we simply compute the dot product between the embedding vectors of c and w :

$$\text{Similarity}(w, c) \approx \mathbf{c} \cdot \mathbf{w} \tag{2.3}$$

The intuition here is that a word is likely to appear next to another word if their embedding vectors are similar. The embedding vectors that represent the words c and w were obtained from the one-hot training process mentioned above.

Noting that we need a probability (and not a similarity measure that ranges from $-\infty$ to ∞ , which is what Eq. 2.3 returns), we use the sigmoid function to obtain a number between 0 and 1:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2.4}$$

thus modeling the probability that word c is a real context word for w as:

$$P(+|w, c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{c} \cdot \mathbf{w})} \tag{2.5}$$

We also need that the total probability of the two possible events (c is a real context word and c is not a context word) sums to 1. Thus, we estimate the probability that the word c is not a real context word for w as:

$$P(-|w, c) = 1 - P(+|w, c) = \sigma(-\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(\mathbf{c} \cdot \mathbf{w})} \quad (2.6)$$

As the last step, we take into account that Eq. 2.5 gives us the probability for one word, but the context window can consider multiple words, before and after the target word. As skip-gram makes the simplistic assumption that words are independent, to obtain a final value for the probability of all words of context in a given context window, we can multiply their probabilities:

$$P(+|w, c_{1:L}) = \prod_{i=1}^L \sigma(\mathbf{c} \cdot \mathbf{w}) \quad (2.7)$$

and taking the log probabilities to avoid under-flows we get:

$$\log(P(+|w, c_{1:L})) = \sum_{i=1}^L \log(\sigma(\mathbf{c} \cdot \mathbf{w})) \quad (2.8)$$

This explanation, described in [33] describes how word embeddings were used in Word2Vec to compute the probabilities of context words, in the case of skip-gram.

With Word2Vec, two major step forwards were achieved: first, word embedding stopped being sparse representation of words and were transformed into dense vector spaces; second, the meaning of words was learned by the network, meaning that words with similar meanings actually had similar representations. This is why it was possible to achieve the result already mentioned in Section 2.5: *King - Man + Woman $\approx$ Queen*.

Although some progresses were made, the embeddings from Word2Vec are known as static embeddings, which prevents them from capture polysemy. In simple terms, a single embedding might be insufficient to represent words with multiple meanings depending on context. For instance, let us consider the following two sentences: 1 - I sat on the bank of the river and enjoyed the view; and 2 - I went to the bank to deposit my paycheck. It is clear that the word “bank” has a completely distinct meaning in each of the sentences. However, because Word2Vec only produces one embedding per word, the embedding of the word “bank” would be the same in both cases.

It is thus clear that, despite the fact that Word2Vec made substantial progresses when comparing it with the initial count-based approaches, word embeddings still needed to suffer a substantial leap forward if they were to accurately represent the meaning of words. To overcome these limitations, the [Transformer](#) model relies on a novel and decisive new mechanism: [Self-Attention](#). We will come back to [Self-Attention](#) in a few paragraphs.

For now, it is important to make clear that in [63], the original paper where the [Transformer](#) architecture was presented, the vectors of the embeddings representing the words were of dimension $d_{model} = 512$.

Returning to Fig. 2.1, we can see that after word embeddings, the inputs also receive positional information through positional encoding. We will now briefly address this step.

2.5.1.2 Positional Encoding

As stated before, in the [Transformer](#) architecture, a core innovation is the use of the [Self-Attention](#) mechanism which allows each token (for instance, words) in a sequence to focus on all other tokens simultaneously. However, this mechanism, by design, lacks an inherent sense of order or position of tokens within a sequence. To address this, positional encodings were introduced to provide the model with explicit information about the position of each token. As explained in [63], these positional encodings had the same dimension of the inputs embeddings, d_{model} , such that the input embeddings and the positional embedding could be summed.

It is possible to choose from a wide variety of positional encodings, learned and fixed. The original [Transformer](#) model proposed a method to generate these encodings using a combination of sine and cosine functions. Specifically, for a given position pos and dimension i :

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (2.9)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (2.10)$$

where d_{model} represents the dimension of the embeddings. One significant advantage of these sinusoidal functions is that they produce values between -1 and 1, ensuring compatibility in scale with typical word embeddings. Moreover, this choice of functions allows the model to discern relative positions in a predictable manner, making it conducive to learn patterns based on word order.

Let us consider two sentences to make things more clear: 1 - She gave him a book; 2 - A book was given to him by her. Although these sentences have the same meaning, the position of the words is different. The positional encodings capture these varying distances and relationships. When sinusoidal functions generate the positional encodings, the model can compute relative positions by leveraging the unique properties of sines and cosines. As pointed out in [63], the sinusoidal functions were chosen because the authors hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k , P_{pos+k} can be represented as a linear function of P_{pos} . This is as such because, from trigonometry, we have that:

$$\sin(A + B) = \sin(A)\cos(B) + \cos(A)\sin(B) \quad (2.11)$$

$$\cos(A + B) = \cos(A)\cos(B) - \sin(A)\sin(B) \quad (2.12)$$

If we let

$$A = \frac{pos}{10000^{\frac{2i}{d_{model}}}} \quad (2.13)$$

and

$$B = \frac{k}{10000^{\frac{2i}{d_{\text{model}}}}} \quad (2.14)$$

we can express $\sin(A + B)$ and $\cos(A + B)$ as a combination (linear transformation) of $\sin(A)$ and $\cos(B)$. The coefficients of this combination come from the values of $\sin(B)$ and $\cos(B)$, which only depend on the fixed offset k . In essence, these trigonometric coefficients act as the linear transformation that can represent PE_{pos+k} in terms of PE_{pos} . The self-attention mechanism of the [Transformer](#) leverages this property to generalize relative positions across different parts of an input sequence. In practical terms, this means that if the attention mechanism in the [Transformer](#) learns that a particular word often has a specific relationship with words three positions away (like a subject-verb relationship or any other linguistic pattern), it doesn't need to relearn this relationship for every possible position in a sentence. Instead, it can generalize the relationship across positions, thanks to the property of the chosen sinusoidal positional encodings.

As already mentioned, the positional encodings would be summed to the inputs embeddings of each token. From here, looking into Fig. 2.1 we move to the multi-head attention mechanism. This is the layer we will be explaining the following paragraphs.

2.5.1.3 The Attention Mechanism

The [Self-Attention](#) is the core evolution of the [Transformer](#) model. In [63] this was graphically represented as follows:

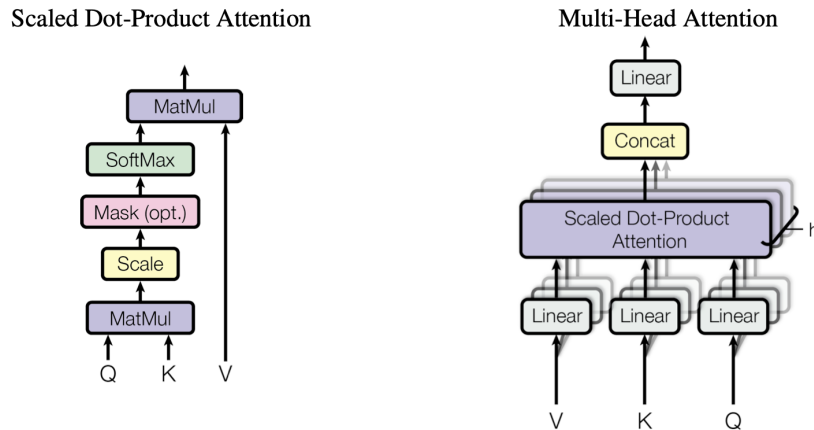


Figure 2.2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

At a high level the multi-head [Self-Attention](#), allows the model to focus on different parts of the input simultaneously, capturing various types of relationships and patterns in the data. The fundamental idea behind the attention mechanism is to compute a weighted sum of values (V) based on their relevance to a given query (Q). This relevance

is determined by calculating the similarity between the query and corresponding keys (K), and then applying a softmax to derive the weights. We will now go in more detail into each of these stages.

First, let us define the role of these three different entities, queries, keys and values. The queries represent the elements we are currently focusing on. In the context of language, when processing a word or token, the query represents that word's attempt to seek or query relevant context from other parts of the input. The keys represent all the elements in the context, that is, every word or token in the sequence provides a key that indicates its potential relevance to any query. Finally, the values are the actual content or information we want to retrieve once we've identified which parts of the context are relevant to our query.

Each of these entities then goes to a linear layer. A linear layer, often called a fully connected layer or dense layer, is one of the fundamental building blocks in neural networks. At its core, a linear layer performs a transformation on its input. Mathematically, for an input vector x , the transformation in a linear layer can be expressed as:

$$y = Wx + b \quad (2.15)$$

where W is the weight matrix which contains the learnable weights and b is the bias vector. The purpose of the weight matrix and bias vector is to allow the network to learn and represent various transformations of the input data. By adjusting the values in W and b during training, the network can learn to produce outputs that are useful for a given task. Linear layers are also essential for the purpose of controlling the dimensionality of the queries, keys and values. As in any neural network, the weights and biases are randomly initialized and optimized during training. When the queries, keys and values go through the respective linear layers, we would get three matrices: W^Q , W^K , W^V . It is also important to note that, as it is visible from Figure 2.2, the values for queries, keys and values go through different linear layers in parallel, more than one time, given that we have several attention heads (hence the name *multi-head* attention). In the original paper the number of attention heads was $h = 8$. For each of these heads, the shape of the projection matrices where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$ and $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, where i represents each of the attention heads and $d_k = d_v = d_{model}/h = 64$.

Having these projection matrices, we then move to the scaled dot-product attention. From Figure 2.2, it is possible to see that the first step at this stage is to multiply the matrices Q and K . The dot product between queries and keys returns the scores which represent how much attention each key-value pair should receive based on that specific query. These scores are then scaled down by dividing by the square root of the depth of the queries/keys (d_k). This scaling helps prevent large dot product values from causing the Softmax function to have extremely small gradients, which can slow down learning or cause it to become stagnant. The scaled scores are passed through a Softmax function. The purpose of this step is to convert the scores into attention weights. After applying the

Softmax, each score is in the range (0,1), and the sum of scores for a given query equals 1, meaning we have a vector of probabilities. For a given vector $\mathbf{z} = [z_1, z_2, \dots, z_n]$ the Softmax function is defined as:

$$(z_i) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} \quad (2.16)$$

The Softmax function serves to highlight the most relevant keys (largest scores) while diminishing the less relevant ones, allowing the model to focus more on certain parts of the input (those with higher scores) while not completely ignoring the others.

Finally, these Softmax attention weights are used to compute a weighted sum of the values matrix (the last step shown on the left part of Figure 2.2). All of these steps can be mathematically described as:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_K}}\right) \mathbf{V} \quad (2.17)$$

The resulting vectors for each head are then concatenated and linearly transformed to produce the final output of the multi-head attention mechanism, as shown in the final two stages of the multi-head attention represented in the right part of the Figure 2.2.

After the multi-head attention layer, we see from Figure 2.1 that the outputs will go through an add & norm, before passing to a [Feed-Forward Layer](#). We will now look into each of these layers.

2.5.1.4 Add and Norm and Feed Forward Layers

In the Transformer architecture, immediately following the multihead attention mechanism, the Add & Norm step is introduced. This step comprises two primary elements: the residual connection and layer normalization.

The residual connection, often referred to as a shortcut or skip connection, directly bridges the input and output of the multi-head attention layer. More specifically, the input to the multi-head attention module is straightforwardly added to its output. To represent this mathematically, let x denote the input and $Attention(x)$ corresponding output. The resulting value after applying the residual connection is given by $x + Attention(x)$. Such residual connections have emerged as a vital tool for training deeper models, mainly because they effectively counteract the notorious vanishing gradient problem. By fostering an environment where gradients can traverse the network with fewer obstructions, they ensure a more fluid and efficient training process.

Subsequent to the residual connection, layer normalization is executed. Unlike batch normalization, which standardizes data across the batch, layer normalization operates by normalizing the data across its features. The primary intent behind this normalization is to harmonize the feature values, thereby achieving a more consistent and stable set of activations.

Following the multi-head attention mechanism and its subsequent normalization, we encounter the position-wise feed-forward network, commonly denoted as the **Feed-Forward Layer**. This layer plays a pivotal role in the functionality of the Transformer model. It operates in a position-wise manner, meaning it addresses each position — be it a word or token representation — independently and uniformly. Such a design ensures that the identical neural network mechanism is applied autonomously to each token's output from the preceding layer, without intermingling information from disparate positions in the sequence. A crucial role of the **Feed-Forward Layer** is the introduction of non-linearity. By comprising two linear transformations (or fully connected layers) separated by a non-linear activation function, typically the **Rectified Linear Unit (ReLU)**, the **Feed-Forward Layer** amplifies the capability of the **Transformer** to delineate relationships and patterns. If these non-linearities were not present in the model, the overall utility of stacking multiple Transformer layers would be compromised, as a cascade of linear operations could essentially be compressed into a singular layer. In the original paper the **Feed-Forward Layer**, consists of two linear transformations with a ReLU activation in between:

$$FNN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2.18)$$

As explained in [63], while the linear transformations are the same across different positions, they use different parameters from layer to layer. The dimensionality of input and output is $d_{model} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

This concludes our overview of the inner components of the **Transformer**. We now need to explore the bigger picture and understand what is the role of the encoder and the decoder.

2.5.1.5 Encoder

The encoder, shown in the left part of Figure 2.1 is composed of a stack of identical layers. In each layer we have the multi-head self-attention, the **Feed-Forward Layer** and the add & norm mechanisms already explained. The encoder in the **Transformer** architecture is responsible for converting a variable-length input sequence into a fixed-length contextual representation. Given an input sequence $x_1, \dots, x_n$, the encoder will produce a sequence $y_1, \dots, y_n$. We can interpret y_i as a representation of the meaning of the token x_i in the context of the sentence $x_1, \dots, x_n$. These embedding can be used as “representations of word meanings in context for any task that might require a model of word meaning”[33]. Multiple encoder layers (6 in the original paper) are stacked to form the encoder. The output from one layer is used as input to the next.

2.5.2 Decoder

The decoder in the **Transformer** architecture is responsible for converting the fixed-length contextual representation provided by the encoder into a variable-length output sequence,

often in a different format or language. The decoder is also composed of a stack of 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. In the original paper of the [Transformer](#), the self-attention sub-layer in the decoder stack was also modified to prevent positions from attending to subsequent positions, such that the objective is to predict the subsequent token. Masking consists of setting to $-\infty$ all values in the input of the softmax function corresponding to illegal connections. This masking, known as masked self-attention, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i [63].

2.5.3 Encoder-Decoder Functioning

To better understand how the encoder and the decoder come together, we can look at the following image:

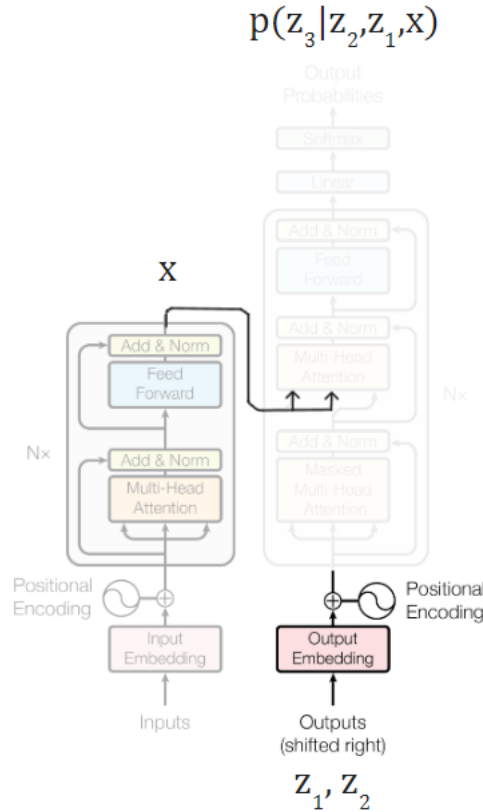


Figure 2.3: Orchestrating the encoder-decoder model.

Figure 2.3, adapted from [7] shows how the encoder and the decoder work together. The [Transformer](#) aims at modeling the probability distribution over all possible output given an input sequence, that is $p(z|x)$, such that we can choose from that distribution the most probable output. For example, given the input sequence *The red Ferrari...* the

model should predict all possible output sequences. This is unfeasible. To overcome this, the authors of [63] decompose the output sequence z into tokens and the objective of the model became to predict the most probable token (which can be a word or a character, for instance). Doing this means that we now have manageable probabilities as the model can generate one token at a time and look at all previous generated tokens. For instance, if we want to generate $p(z_3|z_2, z_1, x)$, x comes from the encoder and z_1 and z_2 is what the model has predicted so far, which goes into the decoder [7], as shown in Figure 2.3.

RELATED WORK

Existing pre-trained language models designed for general purposes are not well suited to handle legal elements [41]. Many reasons can explain such fact, the most obvious one being the way in which legal documents are written. Whether we are considering the law itself or court decisions, the way the text is written is very different from normal text, with sentences and paragraphs being characterized by very long and intricate sequences which can be very hard for humans not trained in the legal domain to understand. Other limitations and challenges are outlined in Chapter 1.3. In this chapter of our work we will look at what has been done by researchers and practitioners to try to circumvent this problem.

3.1 Large Language Models

The evolution of artificial intelligence has been significantly influenced by the development of LLM. These models are pivotal in the field of NLP and rely heavily on advanced deep learning techniques and increasing computational capabilities. As described before, the initial efforts in this field were directed towards creating machines with linguistic abilities, which were predominantly rule-based systems. However, as computational methods advanced, there was a discernible shift towards statistical models, and subsequently, to deep learning models.

Among the various neural network architectures available, transformer-based models have emerged as particularly effective for NLP tasks, due to the already explained attention mechanism, which allows them to assign varying degrees of relevance to different parts of the input data. This mechanism is especially suited to processing sequential data, which is inherent in language. The training of these models requires them to process extensive datasets, often encompassing vast portions of available digital text. This exposure enables the models to discern patterns, nuances, and structures that are characteristic of human language. Once trained, these models have the capability to generate text, respond to queries, and execute a range of tasks that demonstrate a high level of linguistic understanding.

However, the rise of LLMs has brought with it a set of challenges. First, one significant concern is the potential for these models to inherit and perpetuate biases present in the training data. Given that many of these models are trained on data from the internet, they can inadvertently adopt and magnify existing societal biases, leading to ethical concerns regarding their deployment. Second, the computational resources required for training LLMs are substantial, leading to concerns about environmental sustainability due to the energy consumption associated with such processes [12, 6, 10, 57]. Third, hallucinations in LLMs are particularly challenging and harmful because they can lead to the generation of false or misleading information that appears convincingly real. This issue is not only a concern for the integrity and reliability of information but also poses significant risks in scenarios where accurate data is critical, such as medical advice, financial forecasting, and, naturally, legal analysis. These hallucinations can undermine the trustworthiness of LLMs and, by extension, the institutions that rely on them, potentially causing real-world consequences based on inaccuracies. Moreover, distinguishing between accurate information and hallucinations can be difficult, requiring additional layers of verification and oversight, further complicating the deployment and maintenance of these systems. Throughout this work, we will remain agnostic of the set of concerns that surround LLMs. In the next paragraphs we will do an overview over some of the most notable language models.

3.1.1 Bidirectional Encoder Representation from Transformers

BERT, introduced in [21] has been a significant advancement in the field of natural language processing, particularly within the context of deep learning methodologies. Built upon the Transformer architecture, BERT diverges from traditional language models by employing a bidirectional approach to understanding context, which is contrasted against the unidirectional or even shallow bidirectional methods of previous models. This bidirectionality enables BERT to consider both the preceding and succeeding context in a given text, leading to a more comprehensive understanding of semantic relationships within a language structure.

One of the singularities of BERT lies in the training method of the model. Rather than predicting the next word in a sequence, BERT is trained to predict a word that has been intentionally masked out from a given input. This masked language model approach forces the neural network to learn contextual representations of words, as it has to use surrounding words to predict the masked one.

The pre-training phase of BERT involves processing vast amounts of text, where the model learns to make these masked word predictions. Once this phase is complete, BERT can be fine-tuned for specific downstream tasks using relatively smaller, task-specific datasets. Such tasks can range from sentiment analysis and named entity recognition to more complex undertakings like question-answering systems. This two-step process of extensive pre-training followed by task-specific fine-tuning is what allows BERT to achieve

state-of-the-art performance across a number of [NLP](#) benchmarks.

However, the sophistication of [BERT](#) also brings forth computational challenges. The model's depth, combined with bidirectionality, necessitates significant computational resources, both in terms of memory and processing power. This has led to the development of smaller, more optimized versions of [BERT](#) for deployment in resource-constrained environments, which explains why two models were presented in [21]: $BERT_{Base}$ with 12 layers, a hidden size of 768 and 12 self-attention heads; and $BERT_{Large}$ with 24 layers, a hidden size of 1024 and 116 self-attention heads. The base model has 110 million parameters and the large one has 340 million.

[BERT](#)'s introduction and subsequent dominance in various [NLP](#) tasks showed the potential of bidirectional and transformer-based architectures in capturing the intricacies of human language. The model's design and training methodology have set new standards in the field, prompting further research into more efficient and even more potent variants of transformer-based language models.

3.1.2 Sentence-BERT

The quest for more efficient models led to the development of [Sentence-BERT \(SBERT\)](#), a modification of the pre-trained [BERT](#) network that enables it to derive meaningful sentence embeddings more efficiently. Introduced by Reimers and Gurevych in 2019, [SBERT](#) was specifically designed to tackle the challenge of semantic similarity assessment and clustering at the sentence level, tasks for which traditional [BERT](#) models were less suited due to their computational inefficiency in generating sentence embeddings [52].

[SBERT](#) modifies the original [BERT](#) architecture by incorporating a siamese and triplet network structure to the pre-training process, enabling the generation of sentence embeddings that can be compared using cosine similarity. This approach significantly reduces the computational burden associated with using [BERT](#) for sentence-level tasks. While [BERT](#) requires feeding pairs of sentences to the model to compute their similarity — a process that grows quadratically with the number of sentences — [SBERT](#) pre-computes fixed-length vector representations for each sentence. These embeddings can then be compared efficiently, making [SBERT](#) vastly more suitable for real-time applications and large-scale datasets [52].

The implications of [SBERT](#) are substantial for a wide range of [NLP](#) applications, including but not limited to semantic search, sentiment analysis, and information retrieval. By facilitating the generation of semantically meaningful sentence embeddings, [SBERT](#) enables more nuanced and context-aware machine understanding of language, improving the performance of downstream tasks that rely on semantic similarity.

3.1.3 Generative Pre-Trained Transformer Decoder

The [GPT](#) model has emerged as a prominent fixture in the landscape of natural language processing, signifying a notable evolution within deep learning techniques applied to

language generation tasks. Also based in the [Transformer](#) architecture, [GPT](#) offers a unique approach to language modeling compared to its contemporaries.

Unlike [BERT](#) which relies on bidirectional encodings and masked language training methodologies, [GPT](#) follows a unidirectional, autoregressive approach. In this training paradigm, [GPT](#) predicts the next word in a sequence based solely on the preceding words. This design choice simplifies the model’s architecture and training process, yet it surprisingly produces rich semantic representations, allowing the model to generate coherent and contextually relevant sequences of text.

The training strategy of [GPT](#) is typically divided into two main phases. The initial phase involves unsupervised pre-training over vast corpora of text. During this phase, the model learns to predict words in sequences, enabling it to capture a broad range of language structures, nuances, and patterns. As described in [49], given an unsupervised corpus of tokens $U = u_1, \dots, u_n$, the authors use a standard language modeling objective to maximize the following likelihood:

$$L_1(U) = \sum_{i=k}^n \log P(u_i | u_{i-k}, \dots, u_{i-1}; \theta) \quad (3.1)$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters θ . These parameters are trained using stochastic gradient descent. In [49], the authors used a multi-layer Transformer decoder for the language model, which is a variant of the [Transformer](#). This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens.

Following this, the fine-tuning phase adjusts the pre-trained model on specific tasks using smaller, task-relevant datasets. Despite its generative nature, this fine-tuning allows [GPT](#) to be applied effectively to a wide array of [NLP](#) tasks, from text classification and sentiment analysis to more complex challenges like machine translation and question-answering.

The accuracy of [GPT](#) in diverse [NLP](#) tasks underscores its versatility, a feature further amplified in its subsequent iterations, such as GPT-2, GPT-3 and GPT-4, which boast increased model sizes and enhanced performance.

The emergence of [GPT](#) and its subsequent versions have significantly influenced the trajectory of [NLP](#) research and applications. Its design philosophy, rooted in unidirectional, autoregressive training, and its scalability have set it apart, driving continued interest and exploration in the potential of transformer-based models for understanding and generating human language.

Following this trajectory, and as briefly discussed in Chapter 1.2, the introduction of the [CoT](#) task has further exemplified the evolving sophistication of [NLP](#) models, particularly in complex problem-solving and reasoning domains. The [CoT](#) approach, as applied in models like [GPT-3](#), involves prompting the model to articulate a step-by-step reasoning process for solving problems, rather than attempting to reach a solution in a single step.

This methodology enables the model to tackle more complex questions and problems that require multiple inferential steps, mimicking the way humans often approach problem-solving [66].

The CoT task represents a significant leap forward in the quest for AI systems capable of human-like reasoning. By encouraging models to "think aloud," the CoT approach has shown to dramatically improve performance on a variety of tasks, from arithmetic calculations to commonsense reasoning and beyond. This advancement not only highlights the versatility and adaptability of transformer-based models like GPT but also opens new avenues for research and applications in NLP. It suggests that models can not only understand and generate human language but can also engage in complex reasoning processes, providing detailed explanations of their thought processes along the way [66]. This could be of critical importance for law field. For instance, in Portugal, decisions tend to be very long because explanations of how judges reached a given conclusion must be extremely detailed so that they can be syndicated. By forcing a model to explain all the steps taken to reached a given output, a human-like approach could be achieved.

3.1.4 The T-5 Model (Encoder-Decoder)

The T-5 is year another model that arises from the Transformer lineage. Presented in [50] and dubbed as the "Text-to-Text Transfer Transformer", hence the name, the T-5 model is build upon a singular approach to NLP tasks: text-to-text. Essentially, this methodology treats every NLP task as a transformation of one text type into another. This technique provides a unified, consistent framework for a number of NLP tasks, from machine translation to text summarization, and from question-answering to sentiment analysis. Instead of needing separate architectures for each task, the text-to-text method employed on the T-5 offers versatility across multiple domains.

Contrary to BERT (encoder only) and to GPT (decoder only), the architecture of the T-5 closely follows that of the original Transformer presented in [63], being composed both by an encoder and a decoder. The encoder component ingests the input sequence, synthesizing its informational elements into a concise context vector. This vector serves as a distilled representation of the input's essence. Following this, the decoder is responsible for interpreting this context vector, generating the requisite output, be it a translated sentence, a text summary, or any other desired textual format.

A critical aspect of the T-5's performance lies in its strategic training regimen. As explained in [50], initially, the model undergoes rigorous pre-training on vast text corpora. This phase is foundational, providing the model with a broad linguistic understanding, capturing nuances and patterns. Once this foundational knowledge is established, the model transitions to a fine-tuning phase, where it is tailored to specific NLP tasks using more specialized datasets. This hierarchical training approach ensures that the T-5, while possessing a comprehensive understanding of language structures and semantics from its pre-training, can achieve optimal, specialized performance in specific domains through

its subsequent fine-tuning.

3.1.5 The Adaptability of the Transformers Model

The Transformer architecture’s flexibility is evident in how [BERT](#), [GPT](#), and T-5 harness its components differently. While they all rest on the Transformer’s model, [BERT](#) relies only in the encoder mechanism without employing the decoder. In contrast [GPT](#), utilizes only the decoder, bypassing the encoder phase. T-5 stands unique, integrating both the encoder and decoder, offering a more comprehensive sequence-to-sequence formulation. This architectural adaptability of the Transformer allows for diverse model formulations, tailoring to specific [NLP](#) objectives.

In the following paragraphs we will devote our attention to how researchers and practitioners have specifically tackled the problem of [NLP](#) in the legal domain.

3.2 SAILER: Structure-aware Pre-trained Language Model for Legal Case Retrieval

In [\[41\]](#) a new language model for legal case retrieval¹ was presented. In this paper, the authors advocate that despite the success that pre-training has achieved in ad-hoc retrieval tasks, effective pre-training strategies for legal case retrieval remain to be explored, primarily identifying two main reasons that led to this outcome. The first key challenge is related with the fact that most existing language models have difficulty understanding the long-distance dependencies between different structures that characterizes legal documents. A representation of what the authors considered to be the natural legal structure can be seen in [Figure 3.1](#).

The second key challenge identified in [\[41\]](#) specifically concerning **information retrieval in the legal domain respects to the fact that the concept of relevance in the legal domain is quite different from that in general search**. The following example is given by the authors:

1. Situation A: Person X, 24 years old, men, height 180 cm. In 2018, he entered the shopping mall five times where he **stole** one cell phone and two tablet computers, worth a total of 10,000,000. At five o’clock, he returned home and gave the above items to Person Y.
2. Situation B: Person X, 24 years old, men, height 180 cm. In 2018, he entered the shopping mall five times where he **purchased** one cell phone and two tablet computers, worth a total of 10,000,000. At five o’clock, he returned home and gave the above items to Person Y.

¹Formally, in information retrieval, given a query q , and a set of candidate cases D , the task is to retrieve the top- k relevant cases $D_q = d_1, d_2, \dots, d_k$ from a large candidate pool.

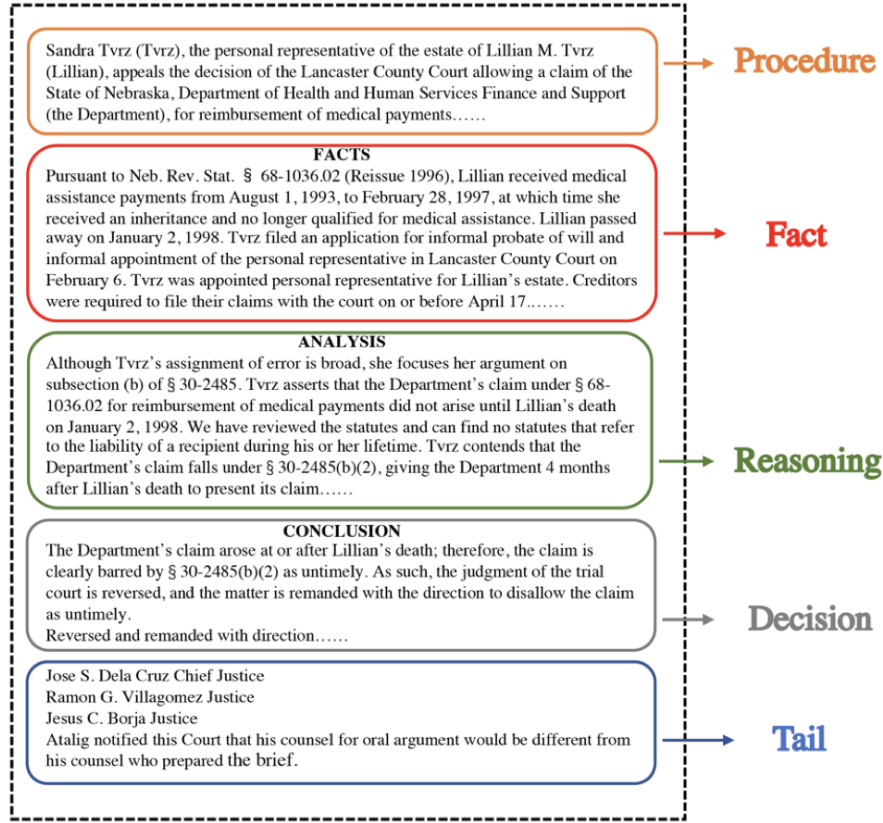


Figure 3.1: The legal structure of documents

In general search, these two paragraphs would be relevant to one another, as they are very similar. However, in a legal context, this would not be the case, because in Situation A a crime was committed and in Situation B no crime was committed.

To tackle these challenges [Structure-aware Pre-trained Language Model for Legal Case Retrieval \(SAILER\)](#) adopts an encoder-decoder architecture to explicitly model and captures the dependency between the Fact and other parts of a legal case document (first challenge). [SAILER](#) also utilizes the legal knowledge in the Reasoning and Decision paragraphs to enhance the understanding of key legal elements (second challenge). Let us describe in more detail how this model works.

As illustrated in Figure 3.1, a candidate case consists of Fact F , Reasoning R , Decision D and Tail T . However, in most practical legal case retrieval scenarios, q only consists of the basic Fact F , while candidates are complete case documents. Particularly, given the basic facts of a query case, lawyers or judges search with the legal case retrieval system to find relevant cases to the query case so that they can better perform their downstream tasks such as judicial judgments [41].

To accommodate the specificity of legal retrieval tasks, in [41] the architecture of [SAILER](#) consists of a query encoder and a document encoder (dual encoder), where the primary goal is to map the text to a high-dimensional embedding h_q and h_d . Dense

retrieval tasks typically make use of dual encoder systems², as in this case. The document encoder and the query encoder are implemented or initialized with a pre-trained language model which outputs an embedding representation (e.g., the embedding of the [CLS] token) by taking the raw text as inputs. These approaches are based on the intuition that the [CLS] token embedding should encode important information of the given text. With the [CLS] token, the dot product is used to compute the semantic relevance scores of h_q and h_d :

$$S(q, d) = h_q \cdot h_d \quad (3.2)$$

In the Fact Encoder, the authors of [41] first randomly replace some tokens of the Fact with a special token [MASK]. Only a small percentage of tokens is replaced, as the majority of the Fact needs to be preserved. Then, the masked input F_{mask} is transformed into a sentence vector by the Fact Encoder $\phi_F(\cdot)$. The final hidden state of the [CLS] token is taken as the representation of the entire sentence:

$$h_f = \phi_F(F_{mask}) \quad (3.3)$$

In a similar way to BERT, the masked facts serve as an input into the encoder to recover $m(F)$, the masked tokens. Then the Mask Language Modeling (MLM) loss is computed as one the pre-training objectives. This is defined as:

$$L_{MLM} = - \sum_{x' \in m(F)} \log p(x' | F \setminus m(F)) \quad (3.4)$$

The following stage of SAILER architecture is the Reasoning decoder. This was constructed through polluting the original text of the Reasoning section $R = [r_1, r_2, \dots, r_n]$. The Reasoning text was spitted into tokens and random selection of a subset $m(R)$ was made for masking. An aggressive mask ratio (30%-60%) is adopted for language reconstruction. In addition, the dense vector h_f replaces the original [CLS] vector which is commonly used as the beginning of decoder inputs [41]. The full decoder input can be formalized as:

$$[h_f, e_{r_1} + p_{r_1}, \dots, e_{r_n} + p_{r_n}] \quad (3.5)$$

where e_{r_1} and p_{r_1} denote the embedding and extra position embedding of r_k , respectively. The learning objective of the Reasoning Decoder is formulated as [41]:

$$L_{REA} = - \sum_{x' \in m(R)} \log p(x' | [h_f, R \setminus m(R)]) \quad (3.6)$$

The Reasoning Decoder needs to rely on h_f to recover the masked Reasoning section, which forces the h_f to focus on useful information related to the Reasoning section in the

²A dual encoder consists of two separate encoders that transform two different inputs into vector representations, which are then compared to produce a result.

facts. By this design, **SAILER** enhances attention to the key legal elements and captures the long dependencies between Facts and Reasoning.

Finally, the **SAILER** model also possesses a Decision Decoder. As explained in **SAILER**, for Chinese legal cases, the Decision usually consists of the relevant law articles, charges, and terms of penalty. Given a Decision section $D = [d_1, d_2, \dots, d_n]$, it was masked according to its specified format. Concretely, the Decision is usually formatted as follows: According to the Criminal Law of the People’s Republic of China $[z_1]$, the defendant committed $[z_2]$ and was sentenced to $[z_3]$, where z_1 denotes the specific law articles, z_2 represents the charges, and z_3 is the term of penalty. $\mathbb{Z}$ was to denote all masked tokens [41]. In other countries where the Decision does not have a specific format, a certain percentage of words with high TF-IDF values was selected which was then masked out. The masked tokens were also represented as $\mathbb{Z}$. The Decision Decoder input and loss function have a similar structure to those of the Reasoning Decoder input, relying on the dense vector h_F . The input is described as:

$$[h_F, e_{d_1} + p_{d_1}, \dots, e_{d_n} + p_{d_n}] \quad (3.7)$$

where e_{d_1} and p_{d_1} denote the embedding and extra position embedding of d_k , respectively. The loss function is formulated as:

$$L_{DEC} = - \sum_{x' \in \mathbb{Z}} \log p(x' | [h_F, D \setminus \mathbb{Z}]) \quad (3.8)$$

During the pre-training phase, the authors aimed at optimizing:

$$L_{Total} = L_{MLM} + L_{REA} + L_{DEC} \quad (3.9)$$

After pre-training, both decoders were dropped and the focus shifted to fine-tune the encoder. The purpose of fine-tuning is to make the query closer to related cases in the vector space compared to the irrelevant ones. Thus, given a query case q , let d^+ and d^- be relevant and negative cases, the loss function L is formulated as follows:

$$L(q, d^+, d_1^+, \dots, d_n^-) = -\log \frac{\exp(s(q, d^+))}{\exp(s(q, d^+)) + \sum_{j=1}^n \exp(s(q, d_j^-))} \quad (3.10)$$

In summary, the key idea of **SAILER** is to make full use of the structure relationships of legal case documents for pre-training. By reconstructing key legal elements and judgment results, **SAILER** generates better legal case representations and owns a stronger discriminate ability, achieving significant improvements over baselines in both low-resource and full-resource settings [41].

3.3 Named Entity Recognition in the Legal Domain

The data used in machine learning and artificial intelligence algorithms can have different formats. Notably, data tends to be labeled as structured, semi-structured and unstructured.

Structured data follows a defined schema and data type for each column, is the easiest one to treat. In the extreme opposite is the unstructured data which, as the name implies, does not have any inherent structure, meaning that the extraction of useful information and insights becomes much more difficult. A legal contract is an example of a source of unstructured data. Notwithstanding, these sources of unstructured data can contain very rich and important information which can be used to, for example, train many different models [28, 59, 54]. With this in mind, in [25], the authors developed a model to analyse the semantics and the structure of German legal contracts, with the aim to conduct NER.

Another example of a work which focuses on NER in the context of German Law is the one present in [40]. In that work, the authors, describe a dataset developed for NER in German federal court decisions, consisting of approximately 67,000 sentences with over 2 million tokens.

In the Greek legal landscape we have also seen some progress in the area of NER. In [3], the authors used 254 daily issues for classes A and D of the Greek Government Gazette over the period of 2000 to 2017. They also made use of Word2Vec, more specifically the skip-gram model, in an unlabelled corpus containing (i) 150,000 issues of the Greek Government Gazette in the period of 1990-2017; (ii) all publicly available pieces of legislation from the foundation of the Greek Nation in 1821 until 1990, which sum up to 50,000; (iii) 1,500 case laws published online by Greek Courts; (iv) all EU Treaties, Regulations and Decisions that have been translated to Greek and published in EUR-Lex; and (v) the Greek part of the European Parliament Proceedings Parallel Corpus, producing 100-dimensional word embeddings for a vocabulary of 428,963 words, based on 615 millions of tokens, included in the unlabelled corpus. These embedding then served as the input for a LSTM architecture network. In another work, this time for the Spanish Law, the authors of [53] tried to publish Spanish consolidated legislation strongly linked to other external resources in a structured form.

In the Portuguese context, we have also seen some recent contribution in this line of work. In [11], the authors finetuned a language model on a large intradomain corpus of unlabeled text, before training it on the final task and explored different scenarios considering two deep language architectures (ELMo and BERT), four unlabeled corpora and three legal NER tasks for the Portuguese language and showed that the BERT model, finetuned on a legal corpus, significantly improves on the state-of-the-art performance on the LeNER-Br corpus, a Portuguese language NER corpus for the legal domain.

DESIGN SPECIFICATION OF A LEGAL SEARCH FRAMEWORK

4.1 Task Definition

From the sections above it becomes clear that although the task of annotating case law with legal references is not completely new, its application to the Portuguese context has also been very limited, specially on what refers to European Portuguese. Nonetheless, in our initial vision, the major novelty in our work would consist of cross-referencing the case case-law decisions with the knowledge contained in the [Commented Legislation](#). The major reason which explains why a similar task has not been done before is the protection of intellectual property rights. In practical terms, any company which wants to make use of the knowledge base contained in [Commented Legislation](#) with a view to profit or gain any kind of economic advantage from it will need to reach a previous agreements with those protected by said intellectual property rights, in order for their consent to be given. This creates a substantial barrier on the use of this information. Since we benefit from the legal exception of using the data of the [Commented Legislation](#) to pursue an academic purpose, we are in a unique position to achieve the objectives presented before.

In this context, as mentioned before, our initial goal was to parse the decisions of the [Supremo Tribunal de Justiça \(the Supreme Court of Justice in Portugal\) \(STJ\)](#) and cross the implicit legal references made in the text of the decision to not only the law articles that may be relevant, but more interestingly to actual knowledge related to those articles. This knowledge will be extracted from [Commented Legislation](#) and it represents an approach which, to the best of our knowledge, has not been explored yet.

For instance, if a given case-law decision talks about murder of first degree, then our model should be able to explain what is a murder of first degree and how it is different from other types of murder, where is it prescribed within the legislation and what are the legal consequences of such a crime. In addition, if the [Commented Legislation](#) also provides examples of previous legal decision that have dealt with this first degree model, the model should also make references to those decisions.

4.2 Architecture

To start implementing our goal of achieving augmented case law analysis, we would be using the framework represented in Figure 4.1. First, as an offline pre-performed step, all the decisions that have been issued and published by the STJ up to and including 5 November 2023 are going to be pre-processed such that the named entity linking task is concluded in advance. In this way, the decisions indexed in Elastic Search will already have been treated. Further explanation of the named entity recognition process can be found in 4.3. With this part of the work concluded, we can then analyse the remaining components of the diagram. The user should start by passing a query to the system, related to the information he is interested in. After the query has been issued, we will perform some operations to it, such as rewriting to facilitate the search. Then, we will send a request to an Elastic Search service that has an index of the decisions of STJ which have already gone through the phase of named entity linking, as explained above. The response of Elastic Search is a list of documents ordered by their score of relevance relative to the query submitted by the user. The final step will then show the annotated case law. In Figure 4.1, the boxes with orange fill represent **processing** stages; boxes with purple fill represent **system** components; boxes with green fill represent **data resources**; and the box with blue fill represents the **final result**.

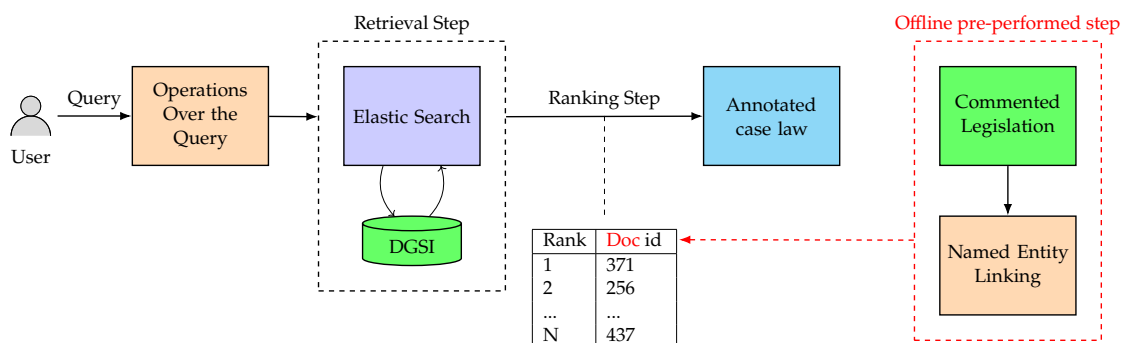


Figure 4.1: Proposed Architecture

4.3 Named Entity Recognition

The crucial step in our system architecture is thus the **NER** linking between the court decisions and the commented legislation. In the list below we enumerate the steps that have to be followed to implement a **NER** model in general while clarifying the specificities present in our particular project:

1. Entities of interest: the first step when developing a **NER** model should consist of deciding which entities we want to recognize. In our case, we want to recognize the

articles that support the case-law decisions (even if they are not expressly mentioned or referenced in the text of that decision);

2. Data collection: having defined the entities to be recognized, the next step is to gathered the data set to be annotated, in our case, the case law decisions. This has already been done in the course of the past months, as part of the work for the preparation of the final dissertation;
3. Data annotation: usually, the annotations to the relevant entities need to be manually done. However, in our project we will leverage the annotations present in [Commented Legislation](#). For this purpose, we collected the data present therein in the Portuguese Civil Code;
4. Choose a model architecture: several model could be used, as those following rule-based approaches, machine learning techniques and deep learning models. We will be focusing on the deep learning techniques as these models can automatically learn intrinsic features from data and have shown state-of-the-art performance in [NER](#) tasks. In particular, we intended to use [BERT](#);
5. Pre-processing: in this step we will be cleaning the data and ensure that all the inputs for the model follow the same format. For instance, the use of our web crawlers implied that the text of the decisions of the [Commented Legislation](#) has new lines commands in it which will need to be removed. In the pre-processing stage we will also performed tokenization of the inputs;
6. Training the model: After the pre-processing stage, we then proceed with training the model;
7. Evaluate and fine-tune the model: this is the final stage of our work and it is better described in Chapter [4.5](#).

4.4 Annotating Legal Descriptors

In the pursuit of enhancing legal text processing, we concentrated on the task of annotating legal descriptors for Portuguese case law decisions. Legal descriptors are succinct terms or phrases that encapsulate the fundamental legal concepts, issues, or themes present in a case. They serve as valuable metadata, facilitating efficient indexing, retrieval, and comprehension of legal documents.

To generate accurate legal descriptors, we employed GlorIA, a decoder-based [LLM](#) tailored for European Portuguese. GlorIA makes use of a causal language modeling objective with cross-entropy loss, enabling it to predict the next word in a sequence effectively — an essential capability for producing coherent and contextually appropriate descriptors.

A more detailed explanation of the work conducted and the results achieved can be found in Section 5.

4.5 Evaluation

The evaluation of the model's results would be done by legal experts. The method we proposed to use is as follows:

- first, the legal expert should submit a query to the model;
- the legal expert will then analyse the quality of the annotated case law taking into account the annotations made by the model, from a scale of 1 to 4: 1 - the annotations made are completely irrelevant and/or misleading; 2 - the annotations made are partially irrelevant and/or misleading; 3 - the annotations made are incomplete, but are mostly relevant and accurate; 4 - the annotations made are complete, accurate and relevant;
- finally, the expert should also identify the amount of annotations which were not made and should have been made in the expert's opinion such that it is possible to determine the percentage of annotation made relative to the total amount of annotations which should have been made.

GENERATION OF LEGAL DESCRIPTORS

5.1 Introduction and base model

As stated in Chapter 1, we proposed to develop a model that could generate descriptors for a given case law text, with a special focus on European Portuguese. The base model chosen to achieve this object, GlorIA, is a decoder-based LLM with an architecture similar to GPT-3’s [12], competing with it in linguistic, physical, and scientific reasoning tasks. Specifically, it adopts the GPT-Neo [8] 1.3B and 2.7B architectures, following the HuggingFace’s implementation of the model [42].

As described in [42], being a decoder, GlórIA uses a causal language model pre-training objective, using cross-entropy as its loss. GPT-Neo also employs local attention [5], which replaces standard self-attention and combines a dilating sliding window strategy with pre-selected global attention on some input locations, making the self-attention scale linearly, and linear attention [56], which optimizes the dot-products by providing linear memory and processing complexities while maintaining representational capability.

In [42], the authors also described how the model was trained. To pre-train GlórIA, a total batch size of 512 was used (128 p/ GPU), with 16 gradient accumulation steps. A GPT-2-like BPE tokenizer was used, with a vocabulary size of 50257 tokens. Training was performed with BF-16 mixed-precision and a weight decay of 0.01. For the 1.3B version, GlórIA was trained for a total of 3M steps, on 4x NVIDIA A100s 40GB, for a total of 21 days (7 days p/ 1M steps), while, for the 2.7B, due to hardware resource constraints, it was trained only for 1M steps on 7x NVIDIA A100s (10 days). A cosine annealing scheduler was used for both models, with hard restarts every 500k steps and 10k warmup steps. Periodic evaluations and data shuffling were conducted every 1 million steps.

In our work, we utilized the smaller version of GlorIA, which has 1.3 billion parameters. This choice was deliberate, as the smaller model offers several advantages, including reduced computational resource requirements and faster processing times, while still providing a robust performance for our specific tasks. Additionally, the smaller model’s efficiency makes it more accessible for environments with limited hardware capabilities, allowing for quicker iterations and more cost-effective experimentation.

5.2 Datasets and pretraining

Given the specificity of the legal language, and although GlorIA was already a robust model on what concerns European Portuguese, our first intent was to provide it with targeted legal domain knowledge and to use specialized training data to enhance its performance and accuracy in legal text processing.

In this context, we pre-trained GlorIA using the text of all decisions published by the STJ up to November 2023¹. The goal was to enable the decoder to learn and predict the most probable token in a sequence of legal text. To achieve this, we employed a web-crawler specifically developed for the task, utilizing widely available Python libraries such as BeautifulSoup to efficiently gather the necessary data.

In addition to the full text of the decisions, we also collected supplementary information, including the summaries and descriptors associated with each decision. These summaries and descriptors are elaborated by the judges who write the main text of the decisions, providing a reliable and trustworthy source of context and insight. By incorporating this additional data, we ensured that the model had access to comprehensive and high-quality inputs, enhancing its ability to understand and generate accurate legal text.

The number of total decisions collected by the web-crawler was 64010. However, a substantial number of these decisions did not have the complete information, due to the fact that the decision was made a long time ago. This meant that in some decisions we did not have access to the complete text of that decision and because of that we were forced to drop that particular decision. In the end, the model was trained with 28,888 different decisions, which corresponds to all those deemed valid, where the full set of data was available. Figure 5.1 shows a sample of a valid decision used to train the model. Figure 5.2 shows an example of an invalid decision which was excluded from the training phase.

Once the model had grasped some of the specificities of the legal domain, we then proceed to fine-tuning it in the task of generating the descriptors for each decision.

5.3 Training GlorIA on the Legal Domain

During the pre-training phase, we experimented with a variety of parameter configurations to determine the optimal settings for our specific use case. We explored many different configurations to fine-tune the model's performance, and below we discuss four representative examples of these experiments. Each example highlights specific parameter adjustments and the rationale behind them

5.3.1 Model 1

- Batch Size (b): 32
- Learning Rate (lr): $5e^{-5}$

¹Decisions are made available on www.dgsi.pt

- Maximum Length (ml): 256
- Gradient Accumulation Steps (ga): 8
- Epochs (e): 5
- Weight Decay (wd): 0.02
- Warmup Steps (ws): 5000
- Max Steps (ms): 100000
- Logging Steps (ls): 50
- Save Steps (ss): 2500
- Half Precision Training (hrr): 2
- Scheduler: Linear
- Rational: In this setup, a smaller batch size of 32 was used, which can lead to more frequent updates but might increase the training time due to the smaller data

processed per update. A lower learning rate of $5e-5$ may result in more stable training but could slow down convergence. The maximum length of 256 tokens allows the model to handle shorter sequences more effectively, while fewer gradient accumulation steps (8) means more frequent updates. With 5 epochs, the model might not fully capture complex patterns if the data requires longer training. A weight decay of 0.02 introduces regularization to prevent overfitting, and a smaller number of warmup steps (5000) may lead to quicker learning rate adjustments. The max steps were set to 100000 to control training duration, with logging and save intervals set to capture frequent checkpoints and updates.

5.3.2 Model 2

- Batch Size (b): 128
- Learning Rate (lr): $1e^{-4}$
- Maximum Length (ml): 512
- Gradient Accumulation Steps (ga): 16
- Epochs (e): 15
- Weight Decay (wd): 0.01
- Warmup Steps (ws): 10000
- Max Steps (ms): 200000
- Logging Steps (ls): 100
- Save Steps (ss): 5000
- Half Precision Training (hrr): 4
- Scheduler: Cosine
- Rational: Using a larger batch size of 128 allows the model to process more data per update, potentially speeding up convergence but requiring more memory. A higher learning rate of $1e^{-4}$ accelerates learning but can risk instability. A maximum length of 512 tokens accommodates longer sequences, enhancing the model's ability to handle complex texts. More gradient accumulation steps (16) smooth out updates by accumulating gradients over multiple steps, which can be useful with large batch sizes. Increasing the epochs to 15 allows more extensive training, potentially improving performance on more complex datasets. Weight decay of 0.01 provides regularization to manage overfitting, while a larger number of warmup steps (10000) helps in a smoother transition to the target learning rate. Longer max steps (200000) and more frequent logging and saving of checkpoints ensure detailed monitoring

and recovery options. The use of a cosine scheduler helps in gradually decreasing the learning rate, which can improve final performance.

5.3.3 Model 3

- Batch Size (b): 64
- Learning Rate (lr): $1e^{-3}$
- Maximum Length (ml): 128
- Gradient Accumulation Steps (ga): 4
- Epochs (e): 10
- Weight Decay (wd): 0.005
- Warmup Steps (ws): 2000
- Max Steps (ms): 50000
- Logging Steps (ls): 25
- Save Steps (ss): 1000
- Half Precision Training (hrr): 8
- Scheduler: Cosine-Hard
- Rational: A batch size of 64 offers a balance between memory efficiency and update frequency. The higher learning rate of $1e^{-3}$ facilitates faster learning but may lead to instability if not properly managed. With a maximum length of 128 tokens, the model can handle shorter sequences, which might be suitable for tasks requiring concise inputs. Fewer gradient accumulation steps (4) result in less smoothing of updates, potentially leading to noisier training. Training for 10 epochs provides adequate exposure to the data while avoiding overfitting for moderate dataset complexity. A weight decay of 0.005 introduces mild regularization to prevent overfitting. Fewer warmup steps (2000) help in quickly reaching the target learning rate, which can accelerate initial learning phases. The max steps of 50000 and more frequent logging and saving (every 1000 steps) ensure detailed monitoring and recovery throughout training. The cosine-hard scheduler adjusts the learning rate according to a cosine function with a sharp drop towards the end of training. This approach helps the model converge more smoothly, as it reduces the learning rate progressively, allowing for finer adjustments in the final stages of training.

5.3.4 Model 4

- Batch Size (b): 64
- Learning Rate (lr): $1e^{-4}$
- Maximum Length (ml): 512
- Gradient Accumulation Steps (ga): 16
- Epochs (e): 10
- Weight Decay (wd): 0.01
- Warmup Steps (ws): 10000
- Max Steps (ms): 100000
- Logging Steps (ls): 100
- Save Steps (ss): 5000
- Half Precision Training (hrr): 4
- Scheduler: Cosine-Hard

With the batch size of 64 we were able to strike a balance between efficient memory usage and stable gradient estimation, allowing for effective training without overwhelming computational resources. A learning rate of $1e^{-4}$ was set to facilitate stable convergence; it is high enough to ensure meaningful progress but controlled enough to prevent instability during training.

We set the maximum length to 512 tokens to handle longer sequences, which is essential for processing detailed legal texts. The gradient accumulation steps were configured to 16, enabling the accumulation of gradients over multiple batches to simulate a larger batch size, thus stabilizing training while managing memory usage effectively.

Training was scheduled for 10 epochs, providing adequate exposure to the data while avoiding overfitting. A weight decay of 0.01 was employed to introduce moderate regularization, balancing between preventing overfitting and maintaining model flexibility.

Warmup steps were set to 10,000 to gradually increase the learning rate, ensuring a smooth transition and stabilizing the early training phases. The maximum number of steps was configured to 100,000, allowing extensive training to refine the model's weights thoroughly.

We chose to log training progress every 100 steps to enable detailed monitoring and timely adjustments. Checkpoints were saved every 5,000 steps to provide periodic snapshots of the model's state, ensuring recovery options and evaluation opportunities.

Finally, half-precision training with an hrr value of 4 was utilized to enhance computational efficiency, reducing memory usage and speeding up training. The cosine-hard

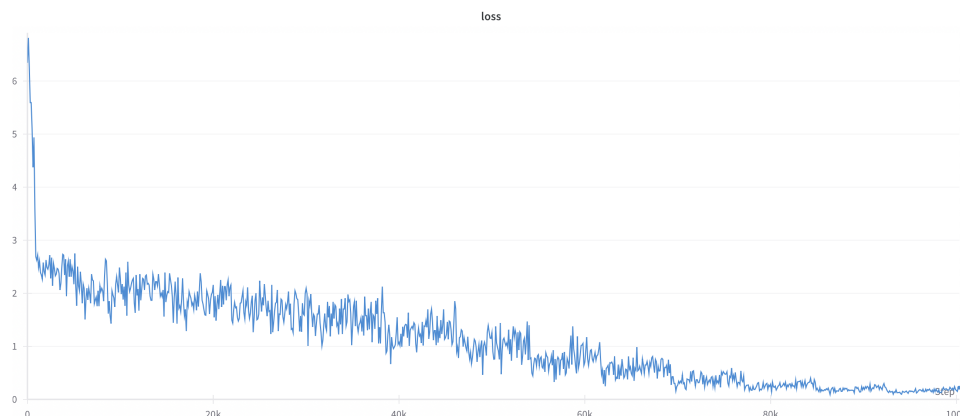


Figure 5.3: Evolution of the training loss, as a function of the training steps

scheduler was selected to adjust the learning rate according to a cosine function with a sharp reduction towards the end of training, helping to fine-tune the model and achieve stable convergence by progressively decreasing the learning rate.

An evolution of the training loss can be seen below in Figure 5.3.

5.4 Model Derivatives

Besides training the model with the text of the case law decisions from the [STJ](#), we decided to build two variations to see if we could improve our results of generating descriptors from the summary of the legal decisions.

The first derivative of the model, besides taking into account the text of the case law decisions, was also trained on the part of the legal corpus. In particular, we decided to train the GlorIA with Civil Code articles, due to its pivotal importance in the legal system of Portugal. For this purpose, we developed another web-crawler that extracted from [Diário da República](#) all articles from the Civil Code².

The second derivative of the model, on top of being trained with the case law from [STJ](#) and the legal text of the Civil Code, was also trained with [Commented Legislation](#). More specifically, we acquired the e-book “*Código Civil Anotado - Volume I*” - the first volume of annotations to the Portuguese Civil Code³ - and used part of the articles to train GlorIA. It was not possible to use all of the annotated articles as the format in which they were extracted from the web-crawler used was not the ideal one. This meant that a lot of manual work had to be done and given our time constraints this made it impossible to use all of the articles. In total, we used ca. 150 annotated articles.

In summary, we developed the following variations of our base model:

- Base Model - trained with the case law decisions;

²The Civil Code can be accessed on <https://diariodarepublica.pt/dr/legislacao-consolidada/decreto-lei/1966-34509075>.

³This book can be acquired at <https://www.almedina.net/c-digo-civil-anotado-volume-i-1563805414.html>.

- Variation 1 - trained with the case law decisions and all articles of the Portuguese Civil Code;
- Variation 2 - trained with the case law decisions, all articles of the Portuguese Civil Code and the first 150 articles of the Annotated Civil Code.

5.5 Results and Discussion

We evaluated our different models using a sample of unseen summaries on which the model was prompted to generate descriptors. We then evaluated the results of the different models with ChatGPT, by asking it to categorize the generated summaries on one of three different categories: not relevant; somewhat relevant; extremely relevant. We used GPT 3.5 Turbo to make the evaluation.

A side note is due at this stage. Initially, we tried to use more standard metrics, like BLEU, ROUGE and METEOR to evaluate the quality of the results generated. However, these metrics proved to be inadequate for two reasons: first, the descriptors that are associated with each decision are not exhaustive, which means that there could be descriptors which are relevant but were not included in the list of the suggested descriptors by the judge who wrote them; second, there could be descriptors that have the same semantic meaning, but different words are used to express that meaning. This meant that using metrics as BLEU, ROUGE and METEOR returned scores that were very low and did not reflect the true quality of the descriptors generated. At this point, the only available solution was to manually annotate the quality of each descriptor or to make use of Chat-GPT to perform this work for us. Given the vast amount of examples to evaluate, we decided to go for the second option.

A sample of the model generation can be seen below, together with the summary that acted as the base for the generation of the descriptors:

1	summary:
2	I - Sendo indiscutível que o dano biológico assume extrema relevância e como tal deve ser especialmente valorado aquando da fixação da indemnização devida ao lesado, não existe ainda uma posição consistente na jurisprudência sobre a sua qualificação como dano não patrimonial ou como dano patrimonial futuro, ou mesmo comotertium genus, vislumbrando-se uma tendência para o qualificar como dano não patrimonial quando o lesado está já reformado e não se prevê que o défice funcional permanente de integridade físico-psíquica afecte a sua capacidade de ganho por não existir à data do acidente actividade profissional a considerar.
3	II - Independentemente da sua qualificação, sendo o dano biológico em si mesmo uma violação da integridade físico-psíquica do lesado, o dano existe mesmo que o lesado já não desempenhe actividade profissional.

4 III - Ponderando a idade da lesada à data do sinistro (65 anos), o período de esperança m
 5 édia de vida para as mulheres em Portugal, o tipo de lesões de que ficou a padecer e
 6 sua relevante repercussão nos actos do dia-a-dia, que acarretam limitações ao nível
 7 da autonomia e liberdade de movimentos, assim como o Défice Funcional Permanente da
 Integridade Físico- Psíquica de 26 pontos, mostra-se justa e adequada a fixação da
 indemnização devida pelo dano biológico, com recurso à equidade, na importância
 peticionada de EUR 35.000,00, valor esse que faz a devida ponderação daquilo que vem
 sendo decidido pelos tribunais superiores em casos similares.

5
 6 descriptors:
 7 DANO BIOLÓGICO, DANOS FUTUROS, DIREITO À VIDA, EQUIDADE, CÁLCULO DA INDEMNIZAÇÃO, DANOS

Listing 5.1: A sample of the descriptors generated

In this particular case, GPT-3.5 Turbo classified the generation as extremely relevant. The overall distribution of the models' results was as follows:

GPT Model	GlorIA Model	Not relevant	Somewhat relevant	Extremely relevant
3.5 Turbo	Case law	1594 (13.24%)	5678 (47.16%)	4769 (39.61%)
3.5 Turbo	Case law + Civil Code	1363 (12.14%)	5284 (47.08%)	4576 (40.77%)
3.5 Trubo	Case law + Civil Code + Annotated law	1367 (12.15%)	5341 (47.48%)	4540 (40.36%)

Table 5.1: The results of the different models as evaluated by GPT 3.5 Turbo

Before discussing these results, we should point out that the reason which explains why the sum of the examples in the 3 categories (not relevant, somewhat relevant and extremely relevant) is not the same in the 3 different models occurs due to the fact that ChatGPT categorize some instances outside the categories considered. For instance, in some cases the model returned categories like “nada relevante descritor”, “muito relevante descritor” or “alguma relevante”. These cases were not considered. The total number of descriptors that the different models evaluated was 14,449.

From Table 5.1 a few aspects emerge as noteworthy. First, it seems clear that adding the civil code and the annotated legislation to the model contributed to a decrease of the non-relevant descriptors, suggesting that additional context helps the model in filtering out irrelevant information better. This difference might have been even more significant if the added context had been broader. For example, if the model had been trained with legislation from other law fields (such as criminal law, tax law, labour law...) it would make sense that the not relevant descriptors would reduced even more.

At the same time, while additional context improves the model's performance in identifying non-relevant results, it seems to not have a big impact on the examples classified as “Somewhat relevant”. It slightly decrease the number of examples in this category for the case of case law and civil code, but it slightly increased for the case of the case law, civil code and annotated law.

We can also see a slight increase on the examples categorized as “Extremely relevant” when more context is added for the last two variations of the model. This seems to

strengthen the case mentioned above - adding more context increases the overall model performance.

Lastly there seems to be no significant difference between the second model (trained with case law and civil code) and the third model (trained with case law, civil code and annotated legislation). This lack of difference can be explained by the small amount of data contained in the annotated legislation used to train the model, as explained in [Section 5.3](#).

Overall, it seems fair to conclude that adding more legal knowledge helps in refining the results to reduce irrelevance and it can also contribute to the model's ability to pinpoint extreme relevance.

5.6 Take away lessons

Originally, our goal was to enhance the annotation of case law decisions by cross-referencing them with the extensive legal knowledge contained in the [Commented Legislation](#). We aimed to perform [NER](#) linking between court decisions and the commented legislation, thereby enriching the legal texts with detailed explanations, legal consequences, and references to related cases. This approach promised to provide a deeper understanding of the legal reasoning and facilitate more informed legal research.

However, as detailed in the previous chapter, we faced significant legal and technical hurdles that necessitated a shift in our strategy. The legal constraints surrounding the use of [Commented Legislation](#), particularly the blockage of our access to the Annotated Civil Code, prevented us from utilizing this resource as initially planned. Moreover, technical challenges related to the complexity of parsing legal texts and performing [NER](#) at the required level of sophistication further impeded progress.

In light of these challenges, we reoriented our focus toward generating legal descriptors for case law texts using GlorIA, a decoder-based [LLM](#) specialized in European Portuguese. This pivot allowed us to stay aligned with our overarching objective of enriching legal texts with additional knowledge, but through a different, more feasible approach.

The generation of legal descriptors serves as a proof of concept for extending annotations with more legal knowledge. Descriptors are concise terms or phrases that capture the essential legal concepts, issues, or themes present in a case. They function as metadata that can significantly enhance the indexing, retrieval, and understanding of legal documents. By automatically generating these descriptors, we aim to assist legal professionals and researchers in quickly identifying relevant cases and grasping the core legal matters involved.

Our work builds upon the foundation laid by the initial proposal in several ways:

5.6.1 Utilizing Specialized Legal Texts

While we were unable to directly use the annotated legislation from [Commented Legislation](#) in its entire extent, we remained committed to incorporating specialized legal texts into our model’s training data. By pre-training GlorIA on the full text of decisions published by the [STJ](#), along with associated summaries and descriptors crafted by judges, we ensured that the model was exposed to high-quality, authoritative legal language and reasoning. This exposure is crucial for the model to understand the nuances of legal discourse and generate accurate descriptors.

Furthermore, we expanded the model’s knowledge base by including the text of the Portuguese Civil Code and a selection of annotated articles from the Annotated Civil Code.

The initial plan involved performing [NER](#) linking between case law decisions and specific legal entities within the annotated legislation. Given the obstacles encountered, we adapted this objective by shifting our focus to the generation of descriptors, which can be seen as an alternative form of annotation. Instead of linking entities within the text to external knowledge bases, we trained the model to generate descriptors that encapsulate the key legal concepts and issues present in the case.

This approach maintains the spirit of our original goal—to augment legal texts with additional layers of information that enhance understanding and accessibility. By generating descriptors, we provide a means to summarize and categorize legal decisions, facilitating more efficient legal research and analysis.

5.6.2 Leveraging GlorIA for Legal Language Processing

Our choice to use GlorIA, a model designed for European Portuguese, aligns with the initial objective of addressing the underrepresentation of Portuguese in legal [NLP](#) applications. GlorIA’s architecture and training make it well-suited for processing legal texts in Portuguese, and our additional pre-training and fine-tuning steps further specialize the model for legal applications.

By training GlorIA on legal decisions, statutory texts, and annotated articles, we equipped the model with a deep understanding of legal language and concepts. This specialization is critical for generating accurate and relevant descriptors, as it allows the model to capture the subtle nuances and terminologies specific to the legal domain.

While our current work focuses on generating descriptors, it lays the groundwork for future efforts to extend annotations with more comprehensive legal knowledge. The methodologies and insights gained from this project can inform subsequent attempts to incorporate additional legal texts, such as other branches of law or more extensive annotated legislation, should the legal constraints be resolved.

Moreover, the successful training and deployment of GlorIA for descriptor generation demonstrate the feasibility of using LLMs for specialized legal tasks in Portuguese. This achievement opens the door for further research and development in this area, potentially

leading to more advanced applications like automated legal summarization, question answering, or even the revival of the original NER linking objectives under more favorable conditions.

5.6.3 Addressing the Initial Objectives Through Alternative Means

Our initial vision included providing explanations of legal concepts, identifying relevant statutes, and referencing related cases—all aimed at enriching the legal text for the user. By generating descriptors, we achieve a similar outcome through alternative means. Descriptors act as concise representations of the legal issues at hand, effectively summarizing the case’s essential elements. They assist users in quickly understanding the nature of the case and its relevance to specific legal topics.

Furthermore, our evaluation of the model’s performance, particularly the increase in extremely relevant descriptors with additional legal context, indicates that the approach is effective in enhancing the information available to users. Although we could not provide detailed annotations linked to external knowledge bases, the descriptors serve as a valuable tool for legal professionals and researchers.

By pivoting to descriptor generation, we navigated around the legal constraints that hindered our initial plan. Generating descriptors based on publicly available case law texts does not infringe on intellectual property rights, as it utilizes data that is openly accessible and permissible for academic use.

Technically, focusing on descriptor generation allowed us to mitigate some of the challenges associated with NER linking and the complexities of parsing and annotating legal texts. While still a demanding task, generating descriptors is more tractable within the project’s time frame and resource limitations.

5.6.4 Conclusion

In summary, our work on generating legal descriptors represents a strategic adaptation of our initial proposal, aligning with our core objectives while accommodating the practical constraints encountered. By using GlorIA to generate descriptors from case law summaries, we extended the annotations of legal texts with additional knowledge in a feasible and legally compliant manner.

This approach serves as a proof of concept for enriching legal documents with machine-generated insights, contributing to the advancement of legal NLP applications in European Portuguese. It demonstrates the potential of LLMs in supporting legal research and offers a foundation for future explorations into more sophisticated annotation and information retrieval systems within the legal domain.

CONCLUSIONS AND FUTURE WORK

6.1 Overview of the Work Conducted

In this thesis, we explored the application of a [LLMs](#) within the legal domain, focusing specifically on European Portuguese. Recognizing the complexities and unique characteristics of legal language, we utilized GlorIA, a decoder-based [LLM](#) developed and trained in European Portuguese. Our objective was to enhance GlorIA's capabilities by fine-tuning it with specialized legal texts, including case law from the Portuguese Supreme Court ([STJ](#)), the text of the Portuguese Civil Code, and selected parts of the Annotated Portuguese Civil Code ([Commented Legislation](#)).

Our primary goal was to develop a model capable of generating legal descriptors from the summaries of court decisions. We also experimented with other tasks, such as generating summaries from the full text of court decisions and determining the relationship between legal articles. While the model did not achieve satisfactory results in all tasks, it demonstrated promising performance in generating legal descriptors from case summaries, showcasing its potential utility in legal research and documentation.

To evaluate the model's outputs, we employed ChatGPT (GPT-3.5 Turbo), leveraging its capabilities to assess the relevance of the generated descriptors. By categorizing the descriptors into scales of relevance, we were able to quantitatively analyze the model's performance across different configurations and datasets.

6.2 Reasons for Abandoning the Initial Work Proposal

After careful consideration and analysis of the preliminary results obtained from our initial work, several critical factors have led to the decision to abandon our design specification of a legal search framework as outlined in [Figure 4.1](#). These factors can be categorized into technical challenges, legal constraints, and overall project feasibility.

6.2.1 Technical Challenges

One of the major technical challenges we encountered was the complexity of parsing case law decisions and performing [NER](#) at the level of sophistication required. While the proposed use of a [BERT](#)-based model for [NER](#) showed some promise, we quickly realized that it would require far more training data than initially anticipated. Although leveraging the existing annotations in [Commented Legislation](#) provided a foundation, the nuances of legal language, the implicit nature of some references, and the varying formats of decisions posed significant obstacles. Further pre-processing steps were needed to handle these inconsistencies, which added considerable complexity to the pipeline.

6.2.2 Legal Data

A significant factor that further complicated the project was the legal constraints around the use of the data in [Commented Legislation](#). While we initially believed that the academic exception would provide sufficient freedom to use this data in a meaningful way, it became evident that the scope of usage allowed under this exception was narrower than anticipated. Although we could access and utilize the information for academic purposes, we quickly encountered a major setback when the account used to purchase the Annotated Civil Code was blocked, restricting our access to this essential resource.

This incident revealed a broader issue: not only was our access to the Annotated Civil Code compromised, but it also became clear that we would face similar barriers with any other annotated legislation. Initially, we had intentions of expanding the system by incorporating various annotated legal texts, including those covering different branches of law, such as criminal or administrative law. However, the account blockage underscored the fragility of relying on third-party sources of annotated legislation that are governed by strict intellectual property laws. It became evident that, even under the academic exception, we could not systematically process or integrate large volumes of proprietary legal data without potentially violating the terms of service of these legal databases.

The risk of facing further restrictions, or even legal repercussions, from using additional annotated legal materials forced us to reconsider the scope of the project. We realized that the same legal constraints that affected the Civil Code would likely extend to any other annotated legal resources we might attempt to use. This limitation severely hindered our ability to enrich case law analysis with in-depth legal commentary, which was a core element of the project's initial vision.

6.2.3 Feasibility and Time Constraints

From a timely completion perspective, the scope of the work proved to be larger than what could be feasibly accomplished within the given time frame. While the architecture and theoretical framework were sound, the practical implementation required significantly more resources than originally anticipated. The data pre-processing, model training,

and evaluation stages each introduced new, unforeseen complexities that stretched the timeline beyond the limits.

Given the significant barriers related to both the technical execution and legal framework, continuing with the initial proposal would have required a fundamental rethinking of the project. Moreover, the time necessary to fine-tune the model and achieve an acceptable level of accuracy was incompatible with the deadlines we faced. Therefore, it was deemed more efficient to pivot the research focus toward an alternative approach.

6.3 Challenges and Limitations

On top of the factors that forced us to move away from the initial architecture, throughout the course of this research, we encountered several other challenges (some of them closely linked to those already identified in the previous section) that impacted the model's performance, as outlined below:

- **Data Availability and Quality:** one of the significant challenges was the availability and quality of the data required for training and fine-tuning the model. Legal texts are often not readily available in formats suitable for machine learning applications. A substantial amount of effort was required to gather the data using custom web crawlers and then process the extracted information into a format compatible with the training requirements. The diversity of the sources—including case law decisions, statutory texts, and annotated articles—meant that data cleaning and pre-processing consumed a considerable portion of the project timeline. Limited access to comprehensive and high-quality datasets constrained the model's ability to learn and generalize effectively across different legal contexts.
- **Legal Constraints:** the legal restrictions surrounding the use of certain datasets, particularly the Annotated Civil Code ([Commented Legislation](#)), posed significant limitations. As discussed in previous the previous section, we faced obstacles when the account used to access the Annotated Civil Code was blocked, which prevented us from utilizing this valuable resource fully. These constraints hindered our ability to incorporate extensive annotated legislation into the model, thereby limiting the breadth of legal knowledge it could assimilate. This limitation also affected our initial plan to perform named entity recognition [NER](#) linking between court decisions and the commented legislation, as explained previously.
- **Model Capacity and Computational Resources:** we utilized the smaller version of GLoRe, containing 1.3 billion parameters, due to computational resource constraints. While this version is efficient and requires less processing power, it may not capture the full complexity and nuances of legal language compared to larger models. Tasks such as generating summaries from full legal decisions, which involve understanding

and condensing extensive and intricate texts, may require more substantial model capacities to achieve satisfactory results.

- **Complexity of Legal Language:** legal language is characterized by specialized terminology, complex syntactic structures, and context-dependent meanings. The model struggled with tasks that required deep comprehension of these complexities, such as generating accurate summaries or understanding the relationships between legal articles. The inherent difficulty of processing legal texts posed a significant challenge, highlighting the need for models specifically tailored and extensively trained on legal language.
- **Evaluation Challenges:** traditional evaluation metrics like BLEU, ROUGE, and METEOR proved inadequate for assessing the quality of the generated descriptors due to the non-exhaustive nature of the descriptors and the variability in legal phrasing. This necessitated the use of alternative evaluation methods, such as leveraging GPT-3.5 Turbo, which, while effective, introduces dependencies on external models and may not align perfectly with human expert evaluations.

6.4 Contributions

Despite the challenges faced, this work makes several contributions to the field of legal natural language processing [NLP](#) and demonstrates the potential of [LLMs](#) in specialized domains:

1. **Development of a Legal Descriptor Generation Model:** we successfully fine-tuned GlorIA to generate legal descriptors from case summaries. This model serves as a proof of concept for the application of [LLMs](#) in automating the annotation and indexing of legal documents, which can significantly aid legal professionals in research and case analysis.
2. **Enhancement of GlorIA's Legal Proficiency:** by training GlorIA on specialized legal texts, we expanded its understanding of legal language and concepts within European Portuguese. This specialization contributes to addressing the underrepresentation of European Portuguese in legal [NLP](#) resources and tools.
3. **Innovative Evaluation Methodology:** we implemented an evaluation approach using ChatGPT (GPT-3.5 Turbo) to assess the relevance of the generated descriptors. This method allowed for efficient and scalable evaluation of a large dataset, overcoming the limitations of traditional [NLP](#) evaluation metrics in the legal domain.
4. **Insights into Model Limitations and Data Challenges:** the exploration of tasks where the model underperformed provides valuable insights into the limitations of current [LLMs](#) in handling complex legal tasks. It emphasizes the importance of data quality, model capacity, and domain-specific training in achieving desired outcomes.

5. **Adaptation to Legal Constraints:** our work demonstrates how to navigate legal and ethical considerations in research, particularly regarding data usage and intellectual property rights. By adjusting our approach in response to these constraints, we ensured compliance while still advancing our research objectives.

6.5 Future Work

Building on the findings and contributions of this research, several avenues for future exploration and enhancement are identified:

Building upon the foundations established in this thesis, several avenues for future research and development present themselves, promising to enhance the capabilities and applications of our legal language model.

One primary direction is the expansion and diversification of the training dataset. Acquiring a more extensive and varied collection of legal documents—including statutes, case law from different courts, and legal commentaries across various areas of law such as criminal, administrative, and labor law—would significantly enrich the model’s knowledge base. Collaborations with legal institutions or leveraging open-access legal resources could mitigate current data limitations, providing the model with a broader perspective of the legal landscape and improving its ability to generate accurate and relevant descriptors.

Utilizing larger and more advanced models is another promising avenue. Experimenting with larger versions of GLoVe or other state-of-the-art large language models could enhance performance on more complex tasks.

Enhancing data pre-processing techniques is also crucial for future development. Developing more sophisticated data pre-processing pipelines—including advanced parsing and formatting tools—would streamline the preparation of legal texts for training. Automation in data cleaning and structuring can save time and reduce errors, ensuring that the model is trained on high-quality data.

With improved data access and pre-processing capabilities, revisiting the initial goal of performing named entity recognition [NER](#) and legal entity linking becomes feasible. Achieving this would bring us closer to the original vision of enriching legal texts with detailed annotations and cross-references, significantly benefiting legal research and practice.

Ethical considerations and legal compliance must continue to be a priority in future research. Integrating ethical guidelines—including data privacy, bias mitigation, and adherence to legal standards—is imperative for responsible [AI](#) deployment in legal contexts. Establishing clear frameworks for the ethical use of [AI](#) in the legal field will ensure that the technology supports justice and fairness, aligning with the fundamental values of the legal profession. Moreover, developing user-centric applications can greatly enhance the practical adoption of the model. Creating user-friendly interfaces or applications that allow legal professionals to interact seamlessly with the model would facilitate its

integration into everyday legal work. Features such as customizable descriptor generation, interactive querying, and integration with existing legal databases would improve usability and make the model a valuable tool for practitioners.

In conclusion, there is substantial potential for advancing this work in various directions. By addressing data limitations, enhancing model capabilities, improving pre-processing and prioritizing ethical considerations, future research can significantly enhance the utility and impact of legal language models. These efforts will contribute to the development of sophisticated AI tools that support legal professionals, enhance access to justice, and foster innovation at the intersection of law and technology.

6.6 Final Remarks

This research underscores both the potential and the complexities involved in applying LLMs to specialized domains such as law. While challenges related to data availability, legal constraints, and model limitations exist, the promising results achieved in generating legal descriptors demonstrate that AI can play a valuable role in supporting legal professionals.

The journey of this project reflects the dynamic nature of interdisciplinary research, where technical innovation must align with domain-specific requirements and constraints. Our ability to adapt to challenges—such as pivoting from the initial plan due to legal restrictions—highlights the importance of flexibility and problem-solving in research.

By contributing to the field of legal NLP, particularly in the context of European Portuguese, this work lays the groundwork for future advancements. The insights gained here can inform subsequent efforts to develop more sophisticated models, improve data handling methods, and address ethical considerations in AI applications.

Ultimately, the integration of AI into legal processes holds the promise of increasing efficiency, accessibility, and understanding within the legal system. Continued research and collaboration across disciplines will be essential in realizing this potential, ensuring that technological advancements align with the needs and values of the legal community.

BIBLIOGRAPHY

- [1] K. J. Adebayo, G. Boella, L. Di Caro, et al. “Neural Reasoning for Legal Text Understanding.” In: *JURIX*. 2016, pp. 175–178.
- [2] M. R. AI4Science and M. A. Quantum. “The Impact of Large Language Models on Scientific Discovery: a Preliminary Study using GPT-4”. In: *arXiv preprint arXiv:2311.07361* (2023).
- [3] I. Angelidis, I. Chalkidis, and M. Koubarakis. “Named Entity Recognition, Linking and Generation for Greek Legislation.” In: *JURIX*. 2018, pp. 1–10.
- [4] I. Beltagy, K. Lo, and A. Cohan. “SciBERT: A pretrained language model for scientific text”. In: *arXiv preprint arXiv:1903.10676* (2019).
- [5] I. Beltagy, M. E. Peters, and A. Cohan. “Longformer: The long-document transformer”. In: *arXiv preprint arXiv:2004.05150* (2020).
- [6] E. M. Bender et al. “On the dangers of stochastic parrots: Can language models be too big?” In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 2021, pp. 610–623.
- [7] L. Beyer. [Public, Approved] *Intro to Transformers*. Accessed: 20 November 2023. URL: https://docs.google.com/presentation/d/1ZXFIhYczos679r70Yu8vV9u06B1J0ztzeDxbnBxD1S0/edit#slide=id.g13dd67c5ab8_0_3666.
- [8] S. Black et al. “Gpt-neox-20b: An open-source autoregressive language model”. In: *arXiv preprint arXiv:2204.06745* (2022).
- [9] D. M. Blei, A. Y. Ng, and M. I. Jordan. “Latent dirichlet allocation”. In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.
- [10] T. Bolukbasi et al. “Man is to computer programmer as woman is to homemaker? debiasing word embeddings”. In: *Advances in neural information processing systems* 29 (2016).

- [11] L. H. Bonifacio et al. "A study on the impact of intradomain finetuning of deep language models for legal named entity recognition in portuguese". In: *Intelligent Systems: 9th Brazilian Conference, BRACIS 2020, Rio Grande, Brazil, October 20–23, 2020, Proceedings, Part I* 9. Springer. 2020, pp. 648–662.
- [12] T. Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.
- [13] C. Cardellino et al. "Improvements in information extraction in legal text by active learning". In: *Proceedings of the 28th Annual Conference on Legal Knowledge and Information Systems*. 2015, pp. 21–30.
- [14] S. Chakravarty et al. "Improving the Processing of Question Answer Based Legal Documents." In: *JURIX*. 2019, pp. 13–22.
- [15] S. F. Chen and J. Goodman. "An empirical study of smoothing techniques for language modeling". In: *Computer Speech & Language* 13.4 (1999), pp. 359–394.
- [16] K. Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).
- [17] N. Chomsky. *Syntactic Structures*. The Hague: Mouton and Co., 1957.
- [18] K. W. Church and W. A. Gale. "Probability scoring for spelling correction". In: *Statistics and Computing* 1 (1991), pp. 93–103.
- [19] I. Dagan, F. Pereira, and L. Lee. "Similarity-based estimation of word cooccurrence probabilities". In: *arXiv preprint cmp-lg/9405001* (1994).
- [20] S. Deerwester et al. "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.
- [21] J. Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).
- [22] J. Earley. "An efficient context-free parsing algorithm". In: *Communications of the ACM* 13.2 (1970), pp. 94–102.
- [23] E. Elwany, D. Moore, and G. Oberoi. "Bert goes to law school: Quantifying the competitive advantage of access to large legal corpora in contract understanding". In: *arXiv preprint arXiv:1911.00473* (2019).
- [24] J. Garofalakis, K. Plessas, and A. Plessas. "Automatic identification, extraction and application of textual amendments in Greek legislative texts". In: *Legal Knowledge and Information Systems*. IOS Press, 2016, pp. 187–190.
- [25] I. Glaser, B. Walzl, and F. Matthes. "Named entity recognition, extraction, and linking in german legal contracts". In: *IRIS: Internationales Rechtsinformatik Symposium*. 2018, pp. 325–334.

- [26] A. Graves, A.-r. Mohamed, and G. Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee. 2013, pp. 6645–6649.
- [27] Z. S. Harris. "Distributional structure". In: *Word* 10.2-3 (1954), pp. 146–162.
- [28] M. Hashmi. "A methodology for extracting legal norms from regulatory documents". In: *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*. IEEE. 2015, pp. 41–50.
- [29] J. R. Hurford. "249The evolution of human communication and language". In: *Sociobiology of Communication: an interdisciplinary perspective*. Oxford University Press, 2008-08. ISBN: 9780199216840.
- [30] T. Joachims. "Text categorization with support vector machines: Learning with many relevant features". In: *European conference on machine learning*. Springer. 1998, pp. 137–142.
- [31] P. Johri et al. "Natural Language Processing: History, Evolution, Application, and Future Work". In: *Proceedings of 3rd International Conference on Computing Informatics and Networks*. Ed. by A. Abraham, O. Castillo, and D. Virmani. Singapore: Springer Singapore, 2021, pp. 365–375. ISBN: 978-981-15-9712-1.
- [32] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Upper Saddle River, New Jersey 07458: Pearson Education, Inc., 2009.
- [33] D. Jurafsky and J. H. Martin. "Vector semantics and embeddings". In: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2019), pp. 270–85.
- [34] T. Kasami. "An efficient recognition and syntax-analysis algorithm for context-free languages". In: *Coordinated Science Laboratory Report no. R-257* (1966).
- [35] S. Klein and R. F. Simmons. "A computational approach to grammatical coding of English words". In: *Journal of the ACM (JACM)* 10.3 (1963), pp. 334–347.
- [36] F. Kuhn. "Annotation of a German Legal Decision Corpus for Argumentation Mining". In: *Legal Knowledge and Information Systems*. IOS Press, 2015, pp. 183–184.
- [37] D. v. Kuppevelt and G. Dijck. "Answering legal research questions about dutch case law with network analysis and visualization". In: *Legal Knowledge and Information Systems: JURIX 2017: The Thirtieth Annual Conference*. Vol. 302. IOS Press. 2017, p. 95.
- [38] J. Lafferty, A. McCallum, and F. C. Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In: (2001).
- [39] J. Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36.4 (2020), pp. 1234–1240.

-
- [40] E. Leitner, G. Rehm, and J. Moreno-Schneider. “A dataset of german legal documents for named entity recognition”. In: *arXiv preprint arXiv:2003.13016* (2020).
- [41] H. Li et al. “SAILER: Structure-aware Pre-trained Language Model for Legal Case Retrieval”. In: *arXiv preprint arXiv:2304.11370* (2023).
- [42] R. Lopes, J. Magalhães, and D. Semedo. *Glória – A Generative and Open Large Language Model for Portuguese*. 2024. arXiv: 2402.12969 [cs.CL]. URL: <https://arxiv.org/abs/2402.12969>.
- [43] J. M. Lourenço. *The NOVAtesis L^AT_EX Template User’s Manual*. NOVA University Lisbon. 2021. URL: <https://github.com/joaomlourenco/novathesis/raw/main/template.pdf>.
- [44] M. P. Marcus. “A theory of syntactic recognition for natural language.” PhD thesis. Massachusetts Institute of Technology, 1978.
- [45] T. Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).
- [46] B. Min et al. “Recent advances in natural language processing via large pre-trained language models: A survey”. In: *ACM Computing Surveys* 56.2 (2023), pp. 1–40.
- [47] N. Oesch. “Social Brain Perspectives on the Social and Evolutionary Neuroscience of Human Language”. In: *Brain Sciences* 14.2 (2024). ISSN: 2076-3425. DOI: [10.3390/brainsci14020166](https://doi.org/10.3390/brainsci14020166). URL: <https://www.mdpi.com/2076-3425/14/2/166>.
- [48] J. Rabelo, M.-Y. Kim, and R. Goebel. “Combining similarity and transformer methods for case law entailment”. In: *Proceedings of the seventeenth international conference on artificial intelligence and law*. 2019, pp. 290–296.
- [49] A. Radford et al. “Improving language understanding by generative pre-training”. In: (2018).
- [50] C. Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5485–5551.
- [51] A. Ratnaparkhi. “A maximum entropy model for part-of-speech tagging”. In: *Conference on empirical methods in natural language processing*. 1996.
- [52] N. Reimers and I. Gurevych. “Sentence-bert: Sentence embeddings using siamese bert-networks”. In: *arXiv preprint arXiv:1908.10084* (2019).
- [53] V. Rodríguez-Doncel et al. “Spanish Legislation as Linked Data.” In: *TERECOM@JURIX*. 2018, pp. 135–141.
- [54] M. Saravanan, B. Ravindran, and S. Raman. “Improving legal information retrieval using an ontological framework”. In: *Artificial Intelligence and Law* 17 (2009), pp. 101–124.
- [55] Y. Shao et al. “BERT-PLI: Modeling Paragraph-Level Interactions for Legal Case Retrieval.” In: *IJCAI*. 2020, pp. 3501–3507.

- [56] Z. Shen et al. “Efficient attention: Attention with linear complexities”. In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2021, pp. 3531–3539.
- [57] E. Strubell, A. Ganesh, and A. McCallum. “Energy and policy considerations for deep learning in NLP”. In: *arXiv preprint arXiv:1906.02243* (2019).
- [58] I. Sutskever, O. Vinyals, and Q. V. Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems* 27 (2014).
- [59] A. Svyatkovskiy et al. “Large-scale text processing pipeline with Apache Spark”. In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 2016, pp. 3928–3935.
- [60] P. TRIVEDI. “A question answering system on regulatory documents”. In: *Legal Knowledge and Information Systems: JURIX 2018: The Thirty-first Annual Conference*. Vol. 313. IOS Press. 2018, p. 41.
- [61] A. M. Turing. “Computing machinery and intelligence”. In: *Mind* 59.236 (1950), pp. 433–460.
- [62] P. D. Turney and P. Pantel. “From frequency to meaning: Vector space models of semantics”. In: *Journal of artificial intelligence research* 37 (2010), pp. 141–188.
- [63] A. Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
- [64] B. Wang et al. “Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by A. Rogers, J. Boyd-Graber, and N. Okazaki. Toronto, Canada: Association for Computational Linguistics, 2023-07, pp. 2717–2739. DOI: [10.18653/v1/2023.acl-long.153](https://doi.org/10.18653/v1/2023.acl-long.153). URL: <https://aclanthology.org/2023.acl-long.153>.
- [65] B. Wang et al. “Towards understanding chain-of-thought prompting: An empirical study of what matters”. In: *arXiv preprint arXiv:2212.10001* (2022).
- [66] J. Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 24824–24837.
- [67] T. Winograd. “Understanding natural language”. In: *Cognitive psychology* 3.1 (1972), pp. 1–191.
- [68] W. A. Woods. “Transition network grammars for natural language analysis”. In: *Communications of the ACM* 13.10 (1970), pp. 591–606.





2024 Democratizing Legal Understanding: A Model for Simplifying Judicial Decisions

José Ferrão