ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Artificial Intelligence

COMPARING LARGE LANGUAGE MODELS ON UNFAIR CLAUSE DETECTION IN TERMS OF SERVICES

CANDIDATE

Marco Panarelli

SUPERVISOR

Prof. Andrea Galassi

CO-SUPERVISORS

Prof. Francesca Lagioia

Prof. Marco Lippi

Academic year 2023-2024

To my future self, who will hopefully look at all this with a wry smile on his face.

Contents

Chapter 1

Introduction

In our everyday activity on the Internet we engage with different websites and applications that require us to actively, or passively, accept a contract that governs the relationship between us and the provider of the service. These contracts are necessary since they govern mutual rights and obligations between the two parties, a legitimate contract is legally binding and companies can enforce its terms by refusing the user access to the service. Users should be genuinely interested in knowing what are the company responsibilities and under which situations it can be deemed liable. Although there is a lack of recent studies on user reading behavior, older literature [61][5][55] as well as everyday anecdotal experience tells that most of us click on the ”I have read and agree to the Terms” button without actually reading the contract. This is also ironically known as ’the biggest lie on the Internet”. The reasons of this phenomenon can be summarized by the sentence ”too many documents, too long to read”. Given the average adult reading speed it would take ~ 15 minutes to read a single ToS, if we sum this for each digital service we employ daily it would cost 200 hours a year [57] or even more. This would be a secondary matter if online platform ToS were fair towards the user, which is not the case especially for bigger service providers [58]. Some initiatives like the ToS;DR[73] project try to bridge the gap with the help of the community by manually checking contracts and flagging those clauses that concern

user rights. However, this process needs to be reiterated every time contracts changes and it is hard to scale it on all the websites, apps and platform we daily use. Most of the unfair clause policing fall back to consumer protection organizations and agencies that has to monitor over companies. Starting legal proceedings is an arduous task that includes searching through documents for potentially unfair clauses. Automating this step would enable a lawyer to focus on a much restricted set of clauses, saving resources and time for those administrative bodies that, often, relies on a limited amount of resources. One effort in this direction is the **CLAUDETTE** project, which applies machine learning methods to detect unfair clauses in terms of service and privacy policies. The latest corpus released, and freely available, consists of 142 ToS documents in English, a sufficiently large dataset to train modern neural models in a supervised setting, making it a valuable resource for advancing automated unfair clause detection. The problem with this kind of approach, however, is that it might become obsolete overtime since the language deployed by service providers may change and adapt to new regulations. Under these conditions, the dataset would need to be relabeled according to the new laws and consequently the models should be retrained with the new dataset. Large language models (LLMs), on the other hand, have recently become a mainstream solution for natural language processing and understanding and they can be easily adapted to classification tasks thanks to their **in-context learning** capabilities. Yet, these models often require very large computational resources to run and, in some cases, the best-performing solutions are proprietary and available only under payment (e.g. ChatGPT or Gemini).

This work wants to explore the effectiveness of using LLMs for unfair clause detection in terms of service contracts. Differently from other machine learning approaches such as transformer-based models, LLMs do not inherently need a fine-tuning phase that aligns them with the context and the task they need to undertake. All the necessary guiding that the model need can be convoluted by designing a suitable prompt which describe both the task

and any context information. In this practical case, the legislation regarding the contract's unfairness conditions may change overtime, transformer models would then need to be fine-tuned again possibly discarding all the previous tuning phase to adapt to the dataset containing the newly annotated examples. LLMs instead would only need to incorporate in their prompt the new legislation, thus they could be directly used off-the-shelf. It's worth to mention that the phase of creating a newly annotated dataset would be needed in any case since in every practical setting it is necessary to have a dataset to assess the performance of the tools. Furthermore, this study prioritizes open-source and smaller-scale models, which are particularly relevant in low-resource contexts. These models require significantly fewer computational resources, both environmentally and economically [51], making them more accessible to civil society and enforcement agencies. Moreover, the transparency of open-source models, with their publicly available architectures, allows for more thorough assessment and accountability.

To this end, this work evaluates open-source models through an extensive set of experiments, comparing 8 classifiers against 9 LLMs in 6 different settings, resulting in a total of 62 combinations of model and setting. As a starting point, a comprehensive literature review was conducted to explore current trends and techniques used in addressing similar legal tasks with LLMs. The various experimental settings presented in this study stem from an extensive effort to identify optimal prompts and develop more effective prompting strategies tailored to both the legal domain and the specific task at hand. The resulting techniques, such as the *pipeline* and *multi-prompt* approaches, demonstrate improved performance compared to plain zero-shot and few-shot prompting. However, even the most effective methods, including those using larger models, still fall short of outperforming supervised models in unfair clause detection. Additionally, as a collateral contribution, a lightweight and user-friendly framework was developed to efficiently evaluate LLMs on the 142 ToS dataset, allowing for systematic testing across different settings and

parameter configurations while being easily adaptable to new tasks.

Chapter 2 focuses, in the first part (2.1), on defining the theoretical foundations of large language models (LLMs), including their formal definition, implementation, and their apparent suitability for a wide range of language processing tasks. It follows a brief part on the most commonly used datasets and benchmarks in the legal domain (2.2), a section on the use of AI in the legal domain (2.3), the description of the CLAUDETTE project (2.4) and a collection of representative work (2.5, 2.6). Chapter 3 details the dataset (3.1) and the in-depth analysis (3.2) of the methodologies employed for experiments on both fine-tuned models and LLMs, followed by the experimental setting (3.3). Chapter 4 discuss the results, detailing the performances of the different approaches 4.2 and the error analysis 4.2.2 on the best performing models. Finally, Chapter 5 concludes the study (5).

Chapter 2

Background

Just as the foundation of a structure determines its stability, understanding and evaluating the influence of AI in any domain, including the legal field, requires a clear definition of its fundamental components. This involves delineating their definitions, formalism, and implementations. Given the focus of this study on the role of Large Language Models in the legal field, it is appropriate to begin by describing what a language model is.

2.1 Language Model

It is sound to begin by providing a formal definition of *language*, drawing initially from the foundational works of Chomsky [9] and subsequently exploring its evolution through modern paradigms. With **language**, we mean a set (finite or infinite) of sentences, each of finite length, all constructed from a finite alphabet of symbols. Following this, we can define as the **grammar** of a language a device that produces all and only those strings[1] that are sentences of the language [9]. This ”device” proves useful in various contexts; from a linguistic perspective, it can aid in better understanding the language,

[1] A string is simply anything obtained by concatenating the symbols of the alphabet (e.g. a word).

such as by supporting semantic analysis. Alternatively, when properly designed, it can be applied to tasks requiring language-related functionalities. From a strictly formal point of view, the concept of a **language model** can be interchangeable with the one of a grammar, however it is standard to identify the former as a statistical model of language, that is, a model that defines a probability distribution over sequences of tokens in a language. These tokens can be characterized as simple words, characters or even part of words. In their most common usage, language models assign probabilities to sequences of words, this means that it is possible to assign probabilities either to a word given some previous context or to an entire sentence, these two concepts are linked by the chain rule of probability.

$$\begin{aligned} P(w_{1:n}) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1:2}) \cdots P(w_n \mid w_{1:n-1}) \\ &= \prod_{k=1}^{n} P(w_k \mid w_{1:k-1}) \end{aligned} \tag{2.1}$$

A sequence of n words is represented as $w_{1:n}$. Knowing the probability of a sentence[2] or what is the next most probable word results very useful in different tasks such as **speech recognition** or **spelling correction**. Such a probabilistic model can be helpful in dealing with homophone confusion or grammatical errors by predicting, for example, that "Your going to loose your keys" is a much less probable english sentence than "You're going to lose your keys". We will start by describing the earliest and most successful type of language model, the **n-gram**.

2.1.1 n-grams

The term n-gram can identify both the probabilistic model and a sequence of n words, by relying on the previous definition (2.1) the probability of the next

[2]With the probability of sentence we refer to the probability of the sentence to appear in a language.

word provided by a **tri-gram** will be:

$$P(w_t) = P(w_{t-2})P(w_{t-1} \mid w_{t-2}) = \prod_{k=1}^{3} P(w_k \mid w_{1:k-1}) \tag{2.2}$$

Generally speaking, when using an n-gram model we are making the following approximation:

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k \mid w_{1:k-1}) \tag{2.3}$$

This implies that we are making a **Markov** assumption, where the probability of a word depends solely on a portion of its preceding context. The formalization of the n-gram model traces back to Markov himself [54], that used bi-gram and tri-gram to predict the likelihood of the next letter being a vowel or a consonant in Pushkin's *Eugene Onegin*. Given a corpus of text, an n-gram model can be trained simply by counting and normalizing the n-grams. This approach to probability estimation is known as **Maximum Likelihood Estimation**. Various smoothing techniques are available to address scenarios where a word in the test set appears in an unfamiliar context despite being in the training vocabulary. Additionally, words absent from the training set are typically managed by converting the problem into a closed-vocabulary one. With these precautions, n-gram models have been successfully applied to systems for **speech recognition** [36] [2], **plagiarism detection** [3], and **machine translation** [53]. However, it is evident from their training method that the number of parameters grows exponentially with the order of the n-gram, which is why no models extend beyond 5-grams. Moreover, n-grams inherently lack the ability to represent the ”similarity” between words, a feature that would facilitate more effective generalization.

2.1.2 Neural Language Model

To address the limitations of n-gram models, Bengio et al. [4] introduced the use of neural networks with distributed word representations. Their approach incorporated many foundational elements later utilized by modern methods, although the specific implementations have evolved significantly. Notably, contemporary architectures, such as Transformers, rely on self-attention mechanisms rather than the feedforward neural networks initially proposed. A word can be represented as a point in an m-dimensional vector space, R^m, commonly referred to as a *feature vector* or, more typically, an **embedding**. These representations are learned based on the distribution of words in a text corpus, aligning with the distributional hypothesis, which posits that words with similar semantic and/or syntactic roles tend to occur in similar contexts. Consequently, similar words are located closer in the vector space due to their similar embeddings. Several methods have been developed to learn these representations. For example, **tf-idf**, which relies on the frequency of terms relative to a collection of documents, serves as a baseline approach, although it produces sparse, high-dimensional vectors. In contrast, algorithms like **Word2Vec** and **GloVe** generate dense vectors with dimensionalities significantly smaller than the vocabulary size ($m << |V|$). Modern techniques, such as those used in Transformer-based models, directly learn contextual embeddings during training, without relying on predefined word representations. The simplest kind of neural network is a **feedforward network** where multiple layer are connected without cycles, each layer is composed by different units that performs a weighted sum of the input, add a bias term and then apply a non-linear function. The results of this operation are called **activations** which will be used as inputs to the subsequent layer. A 3-layer network is illustrated in figure 2.1, it is a fully-connected network since each unit in a layer receives inputs from all units in the previous layer. Using matrix notation, the activations of the first

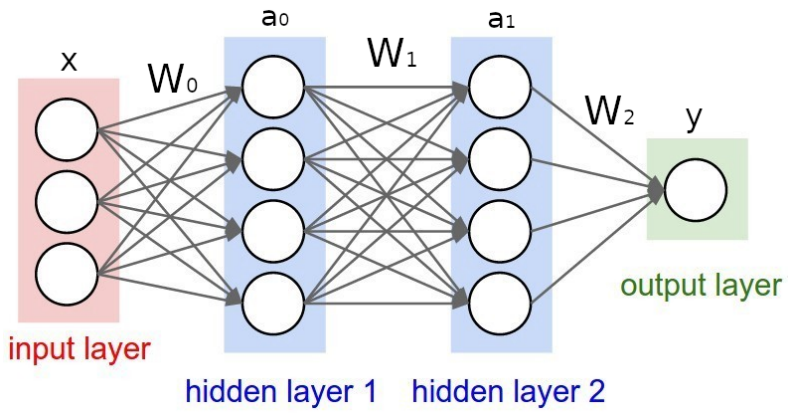

Figure 2.1: The structure of a general multi-layer neural network.

layer (a_0) can be expressed as:

$$a_0 = f(W_0 \cdot x + b) \tag{2.4}$$

Here W_0 is the weight matrix, the **bias** term b is used to avoid 0-valued activations and the **activation function** f is a non-linear function which is applied element-wise to the input vector x. This family of functions are introduced to enable to learn complex, non-linear relationships between features; without them, the network would only learn a composition of linear functions. Since we are dealing with words in a vector space, the input word is a vector with dimension m, consequently the weight matrix W_0 has dimensions $h_0 \times m$ where h_0 is the number of units in the first hidden layer. The same extends to the subsequent layers: W_1 has dimension $h_1 \times h_0$ and W_2 has dimensions $n \times h_1$), where n is the number of output units. The output dimension n varies depending on the task being performed, such as *part-of-speech tagging* or *next word prediction*. The output vector contains real values, also called **logits**, that needs to be transformed into probabilities, this is achieved using the **softmax** function:

$$softmax(s_i) = \frac{exp(s_i)}{\sum_{j=1}^{V} exp(s_j)}, i \in [1..V] \tag{2.5}$$

If the task involves predicting the next word given the previous context, $softmax(s)$ will produce a probability distribution over all the words in a vocabulary with length $|V|$. In this case, the input would consists of the concatenation[3] of the words embeddings.

To train this kind of model, and also more modern architectures, there are three fundamental steps that are necessary. First, a **forward pass** is performed to compute the output of the model, which in this case is a probability distribution. A **loss function** is need to model the distance between this output and the true one expected for the provided input. Secondly, the gradient for all the model's parameters with respect to the loss function must be computed. This is done by an algorithm called **backpropagation**. Lastly, each parameter must be updated according to its gradient with techniques such as **stochastic gradient descent** or **Adam**.

Loss Function

When training a machine learning model, we typically have access to a set of training examples X and their corresponding labels Y. Training the model can be formulated as solving an optimization problem of the form:

$$\theta^* = argmin_\theta L(f_\theta(X), Y), \tag{2.6}$$

where $f_\theta(X)$ represents the model's output given the input data, and L is a loss function that evaluates the discrepancy between the predicted output and the true labels. The loss function L should provide meaningful feedback about the model's performance with respect to the learning objective, whether it involves predicting a discrete set of labels or approximating a probability distribution. Ideally, a loss function is both **convex**, to ensure effective optimization, and **differentiable**, to allow gradient-based optimization. Although neural networks introduce non-linearities that violate these requirements, the

[3] A general pooling operation such as *max* or *avg* can also be applied.

properties of a loss function can still be studied independently of the underlying model. A widely used loss function is the **cross-entropy loss**, which measures the difference between a true probability distribution P (that generated the data Y) and an approximating distribution Q (produced by the model). The continuous form of cross-entropy is:

$$H(P, Q) = -\mathbb{E}_{x \sim P}[\log Q(x)] \tag{2.7}$$

In practical scenarios we are dealing with a finite amount of data ($|X| = |Y| = n$), therefore we must rely on the discrete formulation of the cross-entropy. The model outputs a vector of scores (or *logits*) which is converted into a probability distribution by the *softmax* function. For each input vector x, the true label vector y is typically one-hot encoded, meaning $y_k = 1$ for the correct class k, and $y_j = 0$ for all other classes ($j \neq k$). The discrete cross-entropy loss can then be expressed as:

$$L_{CE}(x, y) = -\sum_{i}^{|V|} y_i \log\left(x_i\right) \tag{2.8}$$

Given the one-hot nature of y, this simplifies to:

$$L_{CE}(x, y) = -\log x_k, \tag{2.9}$$

where k is the index of the correct class. Substituting the $softmax$ function (Eq. 2.5), the loss becomes:

$$L_{CE}(x, y) = -\log \frac{exp(s_k)}{\sum_{j=1}^{|V|} exp(s_j)} \tag{2.10}$$

This formulation shows that the cross-entropy loss quantifies the difference between the true label distribution and the predicted probabilities, driving the model to assign higher probabilities to the correct classes.

Backpropagation

To train a neural network, the primary goal is to minimize the loss function by updating the model's parameters. This requires calculating the derivative of the loss with respect to each parameter, traversing the entire network from the final layer back to the initial layers. These derivatives, or gradients, are then used to adjust the parameters to reduce the loss. The algorithm used for this purpose is called **error backpropagation** [65], which leverages the **chain rule** of calculus and the notion of **computation graph** to compute these gradients efficiently. The backpropagation process begins by performing a forward pass through the network, where the input propagates through each layer, and intermediate results are computed and stored for the subsequent gradient computations. During the backward pass, the derivative of the loss with respect to the model's output is computed first. This initial gradient is then propagated back through the network, layer by layer. At each layer, the gradient of the loss with respect to the layer's parameters is calculated using the chain rule which allows the computation of gradients for earlier layers by combining the gradient from the subsequent layer with the local gradient at the current layer. This process systematically computes gradients for all parameters, ensuring that the contributions of all layers to the loss are taken into account. Once the gradients are obtained, they are used to update the parameters with optimization algorithms.

2.1.3 Training

Training a machine learning model involves finding the optimal weights, denoted as θ^*, by minimizing a loss function. This optimization problem can be solved using **gradient descent**, which iteratively updates the weights by moving them in the opposite direction of the gradient, scaled by a small step size. If the loss function is convex, gradient descent is guaranteed to converge to the global minimum regardless of the starting point in the loss landscape.

For multi-layer neural networks, however, the loss function is typically non-convex due to the non-linear activation functions. This non-convexity introduces challenges, such as saddle points, flat regions, and local minima, which can slow down or hinder convergence. Nevertheless, gradient descent methods are commonly used in deep learning because they often lead to satisfactory solutions with very low loss values in practice. This success is attributed to the properties of neural network loss landscapes, which tend to have large regions of near-optimal solutions rather than isolated local minima [44]. Second-order optimization methods are theoretically faster than first-order methods like gradient descent. However, these methods are challenging to scale to large networks due to the computational cost of inverting the Hessian matrix (containing the second-order partial derivatives), which has a computational complexity of $O(k^3)$, where k is the number of parameters, and requires memory in the order of the square of k. For this reason, first-order methods remain the standard approach for training deep models. Gradient descent updates the model parameters according to the following rule:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \mu \nabla_{\theta} L, \tag{2.11}$$

where $\mu > 0$ is the **learning rate**, controlling the size of the parameter update, and $\nabla_{\theta} L$ is the gradient of the loss function L with respect to the parameters $\boldsymbol{\theta}$. Computing this gradient over the entire training set can be computationally expensive, especially during the initial stages of training when the model is far from optimal. To address this inefficiency, **stochastic gradient descent (SGD)** approximates the gradient by computing it over a randomly selected subset of training examples. In its most extreme form, called **online SGD**, the gradient is computed using only a single example at a time. Algorithm 2.1.3 shows the pseudocode for the online gradient descent.

A more common variant is **mini-batch SGD**, where the training set is divided into smaller subsets, or **batches**, typically of size that is a power of 2

Algorithm 1 Line 7 performs the forward pass given the input $x^{(i)}$, line 9 corresponds to the backward pass while line 10 is the parameter update. This is repeated on all the input data and for a predefined number of epochs E

1: $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ ▷ training data
2: Parameters $\boldsymbol{\theta}$ ▷ Model parameters
3: Loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$
4: **for** `e = 1,...,E` **do**
5: shuffle $\mathcal{D}$
6: **for** `i = 1,...,N` **do**
7: Compute prediction $\hat{\mathbf{y}}^{(i)} = f_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}; \boldsymbol{\theta})$ ▷ $f_{\boldsymbol{\theta}}$ is the actual model
8: Evaluate loss function $\mathcal{L} = \mathcal{L}(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$ ▷ This is optional
9: Compute gradient $\mathbf{g} = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$
10: Update parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \mu \mathbf{g}$
11: **end for**
12: **end for**

(e.g., 32, 64, 128). Mini-batching leverages the parallel processing power of modern GPU architectures, significantly improving computational efficiency. Moreover, the inherent noise introduced by using smaller batches can act as a form of regularization, preventing the model from overfitting to the training data [1, 59].

The learning rate μ is a critical hyperparameter in the optimization process which significantly impacts both the model's convergence speed and its final performance. A learning rate that is too large can cause the optimization process to diverge, while one that is too small can lead to slow convergence. To address this, adaptive learning rate algorithms, such as **Adam** [39], are commonly used. Adam adjusts the learning rate for each parameter individually based on estimates of the first and second moments of the gradients. This approach allows the optimization process to adaptively scale the learning rate during training, leading to faster convergence and improved performance in many cases.

2.1.4 Implementations

brief description of RNN and LSTM, self-supervision, autoregressive generation (no specific techniques), what task can they solve

The **transformer** architecture, first introduced by *Vaswani et al.* [77], has become the standard framework for building large language models. While it has undergone various modifications and improvements, modern implementations of transformers retain the key concepts from the original design. Prior to the introduction of transformers, models such as **recurrent neural networks (RNNs)** [18] and **long short-term memory networks (LSTMs)** [30] were widely used for natural language processing tasks. These recurrent architectures were the first to explicitly incorporate the temporal and sequential nature of language into their designs, enabling them to process input data step by step over time. However, they faced significant challenges that the transformer architecture partially overcame. One of the primary shortcomings of recurrent models is their difficulty in capturing long-term dependencies within a sequence. This limitation arises not only from the vanishing gradient problem, which hampers their ability to propagate information over long time steps during training, but also from their inherently sequential processing. Because each step t depends on the computation performed at step $t-1$, these models lack the ability to parallelize computations effectively, leading to inefficiencies, especially when working with long sequences. The transformer architecture addresses these issues by completely eliminating recurrence. It introduces an **encoder-decoder architecture** designed to process sequences in parallel while efficiently capturing dependencies between input elements by using the **attention mechanism** as the core tool to model these dependencies. The encoder processes the input sequence and generates a set of context-aware representations, while the decoder uses these representations to produce the output sequence, one token at a time. Attention allows the model to weigh the importance of all input tokens relative to one another, regardless of their position in

the sequence, enabling it to capture both short and long-term dependencies effectively. The final output of the transformer is a probability distribution over possible tokens, computed through a **softmax** (Eq. 2.5) function applied to the model's logits. This probability distribution allows the model to generate predictions for tasks such as translation, text generation, or classification. This design facilitates parallel processing, as attention computations can be performed simultaneously across all tokens, making the transformer architecture highly efficient and scalable for modern hardware. Since the transformer is a complex architecture with several details and components, only the main component of the **multi-head attention** will be described along with how these models are trained.

Attention

The attention mechanism has a biological background deriving from how the human brain focus its processing capabilities on specific environment cues to optimize its resources and elaborate only those things that are important. For what concern the language, it would be useful to know how much attention each word in a sequence should take with respect to other words in the same sentence. A classic example is the following:

1. The **student** didn't submit the **assignment** because `it` was too complex.

2. The **student** didn't submit the **assignment** because `it` was too lazy.

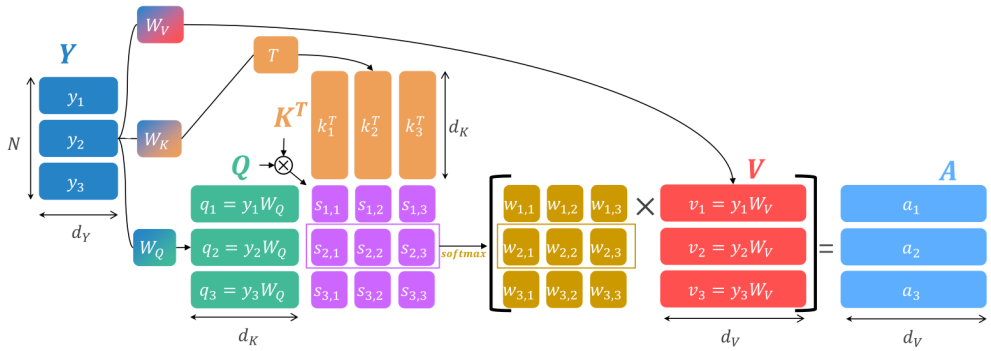

Figure 2.2: Y is the input sequence of N embeddings, each of dimension d_Y. The $softmax$ function is applied row-wise on the scores deriving from the $W_Q \otimes W_K^T$ dot-product.

The meaning of the word `it` could be derived by assessing how much attention it should make to the words "student" and "assignment". Concerning deep learning architecture, we are working with words represented as embeddings with dimension a certain dimension d. Given embedding representation of a sentence x, the whole attention function can be summarized as follows:

$$A = softmax(\frac{QK^T}{\sqrt{d}})V \tag{2.12}$$

where

$$Q = xW_Q; K = xW_K; V = xW_V \tag{2.13}$$

The matrices Q, K and V are different representation of the input embeddings that corresponds to **queries**, **keys** and **values**. Queries are the subject of the attention analysis, these are compared to the *keys* via a dot-product and scaled down by $\sqrt{d}$ to avoid numerical issues when dealing with embeddings having large dimensionalities (e.g. 512 or 1024). The $softmax$ function is applied row-wise to produce the normalized attention weights which are then used to weight the actual *value* stored in V. Since we are using x for all the three matrices, we are performing **self-attention**, the same operation is called **cross-attention** if x is used only as *query* and another element is employed as *key*

and *value*. Finally, The **multi-head attention** primitive arise by computing N attention heads and apply another linear projection W_O to their concatenation.

$$head_i = softmax(\frac{Q_i K_i^T}{\sqrt{d}})V_i \tag{2.14}$$

$$A = (head_1 \bigoplus head_2 \bigoplus ... \bigoplus head_N) \tag{2.15}$$

This operation make it possible to learn N different attention representations. Figure 2.2 depict a single self-attention head, since we want to attend only to previous word in a sentence, it is common to mask the scores associated to future words ($s_{i,j} = -\infty$ where $j > i$).

Training Method

The training process begins with the collection of a comprehensive dataset, often referred to as a corpus. This dataset serves as the foundation for the model's knowledge and it typically comprises text from diverse sources, such as books, articles, websites, and other publicly available content. Commonly used corpora include datasets like Wikipedia, Common Crawl, and OpenWebText, among others. Once the corpus is collected, it undergoes a pre-processing stage to standardize and clean the text for use as input to the transformer model. A key aspect of this step is **tokenization**, which involves breaking the text into smaller units called tokens. These tokens can represent words, sub-words, or even individual characters, depending on the tokenization algorithm used. The resulting tokens are then mapped to embeddings using an embedding matrix, which serves as a lookup table associating each token with a fixed-dimensional vector representation. LLMs are trained using **self-supervised learning**, a method that does not requires any labeling step since the natural sequence of words embody the gold label itself. Two common approaches are **causal language modeling** where the model has to predict the next word in a sentence and **mask language modeling** where the model is tasked to fill missing words in a passage.

In this first stage, known as **pre-training**, the model is exposed to the entire training corpus and is trained to minimize the cross-entropy loss between its predictions and the actual text. Pre-training results in a model that possesses a broad understanding of language but is not yet specialized for specific tasks or aligned with human preferences, but it already has the foundational capabilities necessary for a wide range of downstream applications.

After pre-training, the model undergoes an **alignment phase** to ensure its outputs are useful, accurate, and aligned with human intentions. This phase typically involves fine-tuning the model on smaller, task-specific datasets, which may include human-annotated examples. Reinforcement learning from human feedback (**RLHF**)[38] is a common technique used during alignment. In RLHF, human evaluators rank model outputs based on quality, and this feedback guides the model toward generating more desirable responses. Alignment should ensures that the model's behavior is safe, ethical, and effective for real-world applications, addressing issues such as harmful outputs, biases, or unintended behavior observed during pre-training.

Up to now, the main ingredients that are presents behind a LLM have been described. Many of them may be found also in other areas besides NLP, since most of these concepts are at the foundations of how modern AI techniques are developed. At this point, it is worth to make a brief analysis that better describe the transition to modern LLMs from per-trained language models (PLM).

2.1.5 What makes a LLM

Earlier methods relied on the *pre-training* and *fine-tuning* paradigm which gave light to popular models such as BERT[15] and all its family of related models. As the name suggests, first the model is pre-trained on large-scale unlabeled corpora and then fine-tuned on the specific downstream task. This second step usually relies on an additional smaller dataset that is specifically related to the task that the model has to specialize on. The general idea is that

the pre-train phase let the model learn context-aware representations that can be later optimized according to the downstream task. Upon this paradigm, a wealth of models have been defined, such as GPT-2, BART and RoBERTa[63, 43, 47], that try to explore different architectures and pre-training strategies. Starting from this baseline, several studies found that scaling the model size and the data size of PLMs led to improved performances on downstream tasks. Just to give a comparison, GPT-3 has 175 billion parameters and was trained on 300 billion tokens, making it several order of magnitude larger than GPT-2, which has 1.5 billion parameters and 40 billion tokens. This scaling results in bigger models that are not only able to solve more complex tasks but also shows a range of new abilities, called *emergent*, that were not present in smaller PLMs. Earlier models shows good performances on tasks like sequence classification, named entity recognition and question answering, however they are limited in reasoning capabilities and on generating coherent content by following instructions. On the other hand, LLMs not only performs these tasks better but they can also perform more advanced ones such as coherent text generation, multi-turn conversations, reasoning, code generation, and multilingual translation, often without requiring fine-tuning

Although there is not a formal definition, the research community started to use the term ”*large language models*” after the first articles on these bigger models where published. This size difference rises several challenges on how these models are trained, if models up to few billions parameters can be managed without too much efforts even on personal computers shifting to bigger sizes requires a rigorous engineering approach. Matters such as parallel processing efficient distributed training, memory optimization, and hardware acceleration become critical. Training large-scale models like GPT-3 demands massive computational resources, typically relying on clusters of GPUs or TPUs to handle the enormous parameter space and data throughput. TPUs, in particular, have been extensively used for training large language models due to their specialized architecture, which is optimized for matrix operations and

large-scale parallelism. Additionally, techniques such as model parallelism and pipeline parallelism are necessary to distribute the workload across multiple devices, ensuring that both computation and memory usage are balanced.

Both PLM and LLM relies on the transformer architecture which is also referred to as **encoder-decoder** (ED) but most recent LLMs use only its decoder stack (DO). In the first case, the encoder is fed with the input sequence and use self-attention to produce a vector of the same length as the input of high-dimensional contextual-representations. The decoder instead use a combination of self-attention and masked cross-attention to predict the output sequence with the conditioning of the representations produced by the encoder. In decoder-only architecture instead all tokens are processed equivalently by conditioning only on previous tokens thanks to a causal masking pattern. These models have a simpler architecture and are more naturally suited for a next-world prediction objective while PLMs are usually trained with a masked language modeling approach. As an example, BART[43] is an ED transformers pre-trained by corrupting the input tokens in different ways such as masking, deletion and infilling. This difference in the training objective and in the architecture is reflected also on how the models encode the linguistic information. BERT-like models[35], for instance, encodes surface features at the bottom, syntactic features in the middle, and semantic features at the top while LLMs like tends to gather lexical semantics in shallower layers[4] to later aggregate these information in higher layers to make the predictions[78, 48].

Delving briefly into the previously mentioned emergent abilities, two of the most representative are *in-context learning* (ICL) and *instruction tuning*. With **ICL** we refer to the ability of the model to adapt to new tasks or contexts by processing prompts provided by users without any updates to its underlying parameters. These prompts may include instructions, examples, or contextual information that guide the model's behavior for a specific query

[4]"Shallow" or "bottom" layers refer to those closer to the input, while "top" or "high" layers are closer to the output

or task. Based on the type of prompt fed to the model, techniques such as zero-shot or few-shot prompting can be derived. In the first case, the model is asked to perform a task without any prior examples in the prompt relying purely on its pre-trained knowledge. Few-shot prompting, instead, make use of few examples that acts as demonstrations, helping the LLM to learn the pattern between input and output. Regarding instruction tuning, the model is fine-tuned on a dataset containing a diverse set of tasks formatted as natural language prompts. This approach enhances the model's ability to follow explicit instructions and generalize to unseen tasks, improving its zero-shot and few-shot learning capabilities. By training on a broad range of prompts and responses, instruction-tuned models learn not only task-specific patterns but also general strategies for understanding and executing various instructions. This method has been shown to significantly improve performance across multiple benchmarks, sometimes enabling smaller models to match or even surpass the performance of much larger models trained without instruction tuning.

2.2 Data

Understanding the type of data used in the AI-legal domain requires distinguishing between data employed for pre-training models and data used to evaluate their performance. The same source of data can be used to build both a corpus used for training and a benchmark dataset later submitted to the trained model to evaluate its functionality. Moreover, when it comes to LLMs, it may be hard to determine the extent and the quality of the legal knowledge that trickled into the model as pretraining corpora are often proprietary and not publicly disclosed. Nonetheless, several documented datasets and benchmarks recur in academic literature and merit a brief description. This section distinguishes between corpora used for model training and datasets typically utilized for specific evaluation tasks, although as stated before these two aspects may overlap in some cases.

2.2.1 Corpora

The legal domain inherently generates extensive documentation, with entities such as the European Court of Justice and the Supreme Court of the United States publishing vast amounts of information through dedicated databases [19][68]. Consequently, it is feasible to compile sufficiently large corpora for training or fine-tuning AI models.

- **Eurlex57K** [19] contains 57,000 EU legislative documents extracted from the EUR-Lex database [19].
- **ECHR** [62] includes approximately 11,500 cases from the European Court of Human Rights.
- The **Caselaw Access Project** [42] provides over 160,000 U.S. court cases, available for research purposes.
- **Pile of Law**[29] compiles data from 35 diverse sources to build an heterogeneous[5] open-source dataset with legal and administrative data from different jurisdictions, totaling $\sim$ 10M documents.
- **OPP-115** [79] instead focus on website privacy policies with 115 documents, while **CLAUDETTE** contains 142 terms of service of some of the major players across 10 different market sectors. Being the corpus used in this work, **CLAUDETTE** will be further analyzed in chapter 3.1.

2.2.2 Benchmarks

Benchmarks are essential for evaluating AI model performance as they provide standardized tasks and metrics, enabling consistent comparison and tracking of progress across models.

[5] It encompasses: legal analyses, court opinions and filings, government agency publications, contracts, statutes, regulations, casebooks, and more.

- **LexGLUE** (Legal General Language Understanding Evaluation) [7] is a benchmark comprising seven legal-domain-specific tasks designed to evaluate models on legal text comprehension. It spans multiple jurisdictions (EU and U.S. law) and application domains (e.g., laws, contracts), with the aim of advancing foundation models capable of addressing diverse legal natural language understanding tasks.

- **LegalBench** [27] features 162 tasks covering six categories of legal reasoning[6] drawing heavily from the American body of law. It supports various task formats, including binary classification, multiple-choice questions, open-ended generation, and multi-class/multi-label classification.

2.3 Why AI in the legal field ?

The judiciary system plays a pivotal role in maintaining justice and societal order in modern nations. However, it faces numerous challenges that hinder its ability to achieve a fair and efficient system. Several critical problems affect nearly every justice system, regardless of whether it follows Common or Civil law traditions or the specific legal frameworks employed by individual states.

One of the most recurring and significant issues can be summarized by the maxim: "*too many cases, too few people*". Traditional judiciary systems are overwhelmed by an ever-increasing volume of cases, resulting in backlogs with delays that can span years or even decades. This backlog arises primarily from a shortage of legal professionals, insufficient resources, and the inherently repetitive nature of certain judicial tasks, such as document review, evidence analysis, and drafting legal opinions. While these challenges are often framed as a matter of **efficiency**, they also have broader implications for the fairness of the system and represent a substantial economic burden. In

[6] Issue-spotting, rule-recall, rule-application, rule-conclusion, interpretation and rhetorical-understanding

addressing these inefficiencies, legal AI tools offer significant potential, particularly in automating repetitive and time-intensive tasks. For instance, the transcription of judicial proceedings can be effectively performed by **speech-to-text systems**, which, in their current state, can transcribe entire hearings with high accuracy and reliable speaker diarization [52]. Another recurring task in legal practice is **legal case retrieval**, which involves searching large databases for relevant cases based on specific queries, legal principles, statutes or keywords. Although modern systems are not yet optimal [20], this remains an active area of research, exemplified by annual competitions such as COLIEE [26].

Looking further ahead, systems capable of automating the decision-making process in legal cases could significantly alleviate judges' workloads. However, this is a highly sensitive area that necessitates an interdisciplinary approach, as such tools could have widespread and potentially detrimental societal impacts. Nevertheless, **legal judgment prediction** is an active research topic [21][11], with increasing focus on developing approaches that produce interpretable and consistent outcomes.

Finally, engaging with legal materials is often challenging for individuals without a legal background. In this context, AI applications have the potential to democratize access to legal information. For example, tools for **legal question answering** (e.g. LLeQA and Lawyer LLaMA [49, 31]) enable individuals to address legal queries independently and with an accessible jargon. Similarly, **automatic summarization** of legal documents can benefit both legal professionals and the general public by distilling lengthy and intricate legal texts into concise, digestible summaries [34].

2.3.1 EU Law on unfair terms in contracts

For what concern unfair contractual clauses, European consumers are protected by the directive 93/13[10] that establishes a legal framework aimed at

protecting consumers from unfair contractual provisions in agreements concluded between businesses and consumers. This directive is particularly relevant in contexts where standard contractual terms are pre-drafted by businesses and offered on a take-it-or-leave-it basis, leaving consumers with no opportunity to negotiate. The primary objective of the directive is to ensure that such terms are fair, transparent, and do not create a significant imbalance between the rights and obligations of the contracting parties. The directive applies broadly to consumer contracts across various sectors, including goods, services, and digital transactions. The core principle underpinning the directive is the **fairness test**, which establishes that a contractual term is deemed unfair if it causes a substantial imbalance to the detriment of the consumer, contrary to the principle of good faith. Furthermore, contractual terms must be drafted in plain and intelligible language, ensuring that consumers can fully understand their rights and obligations. Any ambiguities in the contract are to be interpreted in favor of the consumer. The directive also provides a non-exhaustive list of unfair terms (Annex 1), which includes provisions that grant the business excessive discretion over contract performance, such as the unilateral right to modify terms without valid justification. If a contractual clause is found to be unfair, it is deemed non-binding on the consumer, while the remainder of the contract remains in force, provided that its validity is not compromised. A consumer can take a dispute to court to challenge the unfairness of a contract, this is referred as **individual** control of fairness. **Abstract** control, on the other hand, is performed by consumer protection organizations which have the competence to initiate judicial or administrative proceedings. In this case, each member states may have different implementations.

2.4 CLAUDETTE

CLAUDETTE[45] (Automated CLAUse DETeCTEr) is an international research project that attempts to empower consumers and nongovernmental organizations with a machine learning tool to automate unfair clause detection in terms of service and privacy policies. It relies on supervised learning methods and on a source corpus of 142 ToS documents manually annotated by legal experts, grounding on the EU Directive 93/13 and the GDPR. More details on the whole corpus are given in section 3.1. The Unfair Contract Terms Directive sets the minimum standard of consumer protection with regard to contract terms and it deems a clause as unfair if "contrary to the requirement of good faith, it causes a significant imbalance in the parties rights and obligations arising under the contract, to the detriment of the consumer". In its first iteration [45] the CLAUDETTE system was tested with different machine learning techniques over a dataset of 50 documents for a total of ~ 12000 sentences. The task is a multi-label classification over 8 categories of unfairness, meaning each sentence can be assigned to 0 or more categories, however the first experiments were formulated as a binary classification task. The ML techniques employed included Support Vector Machines (SVM), Convolutional Neural Networks (CNN) and Long-Short Term Memory Networks (LSTM), the sentence was represented in 3 different modalities, namely the *bag-of-words* model, *parse tree* and *word embeddings*. The best approach was the combination of 8 different SVM-HMMs[7] (one per category) using n-gram features with a macro-F1 of 0.805 on the binary task. The project continued [64] by exploring the use of legal rationales to both improve the classification accuracy and to offer useful natural language explanation to the classifier output by using memory-augmented neural networks[67]. Since the project is strictly related to the European Union area, a multi-lingual approach was considered by experimenting with different techniques to project annotations

[7] Support vector machine with Hidden Markov models [76]

between documents written in different languages[22]. Strictly related to this, the problem of detecting unfair clauses in ToS was extended across multiple languages, comparing several strategies to extend a classifier from English to other lower-resources languages[23]. In this setting, directly translating the test set directly to English rather than training on a novel corpus for the source language resulted in the best performances (0.66 vs 0.62 F1). On top of this, working on these matters gave valuable insights and supplementary details about the relationship between users and digital platforms [40][16][33]. Related to the CLAUDETTE project, the corpus of 100 ToS[16] employed in several works, is also included in the **LexGLUE** benchmark[7].

2.5 LLMs for Legal Tasks

This section provides an overview of the application of large language models (LLMs) across various legal tasks first focusing on surveys which offers a bird's-eye view of the techniques and systems employed and then delving deeper into notable studies that utilize prompting techniques for legal applications.

Lai et al. [41] conducted a literature review on the use of LLMs in the judicial field, including an analysis of their role in assisting judges. The study identifies several fine-tuned legal LLMs, primarily based on the Chinese legal system, and examines AI-assisted decision-support systems implemented in courts. The authors highlight how such systems can enhance judicial efficiency by expediting case processing, improving the accuracy and consistency of decisions, and providing legal advice. However, these systems face significant challenges, including limited processing capabilities for long texts, inadequate adaptability to individual cases, and privacy and ethical concerns. The study concludes with recommendations for future research, including improving data quality and privacy protection, enhancing long-text processing capabilities, and establishing ethical and regulatory frameworks for legal AI

applications.

Padiu et al. [41] provide a comprehensive survey of legal LLMs, analyzing their advancements, applications across different legal systems, and practical limitations. Their findings indicate that standalone LLMs, such as GPT-3 and LawyerLLaMA, struggle with complex legal reasoning tasks due to hallucination issues and insufficient legal knowledge grounding. However, approaches that integrate external legal knowledge, vector databases, and retrieval systems demonstrate superior performance, even surpassing human experts on certain multiple-choice legal tasks. Additionally, the study highlights the difficulties AI models face in adapting to multicultural and multilingual legal environments, as well as the challenge of keeping pace with the evolving nature of legal systems.

Several studies have explored the use of LLMs for different legal tasks, though relatively few have focused specifically on contract classification. Tang et al. [69] evaluated multiple LLMs on two datasets containing privacy policies from websites (OPP-115 [79]) and mobile applications (PPGDPR [46]). They designed a zero-shot prompting framework comprising three sections: (1) background context, (2) a list of privacy-related categories along with their description, and (3) the task description. Their results indicate that GPT-4 achieved the highest performance, outperforming all other approaches.

Hakimi Parizi et al. [28] experimented with various prompting strategies for legal document classification, including zero-shot and few-shot learning, chain-of-thought (CoT) prompting, activation fine-tuning, and prompt ensembling. They evaluated these techniques on the ECHR (binary classification) and SCOTUS (multi-class classification) datasets, using the OPT language model family[8]. Their results indicate that activation fine-tuning performed best, while CoT prompting was ineffective, likely because the models were not instruction-tuned.

Trautman [74] employed a chaining approach to break down complex legal

[8] OPT-6B and OPT-175B

classification tasks into smaller subtasks for evaluation on ECHR and SCOTUS datasets. In this framework, the output of each prompt serves as input for the next, beginning with generating a summary of the legal case. A few-shot prompt is then constructed by retrieving the eight most semantically similar cases to the input summary. Finally, the initial summary, combined with the eight retrieved cases, is used for label prediction. This chained prompting strategy significantly improved performance over zero-shot prompting and enabled smaller models to outperform larger ones in terms of micro-F1 score.

Yu et al. [80] explored prompting techniques to guide LLMs in developing structured reasoning strategies for the legal entailment task in the COLIEE competition. This task requires determining whether a legal statement is true or false based on a given set of legal articles. Their best-performing approach explicitly instructed the model to employ legal reasoning frameworks such as IRAC (*Issue, Rule, Application, Conclusion*). Similarly, Nguyen et al. [60] compared GPT-3.5 and GPT-4 on the same task, finding that performance varied significantly depending on the context and the nature of the legal question. Their study raises concerns about the generalization capabilities of LLMs in adapting to diverse legal scenarios.

Finally, Trautman et al. [75] conducted an extensive benchmark evaluation to assess the groundedness of legal Q&A systems. Here, groundedness refers to the extent to which AI-generated responses are aligned with the input legal sources. The dataset included legal queries, AI-generated responses[9], and supporting legal texts (e.g., case law, statutes, and regulations). Their findings indicate that prompt chaining [74] achieved the highest performance, followed by direct prompting to GPT-*4o*.

[9]Verified by legal experts to ensure they adhered to legal sources.

2.6 LLMs Limitations

Despite the promising applications of LLMs in the legal field, several studies highlight their limitations and potential risks.

Dahl et al. [12] conducted an extensive study on LLM hallucinations, testing four models across 14 different tasks with varying levels of complexity. They evaluated the accuracy of LLM-generated responses over 5,000 judicial cases spanning different court hierarchies. Their results show that GPT-4 exhibited the highest hallucination rate, generating incorrect information in at least 58% of cases. The study attributes these hallucinations to two key factors: (1) the models' inability to handle misleading inputs, and (2) their lack of self-awareness regarding the certainty of their outputs, which undermines their reliability in legal contexts.

Martinez [56] critically assessed GPT-4's performance on the Uniform Bar Exam (UBE), challenging OpenAI's claim that it ranks in the top 10% of test-takers. The study finds that GPT-4's actual percentile rank is closer to the 62nd percentile overall and the 42nd percentile in essay sections, with performance declining further when compared to licensed attorneys. Concerns are also raised about the validity of OpenAI's essay grading methodology, emphasizing the need for rigorous independent evaluations before integrating AI into legal decision-making.

Chalkidis [6] evaluated GPT-3.5 on the LexGLUE benchmark [7], reporting an average micro-F1 score of 49 across various legal tasks in zero-shot and few-shot settings. One subtask consits of unfair clause detection over the CLAUDETTE dataset[10], it resulted in micro-F1 and macro-F1 scores of 64.7 and 32.5, respectively, further highlighting the limitations of LLMs in legal text classification.

[10] An older version of the dataset consisting of 100 documents.

Chapter 3

Method

3.1 Dataset

The corpus consists of 142 online consumer contracts, Terms of Service (ToS), relying on the previous work by Lippi et al. [45], Ruggeri et al. [67] and Jablonowska et al. [33]. The annotation is performed by legal expert and marked in XML as described in [45] where, including the following revisions, 9 different categories of unfairness are identified. Each category has its own scope definition and the conditions that make such clause unfair.

Jurisdiction $\langle j \rangle$: The jurisdiction clause specifies what courts have the competence to adjudicate disputes. A clause is (potentially) unfair whenever it states that judicial proceeding takes a residence away (i.e., in a different city, different country from the consumer place of residence).

Choice of Law $\langle law \rangle$: The choice of law clause specifies what law will govern the contract and be applied in potential disputes. A clause is (potentially) unfair whenever it states that the applicable law is different from the law of the consumer's place of residence the clause is unfair.

Limitation of Liability $\langle ltd \rangle$: The limitation of liability clause specifies for what actions/events and under what circumstances the providers exclude, limit or reduce their liability, the duty to compensate damages and/or when contains a blanket phrase like "to the fullest extent permissible by law". Such clause is

always (potentially) unfair, unless it is a force majeure case.

Unilateral Change $\langle ch \rangle$: The unilateral change clause specifies if and under what conditions the provider can unilaterally change and modify the contract and/or the service. Such clause is always (potentially) unfair.

Unilateral Termination $\langle ter \rangle$: The unilateral termination clause states that the provider has the right to suspend and/or terminate the service and/or the contract and/or the consumer's account, due to some reasons, or at any time, for any or no reasons with or without notice. Such clause is always (potentially) unfair.

Contract by Using $\langle use \rangle$: The contract by using clause states that the consumer is bound by the terms of use/service simply by using the service, downloading the app or visiting the website. Such clause is always (potentially) unfair.

Content Removal $\langle cr \rangle$: The content removal clause gives the provider a right to modify, delete or remove the user's content, including in-app purchases, under specific conditions or at any time, in his full discretion, for any or no reasons, with or without notice or the possibility to retrieve the content. Such clause is always (potentially) unfair.

Arbitration $\langle a \rangle$: The arbitration clause requires or allows the parties to resolve their disputes through the arbitration, before the case could go to court. A clause is (potentially) unfair whenever the arbitration is binding and not optional and/or should take place in a country different from the consumer's place of residence and/or be based not on law but on other arbitration rules and/or arbiter's discretion.

Privacy Included $\langle pinc \rangle$: Identify clauses (a) explicitly stating that, simply by using the service, the consumer consents to the processing of personal data as described in privacy policy; and/or (b) that the privacy policy is incorporated into and form part of the terms and it is preceded by a content by using clause to such terms. These clauses are always (potentially) unfair.

Is worth to note how each clause is described as *potentially* unfair. From a

strictly legal point of view, a clause is unfair with absolute certainty only if a competent institution expressed itself in that sense. Moreover, the unfairness may depend on the context of application of the clause.

All the analyzed ToS are standard terms available on the provider's website for review by potential consumers and since contract may vary by jurisdiction, only those concerning European customers were selected. Each contract is divided sentence-wise and each sentence can be classified as (1) *clearly fair*, (2) *potentially unfair* and (3) *clearly unfair* with the numeric value appended to the corresponding XML tag. A single sentence can have 0 or more tags assigned, moreover, those that does not concern any legal matter, i.e. "Please read these terms of service", don't have an assigned tag and are treated as *clearly fair* clauses.

3.1.1 Corpus statistics

The corpus contains 37,895 clauses for a total of 38,230 labels, out of these 3375 are unfair corresponding to 3049 ($\sim 8\%$) clauses containing at least 1 unfair tag. The dataset is unbalanced since most of the sentences are fair (34,846), also the distribution of unfair clauses is not balanced with $\langle ltd \rangle$ and $\langle ter \rangle$ taking almost half (47.3%) of all potentially unfair clauses. Most of the unfair clauses[1] are tagged with just one class (2746), while clauses containing 2 (285) or 3 (25) tags are much less frequent. The distribution of categories across documents is reported in Table 3.1. Arbitration and privacy included are the least commons categories, found respectively in 53 and 80 documents, while all the other categories appear in at least 102 documents out of 142 with limitation of liability appearing in almost all the documents (139).

[1] Meaning the ones that contains at least 1 unfair tag

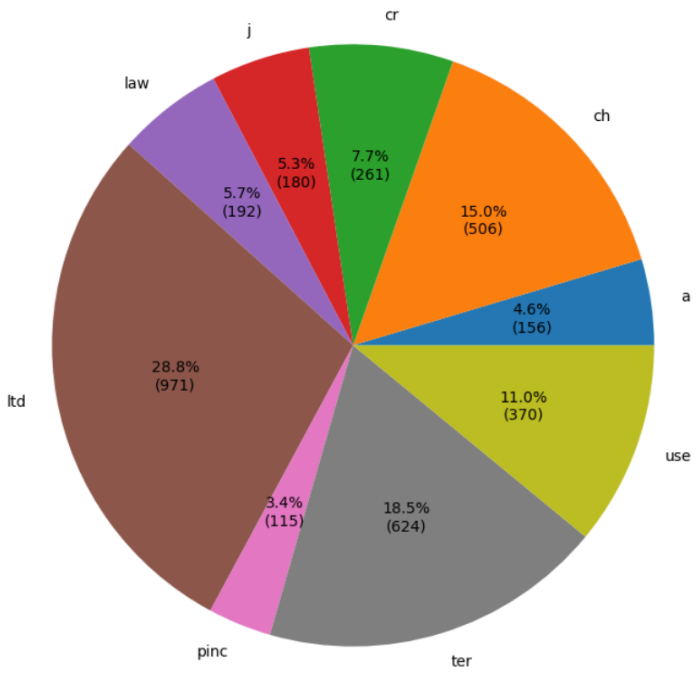

Figure 3.1: Distribution of unfair clauses

3.2 Methodology

The unfair clause detection task addressed in this work can be formally described as a multi-label sentence classification problem. In a multi-label classification task, each input instance can be associated with multiple labels simultaneously. Given a dataset of sentences, our goal is to assign one or

Type of clause	Tag	#Clauses	#Unfair	#Documents
Arbitration	< a >	165	156	53
Unilateral Change	< ch >	506	506	138
Content Removal	< cr >	261	261	102
Jurisdiction	< j >	218	180	110
Choice of Law	< law >	225	192	130
Limitation of Liability	< ltd>	1072	971	139
Unilateral Termination	< ter>	624	624	134
Consent by Using	< use>	370	370	129
Privacy Included	< pinc >	115	115	80

Table 3.1: Composition of the final corpus. The number of unfair clauses groups both *potentially unfair* and *clearly unfair*.

more labels from a predefined set of unfair categories to each sentence. Formally, let $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$ be a collection of N sentences, and let $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_M\}$ be a set of M possible labels. Each sentence s_i is associated with a subset of labels $\mathcal{L}_i \subseteq \mathcal{L}$. A common way to represent label assignments is through a binary vector $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{iM}] \in \{0,1\}^M$, where each element y_{ij} indicates the presence ($y_{ij} = 1$) or absence ($y_{ij} = 0$) of the label ℓ_j for sentence s_i. The goal is to learn a function $f : \mathcal{S} \to \{0,1\}^M$ that maps each sentence to its corresponding binary label vector. The function f can be learned, in principle, with any supervised classifier provided that the training dataset completely represents the event that f tries to approximate.

Existing approaches achieve state-of-the-art performance for unfair clause detection using transformers model such as Legal-BERT and DeBERTa [7]. The assessment over this type of models is repeated by using the new dataset of 142 documents which include also the additional <pinc> category, then, the same task is addressed with LLMs. Three different approaches are tested within the zero-shot and few-shot settings in order to evaluate what strategy is better to undertake this task.

3.2.1 Transformer-based

The setting to assess transformer models is the same used in the LexGLUE benchmark [7]. In general, these models are pre-trained on large-scale unlabeled corpora in a self-supervised manner, where they learn to predict masked tokens. After pre-training, these models are fine-tuned on task-specific annotated datasets by specific layers and optimizing performance through supervised learning. In this case, the input clause is fed to the model which returns its high-level representation h, this is then processed by a linear layer L followed by a sigmoid activation. The output of this process is a $1 \times k$ vector, where k is the number of labels, containing a probability for each of the k labels. Since the *fair* label is explicitly included in the classification, the total

number of labels is $k = 9 + 1$. Since the task is a multi-label classification a binary cross-entropy (BCE) loss function is used to independently evaluates each label as a separate binary classification problem. The BCE loss is a special case of the cross-entropy (eq. 2.8), specifically applied when there are only two possible classes (positive or negative) for each label. For a single sample, the BCE is defined as follows:

$$L_{BCE}(x, y) = -\sum_{i=1}^{k}[y_i \log(x_i) + (1 - y_i) \log(1 - x_i)] \tag{3.1}$$

where x is the output probability vector of the model classification head and y is a one-hot encoded vector of the true label.

3.2.2 LLMs

There are several ways in which the unfair clause detection task can be formulated for LLMs. Since it is well-known that the performance of LLMs heavily depends on the adopted prompt, we experimented with different prompts. In all cases, we define prompts that are mostly based on the annotation guidelines, i.e., on the definitions of the nine categories of clauses and the conditions of their unfairness, as detailed in section 3.1. Indeed, the category definitions provide fundamental information, since the categories' names may not be informative enough. For defining prompts three strategies are implemented, which differ in how they address the tasks of detecting (i.e., recognizing whether a clause is potentially unfair) and classifying it (i.e., identifying the category of unfairness). We name the three strategies *single-prompt*, *multi-prompt*, and *pipeline*. They will be described in detail in the next subsections. When the proposed strategy allows for it, we experimented both with zero-shot and few-shot learning settings.

Single-prompt approach

In this approach, both detection and classification are performed simultaneously over a clause, with a single prompt. The prompt uses a comprehensive template that includes all necessary information: the tasks description, the definition of the nine categories as given in Section 3.1, and detailed instructions for output formatting. The expected output from this prompt is a list containing all the unfairness categories for each input clause. While adequate to zero-shot scenarios, this approach would become cumbersome in a few-shot setting, as it would require incorporating several examples for each category into a single prompt. Incorporating n examples for each of the nine categories leads to an input prompt containing $n \times 9$ examples, alongside the template structure, making it less scalable.

Multi-prompt approach

To address the scalability issues of the single-prompt approach, we decompose the problem into several subtasks. Differently from the previous method each category is addressed separately. Given an input clause, the model is queried with 9 separate prompts, one for each category, so that it focuses exclusively on one category at a time. This design minimizes the risk of misinterpreting the task requirements or the formatting instructions. In this way, the multi-label classification task is effectively transformed into a set of nine binary classification tasks. This approach supports both the zero-shot and few-shot learning settings, and the prompt template is the following:[2]

```
Evaluate the following clause from a terms of service based on
the provided definition of unfairness:
Definition: {category definition}
If the clause is unfair according to the definition respond with
'yes' otherwise respond with 'no'. Provide only 'yes' or 'no'
as the output. Do not include any explanations or additional
```

[2] At execution time, text within brackets is replaced with the corresponding entities.

```
text.
Examples: {examples}[3]
Test Input:
Clause: {clause}
Response:
```

Pipeline approach

The third methodology builds on the multi-prompt approach and further decomposes the task into two distinct phases of a pipeline. During the first phase, i.e., *category identification*, the model determines whether a clause belongs to a specific category. To this aim, we perform one query for each category, using the same definitions of the previous approaches, but omitting the part describing the unfairness conditions. Similarly to the *single-prompt approach*, this phase is always performed in a zero-shot setting for scalability reasons. If all prompts respond negatively (i.e., the clause belongs to none of the categories) the clause is deemed *fair*. This phase can also be implemented in a few-shot learning mode, requiring a dedicated set of examples for each category. To ensure balance, these examples should include both fair and unfair clauses within the target category, as well as clauses from other categories. However, in this study, the initial prompt is designed for zero-shot learning to avoid increasing inference time. The prompt template for this first stage is the following:

```
You are analyzing terms of service. Please, tell whether the
clause talks about the subject specified in the definition.
Definition: {category definition}
Please answer only with 'yes' or 'no'. Do not include any explanations
or additional text.
Test Input:
Clause: {clause}
Response:
```

[3] This line is not present in the zero-shot setting.

The definitions used in this case are the same as before but without the sentence describing the unfairness conditions. If a clause is positively identified in this first step, a second query assesses its unfairness. For clauses associated with one or more categories, we proceed with the second phase, i.e. the *unfairness assessment*. Specifically, if in the first phase a clause is positively identified as belonging to a category, in the second phase, a second query assesses its unfairness for that category. The unfairness conditions corresponds to the second sentence of the category definitions (**??**), with the exception of those that only state "*Such clause is always unfair*", in order to minimize potential bias in the model. The prompt for the second phase is always performed in a few-shot setting. The template is the following:

```
You are analyzing terms of service. Please, tell whether the
input clause is unfair. {unfairness conditions}. If the clause
is unfair respond with 'yes' otherwise respond with 'no'. Provide
only 'yes' or 'no' as the output. Do not include any explanations
or additional text.
Examples: {examples}
Test Input:
Clause: {clause}
Response:
```

Both phases use one prompt per category, for a total of 9 to 18 prompts per sentence.

3.2.3 Selection of examples for few-shot setting

The multi-prompt and the pipeline approaches are designed to support the few-shot learning settings. In these settings, for each category, the prompt includes 8 examples of unfair clauses. These examples, extracted from the training set, are hand–selected according to the following criteria: (i) coverage of different market sectors; (ii) coverage of diverse types of unfair contracting practices, within each unfair-clause category; (iii) length of clauses. Long sentences

usually offer several advantages. As noted, they encapsulate a wide range and different combinations of unfair practices, thus providing a richer and more detailed context. This may help the models to capture complex relationships and generalize across diverse scenarios. However, they also come with disadvantages. Small-scale language models may struggle with processing lengthy sentences, due to input length constraints or difficulty in capturing long-range dependencies. Furthermore, the dense nature of long sentences can lead to increased ambiguity, potentially hindering the models' ability to focus on specific features.

Given these limitations also shorter sentence are considered by identifing 8 new examples. A key advantage of this approach is that short sentences are simpler to process and reduce the risk of truncation. This approach may allow models to focus on specific patterns. However, it comes with potential issues. Indeed, there is a risk of oversimplification. Relying on short sentences may reduce cohesiveness and create a misalignment with real world contractual language, since ToS often contains long and complex sentences. Furthermore, models may lose the broader context necessary to fully understand the relationships between different elements. Thus, practices that depend on interconnected factors may not be fully captured.

3.3 Experiments

For the experimental evaluation, the dataset considered is described in Section 3.1, using the methodology illustrated in Section 3.2. Following the setting of LexGLUE [7], the 142 documents are split into three sets: 85 documents for training, 35 for validation, and 22 for test. To address class imbalance, all the approaches are evaluated by measuring micro-F1 and macro-F1. As for LLMs, we consider 9 different models (Table 3.2): namely, Gemma2-2B, Gemma2-9B, Llama3-8B, Mistral-7B, Law-Chat, MistralNemo-12B, Phi3-14B, Codestral-22B, Qwen-32B. All the experiments have been executed on

two RTX 2080 Ti for a total of 22GB of VRAM. Quantization was employed when necessary, i.e., when the chosen model did not fit in the available VRAM. The 8-bit integer quantization consists of a mixed-precision decomposition where most of the values ($\approx 99.9\%$) are quantized in 8-bit precision, while outlier features are quantized to 16-bit [13]. Concerning 4-bit quantization, the QLoRA [14] method is employed.

3.3.1 Experimental Setting

All the LLMs are instruction-tuned models between 2 and 32 billion parameters (Table 3.2). The input prompt for each model is pre-processed using the recommended template for that particular model. Specific tokens to the input prompt (e.g., [INST]) are appended and pre-pended, ensuring alignment with the model's native requirements. Temperature is set to 1, sampling is disabled, and the greedy search decoding method is employed to ensure reproducible results. The model weights are quantized to 8-bit or 4-bit integers, depending on the size of the model as indicated in Table 3.2, to meet the computational resources at disposal.

Model	#Params	Context Length	Quantization
Qwen [32]	32B	128k	4 bit
Codestral [71]	22B	32k	4 bit
Phi3 [70]	14B	128k	8 bit
Nemo [72]	12B	128k	8 bit
Gemma2 [25]	9B	4k	8 bit
Llama3 [17]	8B	128k	8 bit
Mistral [37]	7B	32k	8 bit
Law-Chat [8]	7B	128k	8 bit
Gemma2 [25]	2B	4k	8 bit

Table 3.2: Details of the large language models.

The LLMs are compared against a linear SVM and 7 pre-trained transformers, as used in the LexGLUE benchmark [7]. The SVM is the same used

in the original CLAUDETTE system [45]: it employs TF-IDF features using n-grams with $n \in [1, 2, 3]$, with hyperparameters[4] optimized through grid search.

All pre-trained models are fine-tuned for up to 20 epochs with early stopping on development data, using the Adam optimizer with a starting learning rate of $3e^{-5}$. Mixed precision ($fp16$) is used to reduce memory requirements, with a batch size of 8 across all experiments. Each setup is repeated five times with different random seeds, reporting test scores based on the seed yielding the best test performance.

3.3.2 Model Ensemble

Combining predictions from multiple models generally leads to improved performance compared to relying on a single model. This is because different models may excel in distinct regions of the solution space, compensating for each other's weaknesses. The ensemble method proposed in this study involves constructing a meta-model that aggregates predictions from multiple models. For a given model m its prediction is represented as a one-hot vector $y^{(m)}$ where $y_i^{(m)} = 1$ if the i-th class is positively predicted. Given n models then their respective one-hot vectors are summed to obtain a single aggregated vector $\hat{y}$ which represents the combined predictions across all models. Based on this framework, three different ensemble strategies are proposed to determine the final output of the meta-model.

Majority Voting

In majority voting, a class is predicted as positive if the majority of models agree on that class, meaning $\hat{y}_i > t$ where t is a threshold equals to $\lfloor n/2 \rfloor + 1$. This ensures that a class is selected only if more than half of the models predict it.

[4] Number of features, regularization strength and loss function

Weighted Voting

Unlike majority voting, where all models contribute equally, weighted voting assigns greater importance to models that demonstrate superior performance. Each model contributes to the final aggregated vector $\hat{y}$ bbased on a normalized weight, which is determined by a performance metric. In this case, the average macro-F1 score is used as the weighting factor, moreover the final prediction threshold can be adjusted based on the desired confidence level of the meta-model.

Softmax Voting

In softmax voting, the aggregated vector $\hat{y}$ is transformed into a probability distribution by applying a $softmax$ function (Eq. 2.5). Again, the final output for a class i is positive if the score for that class is greater than a threshold.

In the experimental evaluation, these three strategies are tested by considering the 4 best models, based on the macro F1, resulting from the previous evaluation. The threshold for the *weighted voting* scheme is set to 0.5 while for the *softmax* scheme it is derived by running a grid search over the ensemble and testing different level of thresholds.

Chapter 4

Discussion

Model	Validation		Test			
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Macro Acc.	Macro Rec.
Bert-base-uncased	95.8 ± 0.1	74.8 ± 0.8	95.6	75.3	80.5	71.2
Roberta-base	94.6 ± 1.5	49.8 ± 25.7	95.6	70.2	72.4	68.8
Deberta-base	95.8 ± 0.1	73.7 ± 0.2	**95.8**	**77.8**	**82.5**	74.6
Longformer-base-4096	93.5 ± 1.5	32.4 ± 20.7	95.5	62.7	70.3	57.6
Bigbird-Roberta-base	95.7 ± 0.0	72.6 ± 2.3	95.3	75.3	77.3	74.9
Legal-Bert-base-uncased	96.0 ± 0.1	76.0 ± 1.4	95.6	76.9	78.5	**77.1**
Custom-Legalbert	95.9 ± 0.1	75.4 ± 0.3	**95.9**	**77.9**	81.2	75.6
TFIDF+SVM	95.0 ± 0.0	62.0 ± 0.0	94.9	61.0	80.7	52.4

Table 4.1: Evaluation of baseline methods on the validation and the test split. For validation, we report the average score over 5 training and the variance. For test, we report the result of the best model.

4.1 Transformer-based

Table 4.1 shows the results obtained by baseline approaches, which are those used also within the LexGLUE benchmark [7]. Results are similar to those reported in LexGLUE, although we are hereby using a larger dataset of 142 documents, which are more diverse in terms of market sectors, and thus more challenging. Furthermore, the privacy included category utilized in this dataset was not part of the LexGLUE benchmark. Custom-LegalBERT is the best performing approach, with 95.9/77.9 of micro/macro-F1, respectively.

4.2 Comparison of prompt strategies

Table 4.2 shows the results obtained with the three different prompt strategies described in Section 3.2. When using few-shot learning, the results obtained with both short and long examples are also reported.

It is straightforward to observe that the performance of LLMs is significantly lower than those obtained by the baselines. The *pipeline* strategy is the one that consistently performs better than the others, for all the considered LLMs. This is probably due to the fact that LLMs tend to consider unfair also clauses that are totally unrelated to any unfairness category. Therefore, splitting the task into two subsequent stages, and first filtering out the clauses that are unrelated to unfairness categories, greatly simplifies the problem for LLMs (each of the two prompts is much simpler to address).

Regarding the complexity of examples provided in the few-shot setting, there is no huge difference in adopting short or long examples, with a slight advantage in using the latter. While small-size LLMs do not produce satisfying results, it is evident that performance increases with model size: Qwen-32B, although quantized with 4 bits only, achieves results that are not far from the best-performing BERT-based baselines. On the other hand, Codestral-22B is underperforming even smaller LLMs. These suboptimal performances can be attributed to its pre-training on a corpus predominantly composed of code, which significantly diverges from the linguistic and conceptual characteristics of legal texts required for the task at hand.

4.2.1 Per-category evaluation

Table 4.3 presents the F1 scores of the best models across all the unfairness categories. In particular, the comparison is between the transformer-based models, i.e., Legalbert and Deberta-base, with small (Gemma2-2B, Gemma2-9B, Phi3-14B, Llama3-8B, Law-Chat-7B, Mistral-7B, Nemo-12B) and large LLMs (Codestral 22B and Qwen 32B). For each model, only the setting that

Model	Technique	Examples	Metrics			
			Micro-F1	Macro-F1	Macro Acc.	Macro Rec.
Gemma2-2B	Single-prompt	/	0.72	0.26	0.43	0.42
Llama3-8B	Single-prompt	/	0.71	0.37	0.38	0.61
Mistral-7B	Single-prompt	/	0.55	0.23	0.25	0.42
Law-Chat	Single-prompt	/	0.22	0.09	0.17	0.07
Gemma2-9B	Single-prompt	/	0.84	0.45	0.39	0.61
Nemo-12B	Single-prompt	/	0.72	0.39	0.42	0.57
Phi3-14B	Single-prompt	/	0.60	0.33	0.30	0.58
Codestral-22B	Single-prompt	/	0.15	0.19	0.37	0.17
Qwen-32B	Single-prompt	/	0.87	0.49	0.43	0.64
Gemma2	Multi-prompt - Zero shot	/	0.64	0.28	0.22	0.70
LLama3-8B	Multi-prompt - Zero shot	/	0.88	0.47	0.43	0.57
Mistral-7B	Multi-prompt - Zero shot	/	0.41	0.16	0.14	0.80
Law-Chat	Multi-prompt - Zero shot	/	0.83	0.39	0.33	0.62
Gemma2-9B	Multi-prompt - Zero shot	/	0.58	0.43	0.35	0.78
Nemo-12B	Multi-prompt - Zero shot	/	0.61	0.26	0.20	0.81
Phi3-14B	Multi-prompt - Zero shot	/	0.79	0.41	0.31	0.77
Codestral-22B	Multi-prompt - Zero shot	/	0.34	0.21	0.19	0.86
Qwen-32B	Multi-prompt - Zero shot	/	0.88	0.61	0.56	0.75
Gemma2-2B	Multi-prompt - Few shot	Long	0.41	0.20	0.17	0.87
		Short	0.57	0.30	0.24	0.86
Llama3-8B	Multi-prompt - Few shot	Long	0.79	0.40	0.30	0.84
		Short	0.78	0.40	0.31	0.78
Mistral-7B	Multi-prompt - Few shot	Long	0.53	0.24	0.20	0.85
		Short	0.52	0.22	0.18	0.87
Law-Chat	Multi-prompt - Few shot	Long	0.51	0.31	0.26	0.71
		Short	0.71	0.43	0.41	0.62
Gemma2-9B	Multi-prompt - Few shot	Long	0.85	0.53	0.42	0.83
		Short	0.89	0.58	0.47	0.85
Nemo-12B	Multi-prompt - Few shot	Long	0.32	0.15	0.15	0.86
		Short	0.31	0.14	0.14	0.87
Phi3-14B	Multi-prompt - Few shot	Long	0.72	0.43	0.34	0.85
		Short	0.80	0.52	0.40	0.80
Codestral-22B	Multi-prompt - Few shot	Long	0.41	0.24	0.21	0.87
		Short	0.44	0.26	0.21	**0.89**
Qwen-32B	Multi-prompt - Few shot	Long	0.90	0.59	0.47	0.84
		Short	0.88	0.61	0.51	0.85
Gemma2-2B	Pipeline	Long	0.89	0.56	0.49	0.72
		Short	0.90	0.57	0.52	0.69
Llama3-8B	Pipeline	Long	0.90	0.58	0.51	0.71
		Short	0.91	0.55	0.54	0.61
Mistral-7B	Pipeline	Long	0.74	0.29	0.24	0.70
		Short	0.59	0.24	0.19	0.87
Law-Chat	Pipeline	Long	0.43	0.20	0.17	0.87
		Short	0.46	0.22	0.18	0.85
Gemma2-9B	Pipeline	Long	0.84	0.51	0.40	0.83
		Short	0.85	0.52	0.41	0.84
Nemo-12B	Pipeline	Long	0.92	0.62	0.57	0.72
		Short	0.92	0.60	0.55	0.71
Phi3-14B	Pipeline	Long	0.87	0.56	0.44	0.84
		Short	0.87	0.57	0.46	0.82
Codestral-22B	Pipeline	Long	0.80	0.45	0.35	0.85
		Short	0.80	0.47	0.36	0.85
Qwen-32B	Pipeline	Long	**0.95**	**0.72**	**0.77**	0.71
		Short	0.94	0.71	**0.77**	0.71

Table 4.2: Evaluation of LLMs over the test set. For each column, we report the best result in bold.

Model	Technique	Category F1									
		fair	a	ch	cr	j	law	ltd	ter	use	pinc
Baselines											
Custom-Legalbert	Fine tuned	0.97	0.56	**0.78**	0.67	**0.96**	**0.93**	0.70	**0.75**	0.79	0.67
Deberta-base	Fine tuned	**0.98**	**0.67**	0.76	**0.70**	0.90	**0.93**	**0.71**	0.73	**0.82**	0.57
Small LLMs											
Gemma2-2B	Pipeline-Short	0.95	0.45	0.58	0.58	0.64	0.61	0.48	0.58	0.52	0.29
Mistral-7B	Pipeline-Long	0.85	0.07	0.19	0.08	0.12	0.1	0.3	0.38	0.14	0.20
LawChat	Multi Few-Short	0.87	0.36	0.32	0.08	0.65	0.24	0.19	0.55	0.62	0.38
Gemma2-9B	Multi Few-Short	0.95	0.35	0.51	0.41	0.68	0.71	0.52	0.59	0.68	0.44
Phi3-14B	Pipeline-Short	0.93	0.33	0.52	0.45	0.74	0.71	0.46	0.61	0.55	0.44
Llama3-8B	Pipeline-Long	0.95	0.49	0.64	0.55	0.68	0.54	0.55	0.58	0.42	0.36
Nemo-12B	Pipeline-Long	0.96	0.33	0.58	0.56	0.72	0.79	0.62	0.63	0.51	0.48
Large LLMs											
Qwen-32B	Pipeline-Long	**0.98**	0.48	0.70	0.64	0.91	**0.93**	0.67	0.57	0.65	**0.69**
Codestral-22B	Pipeline-Short	0.90	0.19	0.54	0.37	0.53	0.47	0.54	0.47	0.18	0.47

Table 4.3: Evaluation of models for each category of unfairness. For each column, it is reported the best results in bold and underline the best result of small LLMs.

yields the best result are considered. The fine-tuned transformer-based models, Legalbert and Deberta-base, outperform other techniques across most categories, thus highlighting the benefits of tailoring models to legal datasets and corpora. As for the aggregate results presented in Table 4.2, small LLMs perform significantly worse both than large LLMs and than BERT-based approaches. In particular, Legalbert achieves the highest F1 scores for unilateral change (0.78 vs 0.76), jurisdiction (0.96 vs 0.90), and unilateral termination (0.75 vs 0.73). Conversely, Deberta-base surpasses Legalbert in arbitration (0.67 vs. 0.56), content removal (0.70 vs 0.67), limitation of liability (0.71 vs 0.70), and contract by using clauses (0.82 vs. 0.79). The two models achieve an equal F1 score for choice of law (0.93). Even the per-category results confirm that the pipeline strategy is the one achieving the best results. As noted above, Nemo-12B is the best performing LLM among the small-sized ones, in particular as regard to content removal (0.64), choice of law (0.80) and privacy included (0.45) for pipeline short. Similarly, Llama3-8B reports the best results for arbitration (0.49), unilateral change (0.64) and limitation of liability (0.55) for pipeline long. Phi3-14B outperforms in assessing jurisdiction (0.74) and unilateral termination clauses (0.61) in pipeline short. The full

comparison between the short and long examples confirms the observation made for aggregate results (Table 4.2), i.e., that that the selection of examples does not strongly influence the performance of small LLMs within the pipeline approach. Differently, Gemma2-9B achieves the best results under the *Multi-prompt* short approach as regards consent by using (0.62). The best performing strategy for Mistral-7B is using *Pipeline-long*, but its results are significantly worse than those of other small LLMs. Gemma2-2B, despite being the smallest among the tested LLMs, delivers performance comparable to larger models, including its bigger counterpart, Gemma2-9B.

The frequency and unfairness distribution of clauses in the dataset correlate with the performance trends of small-size LLMs. Categories with higher clause frequencies, such as Limitation of Liability and Unilateral Termination, generally see better performance from fine-tuned models, reflecting their ability to adapt to well-represented clause types. Conversely, less frequent categories like Privacy Included and Arbitration highlight the limitations of few-shot and pipeline methods and the need for extensive domain-specific training.

4.2.2 Error analysis

Figure 4.1 and 4.2 presents the per-label confusion matrices for the two best performing LLMs, namely Qwen-32B and Nemo-12B both using the pipeline approach with longer examples. It is worth to highlight how even in the categories where they performs similarly ($fair$, cr and ter) they make different mistakes. For both ter and cr, Qwen has more false negatives (FN) than Nemo, while for the $fair$ category is the opposite, with Qwen making more false positive (FP) mistakes. In the law category Qwen performs on par with the fine-tuned models, here the two false negatives errors are the followings:

> *these terms shall be governed by and construed in accordance with the laws of the state of california.*

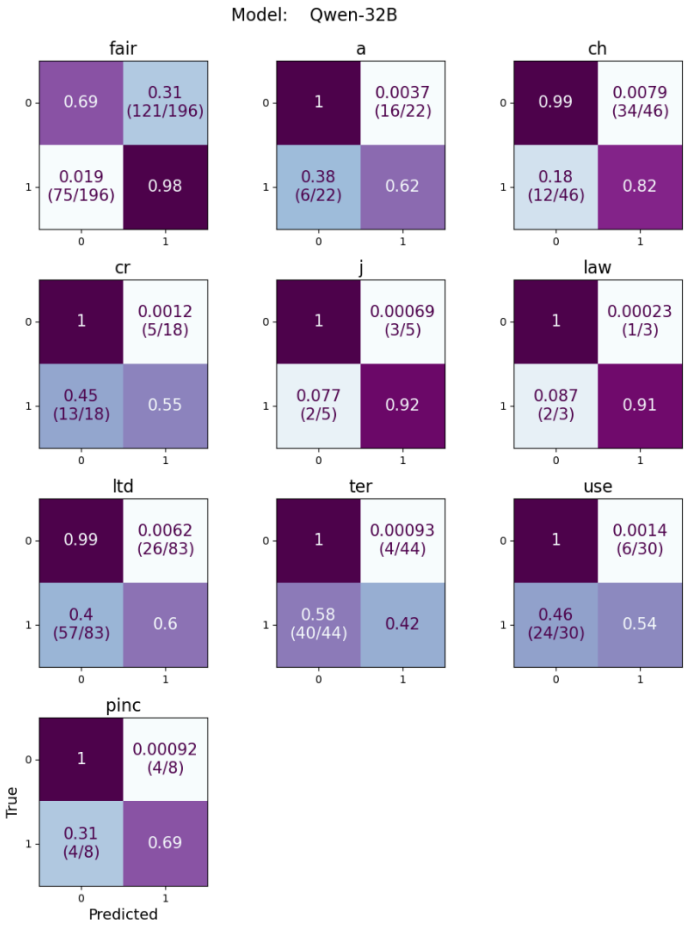

Figure 4.1: Per-label confusion matrix for the Qwen-32B model on the Pipeline-Long approach. All elements are normalized along the rows, also the number of FP and FN errors over the total error count are displayed.

> *the licensee's use of the service may also be subject to other local, state, national or international laws.*

which should be fairly easy to identify given the definition of unfairness used in the prompt for the *Choice of Law* category. The *arbitration* category is the most arduous for both small and large LLMs, in this case both Qwen and Nemo performs the same type of mistakes with an higher number of false positive, the same can be noticed for Llama3 (15 FP vs 6 FN) which is the best performing LLM in general on arbitration. For what strictly concern Qwen, almost all the false positives clauses are instead fair clauses that contains terms such as ”arbitration”, ”arbitrator” or a derivation of the verb ”arbitrate”, meaning

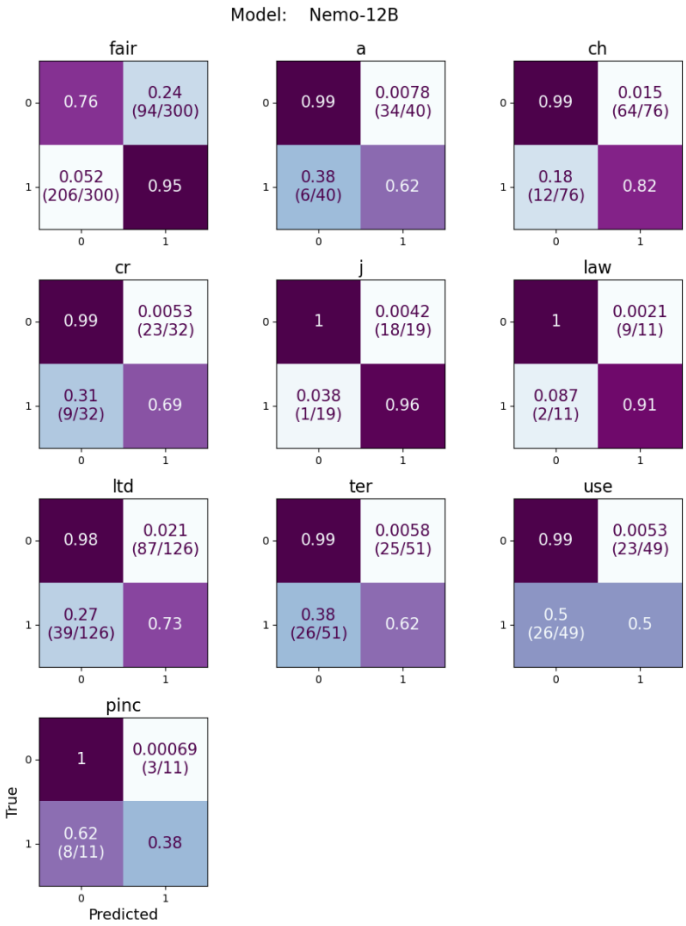

Figure 4.2: Per-label confusion matrix for the Nemo-12B model on the Pipeline-Long approach. All elements are normalized along the rows, also the number of FP and FN errors over the total error count are displayed.

that the model is simply elicited by the presence of such terms and not by the conditions specified in the arbitration definition. The test dataset include 41 clause that are relevant for more than one unfair category[1]. Table 4.4 shows the comparison for this subset of the dataset for both Qwen and Nemo, opposite to the average macro F1 in this case the smaller model performs better on identifying clauses with more than one unfairness.

[1]Respectively 36 clauses for 2 unfair categories and 5 clauses for 3

Model	Technique	Category F1									Micro-F1	Macro-F1
		a	ch	cr	j	law	ltd	ter	use	pinc		
Support		1	16	17	2	2	5	27	10	7		
Nemo-12B	Pipeline-Long	0.0	0.85	0.69	1	1	1	0.79	0.82	0.44	**0.76**	**0.66**
Qwen-32B	Pipeline-Long	0.0	0.77	0.62	1	1	0.33	0.67	0.89	0.92	0.69	0.62

Table 4.4: Comparison of Nemo-12B and Qwen32B on clauses containing more than one unfairness.

4.2.3 Ensembling

Table 4.5 presents the experimental results for the ensembled models applied to the *Pipeline* approach for both long and short examples. The meta-models are constructed in two ways: *4 mix*, which combines the four best models overall, and *4 small*, which selects the four best smaller models based on macro F1. In the *4 mix* ensemble, the models used are Gemma2-2B, Llama3, Nemo, and Qwen, whereas in the *4 small* ensemble, Phi3 replaces Qwen among the selected models.

As expected, ensembling generally improves performance compared to individual models, though the extent of improvement varies across different ensemble strategies. The best-performing approach is the *4 mix* meta-model applied to the *Pipeline-Long* setting using both the hard and weighted voting schemes, achieving a macro F1 of 0.73. However, the improvement over Qwen-32B (0.72 macro F1) is minimal, likely because the smaller models contribute little to the final prediction, with Qwen dominating the ensemble's decision-making.

The advantage of ensembling becomes more apparent when considering the *4 small* meta-model, which achieves a macro F1 of 0.71 with the weighted voting scheme on the *Pipeline-Short* approach. This result consistently outperforms the best individual small model (Nemo-12B), highlighting the effectiveness of ensembling in this setting.

Regarding the three voting strategies, the *hard* scheme performs best among the smaller model ensembles, while the *weighted* scheme yields the best results for both *4 mix* meta-models. However, the differences between the three

strategies are minor, with no single approach standing out as clearly superior. That said, the *hard* voting scheme appears to be the most stable across all tested settings.

Technique	Models	Ensemble Strategy	Metrics			
			Micro-F1	Macro-F1	Macro Acc.	Macro Rec.
Pipeline-Long	4 mix	Softmax	0.94	0.72	0.79	0.66
		Hard	0.94	**0.73**	**0.80**	0.69
		Weighted	**0.95**	**0.73**	0.76	0.73
Pipeline-Short	4 mix	Softmax	0.93	0.69	0.62	**0.80**
		Hard	0.93	0.71	0.73	0.71
		Weighted	0.94	0.72	0.72	0.75
Pipeline-Long	4 small	Softmax	0.92	0.70	0.72	0.69
		Hard	0.92	0.71	0.70	0.74
		Weighted	0.92	0.67	0.60	0.78
Pipeline-Short	4 small	Softmax	0.92	0.67	0.73	0.64
		Hard	0.92	0.69	0.71	0.68
		Weighted	0.92	0.66	0.61	0.75

Table 4.5: Results of the ensembled models for all the three modes over the *Pipeline* approach. The **4 mix** model consists of the 4 best model overall (Gemma2-2B, Llama3, Nemo, Qwen) while **4 small** comprehends the 4 best models among the smaller ones (Gemma2-2B, Llama3, Nemo, Phi3).

Chapter 5

Conclusions

In this work, an extensive evaluation is performed on the capabilities of LLMs compared to BERT-based models in detecting unfair clauses in ToS. Identifying the optimal prompt for an LLM on a given task is a challenging problem, since LLMs are sensitive to context and phrasing in ways that are often unpredictable and difficult to fully explain [50, 24, 66]. Thus, several prompting strategies are tested across 7 small LLMs, and two larger LLMs, against the state-of-the-art transformer models. Small LLMs could provide an attractive alternative to the supervised systems, including reduced dependency on extensive training data, lower resource requirements, and a smaller environmental footprint compared to larger models. Despite these benefits, the experiments results show that small LLMs currently fall short of delivering satisfactory performance in unfair clause detection. Among small models, Nemo-12B emerges as the best-performing one, with a pipeline approach, that decomposes the task into category identification and unfairness assessment, yielding the best results. However, no significant differences are observed between short and long example prompts. A larger model, Qwen-32B, demonstrates promising performance, with potential for further enhancement through fine-tuning. Conversely, Codestral-22B shows poor results across all the several approaches. Based on these findings, at this stage, BERT-based models remain superior for detecting and classifying unfair clauses in consumer contracts. In

conclusion, the experimental results show that small LLMs face significant limitations in processing the complex and varied language structures of legal contracts even when provided with relevant examples.

Regarding the possible directions for future research, one important next step is to test these models on multilingual datasets to see how well they adapt across different legal systems and languages. Additionally, experimenting with alternative prompting techniques, such as chain-of-thought prompting, could improve both the interpretability and accuracy of the models. Another key challenge is keeping up with changes in legal language and regulations. Exploring continual learning approaches could help ensure that these models remain reliable and effective as laws evolve over time.

Bibliography

[1] Àlex R. Atrio and Andrei Popescu-Belis. Small batch sizes improve training of low-resource neural MT. *CoRR*, abs/2203.10579, 2022. DOI: 10.48550/ARXIV.2203.10579. arXiv: 2203.10579. URL: https://doi.org/10.48550/arXiv.2203.10579.

[2] J. Baker. The dragon system–an overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29, 1975. DOI: 10.1109/TASSP.1975.1162650.

[3] Alberto Barrón-Cedeño and Paolo Rosso. On automatic plagiarism detection based on n-grams comparison. In Mohand Boughanem, Catherine Berrut, Josiane Mothe, and Chantal Soule-Dupuy, editors, *Advances in Information Retrieval*, pages 696–700, Berlin, Heidelberg. Springer Berlin Heidelberg, 2009. ISBN: 978-3-642-00958-7.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, March 2003. ISSN: 1532-4435.

[5] Aindrila Chakraborty, Ramesh Shankar, and James R. Marsden. An empirical analysis of consumer-unfriendly e-commerce terms of service agreements: implications for customer satisfaction and business survival. *Electronic Commerce Research and Applications*, 53:101151, 2022. ISSN: 1567-4223. DOI: https://doi.org/10.1016/j.elerap.2022.101151. URL: https://www.sciencedirect.com/science/article/pii/S1567422322000357.

[6] Ilias Chalkidis. Chatgpt may pass the bar exam soon, but has a long way to go for the lexglue benchmark. *CoRR*, abs/2304.12202, 2023. DOI: 10.48550/ARXIV.2304.12202. arXiv: 2304.12202. URL: https://doi.org/10.48550/arXiv.2304.12202.

[7] Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael J. Bommarito II, Ion Androutsopoulos, Daniel Martin Katz, and Nikolaos Aletras. Lexglue: A benchmark dataset for legal language understanding in english. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 4310–4330. Association for Computational Linguistics, 2022. DOI: 10.18653/V1/2022.ACL-LONG.297. URL: https://doi.org/10.18653/v1/2022.acl-long.297.

[8] Daixuan Cheng, Shaohan Huang, and Furu Wei. Adapting large language models to domains via reading comprehension, 2024. arXiv: 2309.09530 [cs.CL]. URL: https://arxiv.org/abs/2309.09530.

[9] Noam Chomsky. Three models for the description of language. *IRE Trans. Inf. Theory*, 2(3):113–124, 1956. DOI: 10.1109/TIT.1956.1056813. URL: https://doi.org/10.1109/TIT.1956.1056813.

[10] Council directive 93/13/eec of 5 april 1993 on unfair terms in consumer contracts. URL: https://eur-lex.europa.eu/eli/dir/1993/13/oj/eng.

[11] Junyun Cui, Xiaoyu Shen, and Shaochun Wen. A survey on legal judgment prediction: datasets, metrics, models and challenges. *IEEE Access*, 11:102050–102071, 2023. DOI: 10.1109/ACCESS.2023.3317083. URL: https://doi.org/10.1109/ACCESS.2023.3317083.

[12] Matthew Dahl, Varun Magesh, Mirac Suzgun, and Daniel E. Ho. Large legal fictions: profiling legal hallucinations in large language models. *CoRR*, abs/2401.01301, 2024. DOI: 10.48550/ARXIV.2401.01301.

arXiv: 2401.01301. URL: https://doi.org/10.48550/arXiv.2401.01301.

[13] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *CoRR*, abs/2208.07339, 2022. DOI: 10.48550/ARXIV.2208.07339. arXiv: 2208.07339. URL: https://doi.org/10.48550/arXiv.2208.07339.

[14] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: efficient finetuning of quantized llms. *CoRR*, abs/2305.14314, 2023. DOI: 10.48550/ARXIV.2305.14314. arXiv: 2305.14314. URL: https://doi.org/10.48550/arXiv.2305.14314.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. DOI: 10.18653/V1/N19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

[16] Kasper Drawzeski, Andrea Galassi, Agnieszka Jablonowska, Francesca Lagioia, Marco Lippi, Hans Wolfgang Micklitz, Giovanni Sartor, Giacomo Tagiuri, and Paolo Torroni. A corpus for multilingual analysis of online terms of service. In Nikolaos Aletras, Ion Androutsopoulos, Leslie Barrett, Catalina Goanta, and Daniel Preotiuc-Pietro, editors, *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 1–8, Punta Cana, Dominican Republic. Association for Computational Linguistics, November 2021. DOI: 10.18653/v1/

2021.nllp-1.1. URL: https://aclanthology.org/2021.nllp-1.1/.

[17] Abhimanyu Dubey and Abhinav Jauhri et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. DOI: 10.48550/ARXIV.2407.21783. arXiv: 2407.21783. URL: https://doi.org/10.48550/arXiv.2407.21783.

[18] Jeffrey L. Elman. Finding structure in time. *Cogn. Sci.*, 14(2):179–211, 1990. DOI: 10.1207/S15516709COG1402_1. URL: https://doi.org/10.1207/s15516709cog1402%5C_1.

[19] Eur-lex. URL: https://eur-lex.europa.eu/homepage.html.

[20] Yi Feng, Chuanyi Li, and Vincent Ng. Legal case retrieval: A survey of the state of the art. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 6472–6485. Association for Computational Linguistics, 2024. DOI: 10.18653/V1/2024.ACL-LONG.350. URL: https://doi.org/10.18653/v1/2024.acl-long.350.

[21] Yi Feng, Chuanyi Li, and Vincent Ng. Legal judgment prediction: A survey of the state of the art. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 5461–5469. ijcai.org, 2022. DOI: 10.24963/IJCAI.2022/765. URL: https://doi.org/10.24963/ijcai.2022/765.

[22] Andrea Galassi, Kasper Drazewski, Marco Lippi, and Paolo Torroni. Cross-lingual annotation projection in legal texts. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 915–926, Barcelona, Spain (Online). International Committee on Computational Linguistics,

December 2020. DOI: 10.18653/v1/2020.coling-main.79. URL: https://aclanthology.org/2020.coling-main.79/.

[23] Andrea Galassi, Francesca Lagioia, Agnieszka Jabłonowska, and Marco Lippi. Unfair clause detection in terms of service across multiple languages. *Artificial Intelligence and Law*, 2024. URL: https://api.semanticscholar.org/CorpusID:268912855.

[24] Chengguang Gan and Tatsunori Mori. Sensitivity and robustness of large language models to prompt template in japanese text classification tasks. In *PACLIC*, pages 1–11. Association for Computational Linguistics, 2023.

[25] Google DeepMind Gemma Team. Gemma 2: improving open language models at a practical size. *CoRR*, abs/2408.00118, 2024. DOI: 10.48550/ARXIV.2408.00118. arXiv: 2408.00118. URL: https://doi.org/10.48550/arXiv.2408.00118.

[26] Randy Goebel, Yoshinobu Kano, Mi-Young Kim, Juliano Rabelo, Ken Satoh, and Masaharu Yoshioka. Overview and discussion of the competition on legal information, extraction/entailment (COLIEE) 2023. *Rev. Socionetwork Strateg.*, 18(1):27–47, 2024. DOI: 10.1007/S12626-023-00152-0. URL: https://doi.org/10.1007/s12626-023-00152-0.

[27] Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, K. Aditya, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, Jessica Wu, Joe Nudell, Joel Niklaus, John J. Nay, Jonathan H. Choi, Kevin Tobia, Margaret Hagan, Megan Ma, Michael A. Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt, Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams,

Sunny Gandhi, Tom Zur, Varun Iyer, and Zehua Li. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/89e44582fd28ddfea1ea4dcb0ebbf4b0-Abstract-Datasets%5C_and%5C_Benchmarks.html.

[28] Ali Hakimi Parizi, Yuyang Liu, Prudhvi Nokku, Sina Gholamian, and David Emerson. A comparative study of prompting strategies for legal text classification. In Daniel Preo□iuc-Pietro, Catalina Goanta, Ilias Chalkidis, Leslie Barrett, Gerasimos Spanakis, and Nikolaos Aletras, editors, *Proceedings of the Natural Legal Language Processing Workshop 2023*, pages 258–265, Singapore. Association for Computational Linguistics, December 2023. DOI: 10.18653/v1/2023.nllp-1.25. URL: https://aclanthology.org/2023.nllp-1.25/.

[29] Peter Henderson, Mark S. Krass, Lucia Zheng, Neel Guha, Christopher D. Manning, Dan Jurafsky, and Daniel E. Ho. Pile of law: learning responsible data filtering from the law and a 256gb open-source legal dataset. *CoRR*, abs/2207.00220, 2022. DOI: 10.48550/ARXIV.2207.00220. arXiv: 2207.00220. URL: https://doi.org/10.48550/arXiv.2207.00220.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[31] Quzhe Huang, Mingxu Tao, Zhenwei An, Chen Zhang, Cong Jiang, Zhibin Chen, Zirui Wu, and Yansong Feng. Lawyer llama technical report. *CoRR*, abs/2305.15062, 2023. DOI: 10.48550/ARXIV.2305.15062. arXiv: 2305.15062. URL: https://doi.org/10.48550/arXiv.2305.15062.

[32] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report. *CoRR*, abs/2409.12186, 2024. DOI: 10.48550/ARXIV.2409.12186. arXiv: 2409.12186. URL: https://doi.org/10.48550/arXiv.2409.12186.

[33] Agnieszka Jablonowska, Francesca Lagioia, Marco Lippi, Hans-Wolfgang Micklitz, Giovanni Sartor, and Giacomo Tagiuri. Assessing the cross-market generalization capability of the CLAUDETTE system. In Erich Schweighofer, editor, *Legal Knowledge and Information Systems - JURIX 2021: The Thirty-fourth Annual Conference, Vilnius, Lithuania, 8-10 December 2021*, volume 346 of *Frontiers in Artificial Intelligence and Applications*, pages 62–67. IOS Press, 2021. DOI: 10.3233/FAIA210318. URL: https://doi.org/10.3233/FAIA210318.

[34] Deepali Jain, Malaya Dutta Borah, and Anupam Biswas. Summarization of legal documents: where are we now and the way forward. *Computer Science Review*, 40:100388, 2021. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2021.100388. URL: https://www.sciencedirect.com/science/article/pii/S1574013721000289.

[35] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics, July 2019.

DOI: 10.18653/v1/P19-1356. URL: https://aclanthology.org/P19-1356/.

[36] F. Jelinek. *Self-organized language modeling for speech recognition*. In *Readings in Speech Recognition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990, pages 450–506. ISBN: 1558601244.

[37] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023. DOI: 10.48550/ARXIV.2310.06825. arXiv: 2310.06825. URL: https://doi.org/10.48550/arXiv.2310.06825.

[38] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback. *CoRR*, abs/2312.14925, 2023. DOI: 10.48550/ARXIV.2312.14925. arXiv: 2312.14925. URL: https://doi.org/10.48550/arXiv.2312.14925.

[39] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL: http://arxiv.org/abs/1412.6980.

[40] Francesca Lagioia, Agnieszka Jabłonowska, Rūta Liepiņa, and Kasper Drazewski. Ai in search of unfairness in consumer contracts: the terms of service landscape. *Journal of Consumer Policy*, 45:481–536, 2022. URL: https://api.semanticscholar.org/CorpusID:250657710.

[41] Jinqi Lai, Wensheng Gan, Jiayang Wu, Zhenlian Qi, and Philip S. Yu. Large language models in law: A survey. *AI Open*, 5:181–196, 2024.

DOI: 10.1016/J.AIOPEN.2024.09.002. URL: https://doi.org/10.1016/j.aiopen.2024.09.002.

[42] Harvard law school. Caselaw access project. URL: https://case.law/.

[43] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics, 2020. DOI: 10.18653/V1/2020.ACL-MAIN.703. URL: https://doi.org/10.18653/v1/2020.acl-main.703.

[44] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6391–6401, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/a41b3bb3e6b050b6c9067c67f663b915-Abstract.html.

[45] Marco Lippi, Przemysław Pałka, Giuseppe Contissa, Francesca Lagioia, Hans-Wolfgang Micklitz, Giovanni Sartor, and Paolo Torroni. Claudette: an automated detector of potentially unfair clauses in online terms of service. *Artif. Intell. Law*, 27(2):117–139, June 2019. ISSN: 0924-8463. DOI: 10.1007/s10506-019-09243-2. URL: https://doi.org/10.1007/s10506-019-09243-2.

[46] Shuang Liu, Baiyang Zhao, Renjie Guo, Guozhu Meng, Fan Zhang, and Meishan Zhang. Have you been properly notified? automatic compliance analysis of privacy policy text with GDPR article 13. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia, editors, *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 2154–2164. ACM / IW3C2, 2021. DOI: 10.1145/3442381.3450022. URL: https://doi.org/10.1145/3442381.3450022.

[47] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[48] Zhu Liu, Cunliang Kong, Ying Liu, and Maosong Sun. Fantastic semantics and where to find them: investigating which layers of generative llms reflect lexical semantics. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 14551–14558. Association for Computational Linguistics, 2024. DOI: 10.18653/V1/2024.FINDINGS-ACL.866. URL: https://doi.org/10.18653/v1/2024.findings-acl.866.

[49] Antoine Louis, Gijs van Dijck, and Gerasimos Spanakis. Interpretable long-form legal question answering with retrieval-augmented large language models. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 22266–22275. AAAI Press, 2024.

DOI: 10.1609/AAAI.V38I20.30232. URL: https://doi.org/10.1609/aaai.v38i20.30232.

[50] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: overcoming few-shot prompt order sensitivity. In *ACL (1)*, pages 8086–8098. Association for Computational Linguistics, 2022.

[51] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *J. Mach. Learn. Res.*, 24:253:1–253:15, 2023. URL: https://jmlr.org/papers/v24/23-0069.html.

[52] Alan Lyra, Carlos Barbosa, Herbert Salazar, Matheus Argôlo, Yuri Lima, Rebeca Motta, and Jano Moreira de Souza. Automatic transcription systems: a game changer for court hearings. In *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KMIS*, pages 163–174. INSTICC, SciTePress, 2024. ISBN: 978-989-758-716-0. DOI: 10.5220/0012891400003838.

[53] José B. Mariño, Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, José A. R. Fonollosa, and Marta R. Costa-jussà. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549, December 2006. ISSN: 0891-2017. DOI: 10.1162/coli.2006.32.4.527. URL: https://doi.org/10.1162/coli.2006.32.4.527.

[54] Andrey Andreyevich Markov. An example of statistical investigation of the text eugene onegin concerning the connection of samples in chains. *Bulletin de l'Académie Impériale des Sciences de St.-Pétersbourg*:153–162, 1913.

[55] Thomas J. Maronick. Do consumers read terms of service agreements when installing software? - a two-study empirical analysis. *International Journal of Business and Social Research*, 4(6):137–145, 2014. URL: https://EconPapers.repec.org/RePEc:mir:mirbus:v:4:y:2014:i:6:p:137-145.

[56] Eric Martinez. Re-evaluating gpt-4's bar exam performance. *Artificial Intelligence and Law*, 2024. DOI: 10.1007/s10506-024-09396-9. URL: https://doi.org/10.1007/s10506-024-09396-9.

[57] Aleecia M. McDonald and Lorrie Faith Cranor. The cost of reading privacy policies. In 2009. URL: https://api.semanticscholar.org/CorpusID:197633124.

[58] Hans Wolfgang Micklitz, Przemyslaw Palka, and Yannis Panagis. The empire strikes back: digital control of unfair terms of online services. *Journal of Consumer Policy*, 40:367–388, 2017. URL: https://api.semanticscholar.org/CorpusID:157362840.

[59] Kensuke Nakamura, Stefano Soatto, and Byung-Woo Hong. Stochastic batch size for adaptive regularization in deep network optimization. *Pattern Recognit.*, 129:108776, 2022. DOI: 10.1016/J.PATCOG.2022.108776. URL: https://doi.org/10.1016/j.patcog.2022.108776.

[60] Ha-Thanh Nguyen, Randy Goebel, Francesca Toni, Kostas Stathis, and Ken Satoh. Black-box analysis: gpts across time in legal textual entailment task. *CoRR*, abs/2309.05501, 2023. DOI: 10.48550/ARXIV.2309.05501. arXiv: 2309.05501. URL: https://doi.org/10.48550/arXiv.2309.05501.

[61] Jonathan A. Obar and Anne Oeldorf-Hirsch. The biggest lie on the internet: ignoring the privacy policies and terms of service policies of social networking services. *Information, Communication & Society*, 23(1):128–147, 2020. DOI: 10.1080/1369118X.2018.1486870.

eprint: https://doi.org/10.1080/1369118X.2018.1486870. URL: https://doi.org/10.1080/1369118X.2018.1486870.

[62] Alexandre Quemy and Robert Wrembel. Echr-od: on building an integrated open repository of legal documents for machine learning applications. *Information Systems*:101822, 2021. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2021.101822. URL: https://www.sciencedirect.com/science/article/pii/S0306437921000636.

[63] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. In 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.

[64] Federico Ruggeri, Francesca Lagioia, Marco Lippi, and Paolo Torroni. Detecting and explaining unfairness in consumer contracts through memory networks. *Artificial Intelligence and Law*, 30:59–92, 2021. URL: https://api.semanticscholar.org/CorpusID:235408382.

[65] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. URL: https://api.semanticscholar.org/CorpusID:205001834.

[66] Abel Salinas and Fred Morstatter. The butterfly effect of altering prompts: how small changes and jailbreaks affect large language model performance. *CoRR*, abs/2401.03729, 2024.

[67] Sainbayar Sukhbaatar, arthur szlam arthur, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/8fb21ee7a2207526da55a679f0332de2-Paper.pdf.

[68] Supreme court database. URL: http://scdb.wustl.edu/.

[69] Chenhao Tang, Zhengliang Liu, Chong Ma, Zihao Wu, Yiwei Li, Wei Liu, Dajiang Zhu, Quanzheng Li, Xiang Li, Tianming Liu, and Lei Fan. Policygpt: automated analysis of privacy policies with large language models. *CoRR*, abs/2309.10238, 2023. DOI: 10.48550/ARXIV.2309.10238. arXiv: 2309.10238. URL: https://doi.org/10.48550/arXiv.2309.10238.

[70] Microsoft Phi-3 team. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, abs/2404.14219, 2024. DOI: 10.48550/ARXIV.2404.14219. arXiv: 2404.14219. URL: https://doi.org/10.48550/arXiv.2404.14219.

[71] Mistral AI team. Codestral, 2024. URL: https://mistral.ai/news/codestral/.

[72] Mistral AI team. Mistral nemo, 2024. URL: https://mistral.ai/news/mistral-nemo/.

[73] Terms of service; didn't read. URL: https://tosdr.org/en.

[74] Dietrich Trautmann. Large language model prompt chaining for long legal document classification. *CoRR*, abs/2308.04138, 2023. DOI: 10.48550/ARXIV.2308.04138. arXiv: 2308.04138. URL: https://doi.org/10.48550/arXiv.2308.04138.

[75] Dietrich Trautmann, Natalia Ostapuk, Quentin Grail, Adrian Alan Pol, Guglielmo Bonifazi, Shang Gao, and Martin Gajek. Measuring the groundedness of legal question-answering systems. *CoRR*, abs/2410.08764, 2024. DOI: 10.48550/ARXIV.2410.08764. arXiv: 2410.08764. URL: https://doi.org/10.48550/arXiv.2410.08764.

[76] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, Yasemin Altun, and Yoram Singer. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(9), 2005.

[77] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[78] Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. Label words are anchors: an information flow perspective for understanding in-context learning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9840–9855. Association for Computational Linguistics, 2023. DOI: 10.18653/V1/2023.EMNLP-MAIN.609. URL: https://doi.org/10.18653/v1/2023.emnlp-main.609.

[79] Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N. Cameron Russell, Thomas B. Norton, Eduard H. Hovy, Joel R. Reidenberg, and Norman M. Sadeh. The creation and analysis of a website privacy policy corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. DOI: 10.18653/V1/P16-1126. URL: https://doi.org/10.18653/v1/p16-1126.

[80] Fangyi Yu, Lee Quartey, and Frank Schilder. Exploring the effectiveness of prompt engineering for legal reasoning tasks. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13582–13596. Association for Computational Linguistics, 2023. DOI: 10.18653/V1/2023.FINDINGS-ACL.858. URL: https://doi.org/10.18653/v1/2023.findings-acl.858.