Universiteit Leiden

TNO

# Master Computer Science

Legal question-answering in the age of large language models


| | |
|---|---|
| Name: | Jonathan Strijker |
| Student ID: | s1455141 |
| Date: | 12/04/2024 |
| Specialisation: | Data Science |
| 1st supervisor: | Prof. dr. Suzan Verberne |
| 2nd supervisor: | Dr. Zhaochun Ren |


Master's Thesis in Computer Science


Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

**Abstract**

This research explores the capabilities of current state-of-the-art large language models (LLMs) at answering legal questions. We initialized the project by creating a dataset of legal questions and corresponding answers that we scraped from Dutch law exams found online. As 'legal questions' can cover a wide range of legal topics, we have decided to slightly narrow our scope by including only questions regarding **Dutch Civil Law** into the dataset. The dataset is divided into open-ended and multiple-choice questions.

The question-answering (QA) models that we test, are the closed-source **GPT-3.5-turbo** and **GPT-4** models developed by OpenAI. In our experiments, we test several prompting methods; most notably: zero-shot prompting, few-shot prompting, zero-shot Chain-of-Thought (CoT) prompting, and Retrieval Augmented Generation (RAG). As legal questions typically require reasoning over statutory articles, the RAG method aims to first retrieve the relevant articles to a question from a (vector) database, after which they are prepended to the question in the prompt. As we have faced many difficulties in implementing the IR component of RAG, we evaluate it in the 'oracle' setting.

Several metrics have been explored to evaluate the performance of the models on answering the open-ended questions, but we struggled to find automated metrics that are on par with human evaluation. Besides the standard QA evaluation metrics, that are based on word overlap and semantic similarity, we also look at the precision and recall of models in regards to referring to the *relevant* statutory articles in their answers. In general, we have found substantial performance differences across the different prompting methods. In that respect, we generally recommend the use of the more advanced prompting methods that we have tested over the basic zero-shot prompting setting in legal QA.

To conclude, we hope this work may provide the reader with helpful tips, pitfalls to avoid, and general recommendations for any related future work in legal QA regarding Dutch Civil Law.

# Acknowledgements

I would like to take this moment to express my gratitude to all who helped me realize this thesis.

First and foremost, I would like to thank my academic supervisor Suzan Verberne for all her guidance, support, and expertise throughout the process of writing my thesis. Second, I would like to thank my second academic supervisor Zhaochun Ren for his valuable feedback during the last stages of my thesis. Furthermore, I would like to thank TNO for providing me the opportunity to write my thesis with them. A special thanks to: Anne Merel Sternheim, Eveline Kalff, and Konstantinos Sismanidis, for their help during my time there.

Lastly, my sincere thanks to my family and friends, and all others who displayed interest in my thesis progress.

# Contents

# 1 Introduction

The year 2023 was an eventful one in the world of Artificial Intelligence (AI). It was the year that *generative* AI really managed to capture the attention of the general audience.[1] In generative AI, models are trained to generate data; usually in the form of text, pictures, or sound. As a result of the ever-continuing progress in the field, this generated data has become increasingly indistinguishable from data produced by humans. We have all seen the rise of AI models capable of generating hyper-realistic pictures (deepfakes) in the last decade, but this year the eyes are on a different type of generative model; one that focuses on generating **text**.

Text-generating models are nothing new: *N-gram language models* have been used for text-generating purposes for ages [1]. However, their success pales in comparison to the text-generating models that are used today as the development of the transformer architecture [2] has led to much better predictive models. By now most people in the world have, in some way or another, heard about the wonders of **ChatGPT** and its applications. Writing your school assignment, or generating some catchy slogan for a new product; it is all within the capabilities of a generative *Large Language Model* (LLM). And although such models have been around since 2018 [3], they have become much bigger and more powerful over the years; acquiring abilities far beyond simple text generation [4]. However, it seems that knowledge of their existence has only seemed to take off after the release of ChatGPT by OpenAI in November 2022.[2]

The release of ChatGPT and its subsequent success marked the start of a race between the tech giants. *Google*, *Meta*, and *Amazon* are all examples of companies that wanted a piece of the pie and thus started accelerating the development of their own LLMs. These models, developed by large companies, are called **foundation models**, and they have been *pre-trained* on large amounts of texts using substantial computational resources [5]. Often these pre-trained models are then *fine-tuned* such that they can be utilized for different downstream applications. For instance, ChatGPT is a fine-tuned version of the base model *GPT-3.5*, making it better suited for chatbot applications.[3]

Despite the huge diversity of generative LLMs we see today, all are based on the same **transformer-based** backbone [6]. Inherent to this technology come the same set of challenges terrorising all generative LLMs. One particular tough challenge being the phenomenon of **hallucination**; the generation of content that is "nonsensical or unfaithful to the provided source content" [7].

A lot of research today goes into tackling or mitigating hallucinations, but finding a solution is no easy feat. One popular approach involves the use of *Reinforcement Learning from Human Feedback* (RLHF) in which models are fine-tuned based on human preferences [8]. Besides reducing hallucinations, this method is one of the main drivers for reducing toxic output in LLMs [8]. A second important approach involves formatting the prompts in a specific way to improve the generated output; this method is called **prompt engineering** [9]. A typical example of a prompt engineering technique is called *in-context learning* [4], and it revolves around including task examples as demonstrations into the prompts. Lastly, another approach that can be used to reduce hallucination, is to 'ground' generative LLMs

[1] `https://nos.nl/l/2473328`
[2] `https://nos.nl/l/2469655`
[3] `https://help.openai.com/en/articles/6783457-what-is-chatgpt`

with knowledge such that it is elicited to base its generated answer on that knowledge. This method is called **Retrieval-Augmented Generation** (RAG) as it combines the field of IR with the field of text generation [10]. The simplest implementation that encapsulates this, is an LLM that when given a query: first tries to retrieve relevant documents, after which it takes in both the query and relevant documents as input. Many more methods are constantly being proposed to tackle hallucinations. One notable method involves the implementation of self-verification mechanisms in LLMs, e.g.: `Chain-of-Verification` (CoVe) [11].

So far, OpenAI has had great success in using RLHF for improving their models as arguably the most successful LLM around is theirs (GPT-4) [12]. The model is in fact so good, that in March of 2023, a paper was released reporting that GPT-4 had passed the notorious American 'Bar Exam', and that it significantly outperformed human test-takers and prior models in doing so [13]. This was all done in a **zero-shot prompting** setting (simply providing the question to the model and asking for the result). This achievement evidently sparked my interest as a student of both law and computer science. As I was not aware of any similar previous research done for the **Dutch legal domain** at the start of my thesis, it seemed like an interesting topic to do research on. Consequently, tackling legal question-answering with LLMs became the basis of this thesis.

## 1.1 Research questions

Given the online availability of old law exams designated to test the knowledge of Dutch law students [4], it seemed feasible to create a well-grounded dataset with legal questions. Thus, a complimentary goal of this project is to collect and curate a dataset with legal questions and answers that could be used for future research.

Knowing that legal scholars often need statutory norms to answer legal questions, it feels logical to enable an LLM to have access to this information as well; therefore opening up the possibility of augmenting LLMs with a **retrieval component** specifically used to retrieve those types of documents.

Simultaneously, in light of the rapid emergence of **different prompting techniques**[5], we hope to try out their effectiveness for our question-answering problem in the Dutch legal domain. We believe that the use of these novel techniques leads to better results over basic zero-shot prompting.

By researching the effectiveness of both these enhancements, we hope to ultimately improve both the **performance and explainability** (why does a model answer how it answers) of current state-of-the-art LLMs. In this research, we aim to find answers to the following research questions:

- How successful are current large language models at answering questions regarding Dutch Civil Law?
  - What methods can be used to at forehand retrieve the relevant statutory articles to these questions?
  - Given the relevant statutory articles to a legal question as context, to what extent are large language models able to provide the correct answer?
  - What prompting techniques can be used to improve their ability to answer these questions?
  - What aspects should be evaluated, and what evaluation metrics should be used, in order to objectively evaluate their performance?

[4] https://www.studeersnel.nl/nl

[5] https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/

# 2 Background

In this section, we will be going over all previous work related to our research. We will start by describing the heart of LLMs; the transformer architecture and its workings. Afterwards, we will give an introduction to the field of IR and how techniques from the field can be used to retrieve legal data. Subsequently, we will describe our main objective; tackling the problem of legal Question-Answering (QA). Furthermore, we will conclude by describing the background on LLMs, as well as previous work in which LLMs have been used for legal QA.

## 2.1 Transformers

As previously mentioned, the transformer is the backbone of LLMs, and this neural network architecture was first proposed in the paper "Attention is all you need" [2] by researchers at Google in 2017. This original transformer was a **sequence-to-sequence** model that was created to tackle sequence modeling/transduction problems, such as machine translation. Before the rise of the transformer, these types of problems were historically tackled with Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). However, for the task of machine translation, these architectures were outperformed by the transformer in both translation quality and training cost [2][14].

The transformer uses the encoder-decoder framework; a framework that has been proposed for machine translation around 2014 [15]. In comparison to early architectures, the transformer however introduces some important **enhancements**. As explaining every part of the transformer architecture is beyond the scope and focus of this thesis, we will give a 'general' overview of its workings in the following paragraphs. An illustration of the architecture can be seen in Figure 1. We will now go over this architecture step-by-step (see Section 2.1.1).

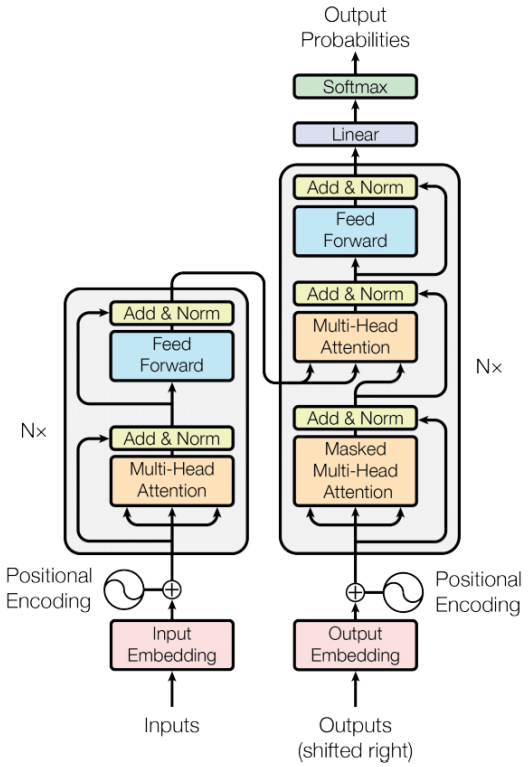


Figure 1: The transformers architecture [2]

### 2.1.1 Encoder

On the left side of the architecture in Figure 1, we see the encoder part of the architecture.

Given a sequence of words, or a sequence of **tokens** as they are called in this context, as input; the first step for the encoder is to transform these tokens into **token embeddings**. These token embeddings are simply large vectors/arrays of numbers that act as a high-dimensional representation of the corresponding token. This transformation is needed because neural networks do not

directly take words as input. However, during this process, the order of the token sequence is lost. Therefore, one had to find a way to add positional information to the created embeddings [2][14].

The solution was to add **positional encodings** to the input embeddings. These positional encodings have the same dimension as the input embeddings, such that the two can be summed [2]. The positional encodings use sine and cosine functions to encode the positional information. For more information on how this works, we refer the reader to the original paper by Vaswani et al. [2].

The key innovation of the transformer comes up next in the pipeline: the **self-attention** [1] blocks. At the core of self-attention lies the ability to compare every word with all the other words in the input, and assign those relationships a relevance score in the current context. The simplest method to compare word embeddings is to calculate their dot product. Subsequently, self-attention builds on this principle but in a complex manner [1].

The self-attention block is defined by a stack of stack of $N$ identical layers. These layers consist of 'multi-head self-attention' and 'position-wise fully connected feed-forward' sublayers. Furthermore, residual connection [16] and normalization layers [17] are employed after both sublayers [2]. The multi-head self-attention layer in the transformer model uses '**scaled dot-product attention**' which takes as input the three matrices: $Q$, $K$, and $V$, after which it computes the matrix of outputs as follows [2]:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

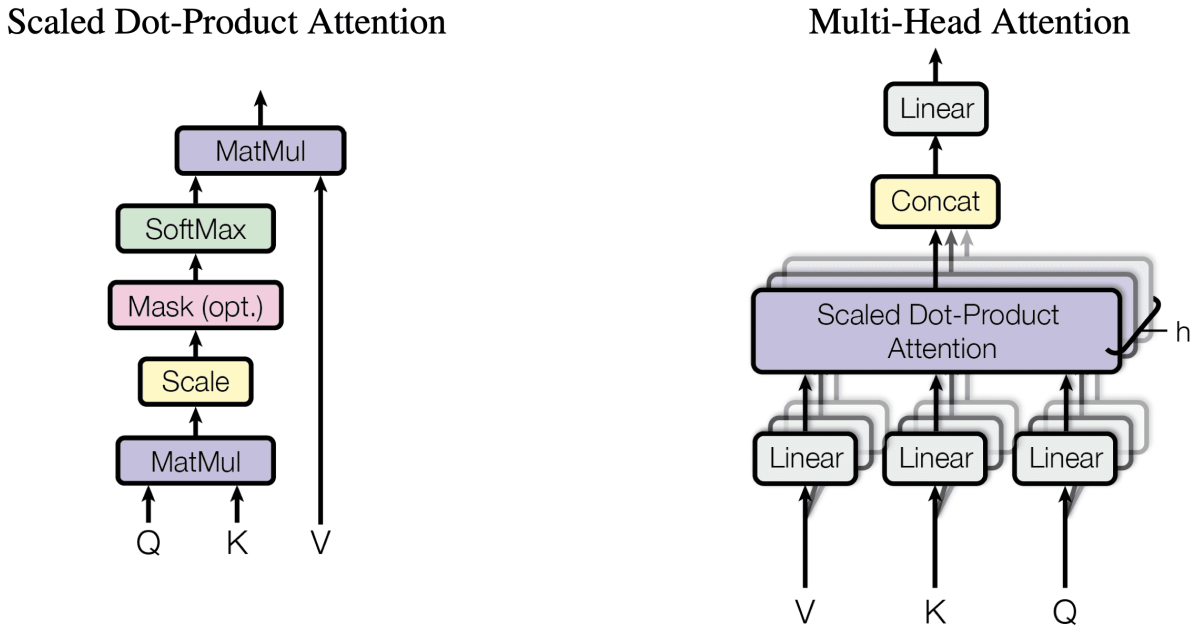


Figure 2: Scaled dot-product attention & multi-head attention [2]

On the left side of Figure 2, one can see the calculation of scaled dot-product attention visualized. The idea is that all token embeddings are turned into query, key, and value embeddings by doing matrix multiplication with their corresponding (trainable) weight

matrices $W^Q$, $W^K$, $W^Q$ [1]. The matrix multiplication between $Q$ and $K$ represents the comparisons between all token embeddings in the input with each other. Subsequently, after scaling and applying a softmax function to the resulting matrix, we multiply the softmax outputs with the $V$ matrix as it represents the 'value' per token; the intuition here is to "drown-out irrelevant words" [18]. This whole process is done $h$ times in parallel as seen on the right side of Figure 2, with each attention 'head' using its own set of weight matrices. Subsequently, the $h$ outputs are concatenated and passed to a learnable linear layer in order to reduce the final embedding to the original dimension size. This approach is called '**multi-head attention**' [2].

Subsequently, the output embedding goes via residual connection and normalization layers to the feed-forward layer. This layer consists of two learnable linear transformations with a ReLU activation function in between. Lastly, the output goes through another residual connection and normalization layer before going into the decoder part of the transformer [2].

### 2.1.2 Decoder

The decoder part of the transformer (on the right side of Figure 1) has sublayers similar to the encoder part, but it also has a few extra sublayers. At the start of the decoder part, we see that a **masked** multi-head attention layer was employed. This layer is similar to the regular multi-head attention layer except for the additional masking operation in the calculation of the scaled dot-product attention (see Figure 2). This additional layer restricts self-attention to only be calculated between a word and a preceding word. Put simply, the decoder is restricted from looking ahead when predicting the next word in a sequence; this kind of self-attention is often referred to as *auto-regressive attention* [19]. Lastly, we see that at the end of the decoder, a learned linear transformation and softmax function are used to convert the embeddings to the 'predicted next-token probabilities' [2].

### 2.1.3 Training and inference

The model's workings differ a little bit during training and doing inference. During the training of the transformer architecture, the output embedding consists of the embedded target output sequence. The target output sentence, however, is shifted one position to the right with a special start symbol (<*start*>) at position 0. Given this ground truth target output sequence, the parameters of the model are then tuned using gradient-based optimization.

During inference, the target output sentence is naturally not available, thus the initial output sentence starts with just the start symbol. Subsequently, after a pass through the decoder, the next most probable word is appended to the sentence. This new sentence is then iteratively fed to the decoder until the sequence is complete and the 'end of sentence' token (<*EOS*>) has been appended to the target output sentence [18].

Despite many parts of the transformers being parallelizable, one fundamental issue of the transformer remains the **quadratic complexity** with respect to the length of the input when calculating self-attention. It significantly limits the length of any input sequence as it makes *training* expensive [19][18]. Consequently, one might opt to train only on short texts, but this has a considerable downside. If a transformer has never seen a long sequence during training, it will not perform well on long sequences during inference; **extrapolation** to unseen sequence lengths is a challenge [20]. Nevertheless, many techniques have emerged that try to

make the transformer work around the quadratic complexity [21][22][23] and the extrapolation problem [24].

### 2.1.4 Encoder-only architectures

Up until now, we have discussed the original transformer architecture in its entirety (encoder-decoder). However, following its release in 2017, derived models have emerged that only use the encoder or decoder part of the architecture. In October 2018, an encoder-only model named **BERT** was released that reached state-of-the-art performances on many `Natural Language Understanding` (NLU) tasks like text classification and sequence labelling [19][25]. BERT, an abbreviation for `Bidirectional Encoder Representations from Transformers`, is a **pre-trained** model that is able to be **fine-tuned** for many natural language processing tasks [25]. This means that the model was first trained on a very large collection of unlabeled data before being able to be fine-tuned on a much smaller collection of labeled task-specific data. This 'pre-training and fine-tuning' paradigm, an instance of transfer learning [1], inspired a large number of later works [6].

As the pre-training stage uses unlabeled data, the model has to create self-defined pseudo labels, which it then has to predict. We call this paradigm: `Self Supervised Learning` (SSL) [26]. In BERT, this pre-training was done using `Masked Language Modeling` (MLM) and `Next Sentence Prediction` (NSP). For the former objective, words in the input were masked randomly and the model had to predict the missing word based on the context. For the latter, the model had to predict if the last sentence in a self-sampled <*sentence, sentence*> pair was the actual next sentence [25]. Following the great success of BERT, other BERT-like models were released that introduced some modification to the model; for example by utilizing different pre-training objectives [27][28].

### 2.1.5 Decoder-only architectures

The BERT model works bidirectionally; any two words in a sentence can see each other. Consequently, this hinders its `Natural Language Generation` (NLG) ability [26] and so **decoder-only architectures** are typically used for this purpose. These architectures are commonly pre-trained using the `Language Modeling` (LM) task; which means that they have to **autoregressively**[6] predict the next tokens in a sentence given the preceding tokens in that sentence. Subsequently, these models can be fine-tuned to perform better on downstream tasks [6].

Today, there exists a wide range of decoder-only models; both open-source and commercial models. In literature, they are interchangeably referred to as e.g.: `Pre-trained Language Models` (PLMs) [29], `Large Language Models` (LLMs) [6], `Pre-Trained Models` (PTMs), `foundation models` [5] and `Pre-trained Foundation Models` (PFMs) [26]. Some of these definitions also include encoder-only and encoder-decoder models in their scope, while others might discriminate between smaller and larger models. Nevertheless, examples of these models include: the `Generative Pre-trained Transformer` (GPT) series [3][30][4][12], Palm [31], and Llama [32]. One of the most noteworthy developments in newer models is that they have been massively scaled as compared to the original

[6] "Incrementally generate words by repeatedly sampling the next word conditioned on our previous choices" [1]

transformer models, and this had some interesting implications. Thus the term '**Large Language Model**' (LLM) is typically used for these models [6].

### 2.1.6 Encoder-decoder architectures

For completeness, we would like to mention the existence of models utilizing both encoder and decoder in their architecture similar to that of the original transformer. This family of encoder-decoder models includes T5 [33] and BART [34]. These are sequence-to-sequence models that can perform both natural language understanding and generation; meaning we would be able to use them for tasks like question-answering [19]. However, they seem to be outperformed by decoder-only models such as GPT-3 on prompt-based inference (i.e., in-context learning) [35]. As a large part of this work revolves around testing different prompting techniques, using a decoder-only model for our text generation purposes makes more sense in that regard.

## 2.2 Information retrieval

To implement a retrieval component in our pipeline for legal QA, we will first go over the definition of information retrieval.

> "Information retrieval or IR is the name of the field encompassing the retrieval of all manner of media based on user information needs." [1]

Concretely, in IR we are typically looking to find a document $d$ from the $D$ documents in some collection (or 'corpus') that best matches a query $q$ [1]; this is called **ad hoc retrieval**. A document can be any unit of text [1]. For instance, within the scope of our defined problem, a document refers to statutory articles. Furthermore, the query is defined as a legal question we hope to find relevant statutory articles for.

### 2.2.1 Lexical retrieval

Our legal question and our documents are made up of words, referred to as **terms** in IR [1]. To find a relevant document in a collection, the most straightforward way to find relevant documents would be to retrieve documents with similar terms to the query; the **lexical** approach. A commonly used weighting scheme to implement this approach is called **TF-IDF**. It is defined as the product of the Term Frequency and the Inverse Document Frequency. The former metric represents how often a term from the query occurs in a document, the latter represents how rare this term is in the collection of documents; i.e. the amount of documents in the collection that contain this term [1].

We can calculate TF-IDF for all terms in the vocabulary against all documents in the collection. This can then be represented as a 'term-document matrix' in which the rows represent the terms and the columns represent the documents. A column will represent the respective document as a vector. Similarly, you can create a vector for the query.[7] Cosine similarity can then be used to measure the angle, or similarity, between the document vectors and the query vector. This approach in IR is called the **vector space model**. Since most of the elements in these vectors are zero, they are called **sparse** vectors [1][36].

[7] Elements in the query vector are 1 when the terms are in the query, otherwise 0.

A variant of TF-IDF called **BM25**, was the state-of-the-art retrieval approach for decades and remains a very robust baseline [37]. It is slightly different from TF-IDF, as it is a probabilistic model. Thus, instead of comparing queries and documents using vectors, it uses a formula from probability theory to do these comparisons [36].

However, lexical models are not without flaws. For example, they do not take into account the **order of terms**. This means that the two one-sentence documents: "man sues police" and "police sues man" would be the same in the eyes of such models. Because of this characteristic, they are also called *bag of words* models [36].

Lexical models are additionally called *exact term matching* models [38]. As the name suggests, they only match terms in documents that exactly match the terms in the query. This means that synonyms and ambiguous terms are obvious problems for them. As for synonyms, a query term like 'judgement' would not match the term 'decision' in a document, even though they are both used to describe the outcome of a legal case. This problem is referred to as the **vocabulary mismatch problem** [1][39]. Furthermore, ambiguous terms could lead to a query containing the legal term 'bar' to match a document referring to a 'bar' in the context of a drinking establishment.

To allow for some flexibility in the exact term matching problem, several '**word normalization**' techniques can be applied in the pre-processing stage of an IR pipeline. Stemming and lemmatization are two commonly used examples of such techniques. Lemmatization tries to reduce terms with inflectional endings to their dictionary form, e.g.: ['am', 'are', 'is'] $\rightarrow$ 'be', while stemming simply strips suffixes from the end of words, e.g.: ['article', 'articles'] $\rightarrow$ 'article' [40][1]. These techniques are examples of 'lossy compression' techniques, as they do not preserve all information. The assumption when using these techniques is that this loss of information does not affect the performance of the IR system [36]. It is outside the scope of this thesis to research the actual significance of this effect. It is not unthinkable that the loss of nuanced differences in the input could be extra hurtful in the legal domain, as opposed to other domains. Furthermore, the spelling correction and case folding (removing capitalization) of terms are also examples of commonly applied pre-processing techniques in IR [36].

Stopword removal, the removal of high-frequency words from both queries and documents, is something that can also be used to improve IR performance. However, seeing as the IDF weighting in lexical models already penalizes such terms, it is not very common to use [1].

Furthermore, several more advanced techniques have been proposed to improve the lexical models. Two notable techniques used specifically to tackle the vocabulary mismatch problem are **query expansion** and **document expansion** [41]. The idea behind this is to augment queries and documents with additional terms to bring relevant query-document pairs more in 'alignment' [41]. In the case of extraneous terms also being removed, 'rewriting' might be a more appropriate notion. Historically, it has been found that the use of query expansion generally increases recall (see Section 4.4.1) [36]. In addition, another approach similar to document expansion is the automatic adjustment of term weights for documents such that the importance of terms matches their weight [41].

In terms of practicality, 'document expansion' and 'document term reweighing' have been found to be simple and effective [41]. In contrast to the earlier findings [36], query expansion has received more mixed results [41].

### 2.2.2 Dense retrieval

In the previous section, we talked about the vector space model using sparse vectors. One of the problems with these models is that they depend on exact term matching, and are thus affected by the vocabulary mismatch problem [1]. Two synonyms like 'car' and 'automobile' would not have the same representation using sparse vectors however, it would make more sense to assign them a similar representation. The process of learning useful representations from text is called *representation learning* [1]. In order to capture the meaning of words, we would have to look at the context of words (surrounding words).

This is where dense representations come in; representations that are able to capture semantic matches, or *meaning*. When using dense vectors to represent text, we are talking about using '**embeddings**' [1]. Early methods to create embeddings were word2vec (2013) [42] and GloVe (2014) [43]. These created *static* word embeddings to represent words; this means that for every word, there is one vector representation. Naturally, this did not solve the problems with homonymy and polysemy. Given the example of the last section: the legal term 'bar' would still share the same vector representation as a 'bar' in the context of a drinking establishment.

Subsequently came the contextualized word embeddings, these were made to capture the complex characteristics of language as well as how meanings vary across linguistic contexts [41]. One early method was ELMo (2018) [44], which used a bidirectional LSTM network in its architecture. However, the real revolution started with the release of **BERT** to capture contextualized word embeddings. BERT was based on the highly parallelizable transformer architecture (see Section 2.1.4) [6]. In the years after its release, many architectures were released that were based on BERT e.g.: RoBERTa [27] and SpanBERT [45].

One of the first implementations of BERT for IR was in 2019 where it was used for the task of 'query-based passage re-ranking' [46]. This was done in the context of a **multi-stage retrieval** pipeline where BM25 was used to first retrieve a large number of candidate documents/passages, after which BERT would re-rank them based on relevance to the query [46]. These multi-stage retrieval/ranking pipelines are now typical for modern search systems, and they often deploy a term-based model like BM25 in the first stage [47].

The re-ranking process involved concatenating every query-document pair before passing it to the BERT model. Its output would then yield the probability score of the document being relevant to the query. All documents were then re-ranked with respect to these probabilities [46]. This methodology of concatenating multiple inputs into a single sequence and passing it to a model to get a relevance score is called the **cross-encoder** approach [41]. Another example of this cross-encoder approach can be seen in a pairwise ranking model named duoBERT[8] [48]. It concatenates the query and two different documents into a single sequence in order to estimate the relevance of a document *relative* to another document [41].

A completely different approach was introduced in 2019 called Sentence-BERT, or **SBERT** for short [49]. It introduces the following methodology: one takes two BERT models, one to create a query embedding and the other to create document embeddings. After creating the embeddings, one can then take the dot product between a query vector and a document vector to generate a similarity score. The documents with the highest scores can then be selected for retrieval. This type of double (Siamese) network architecture is also called the

[8] This contrasts with the pointwise ranking approach of a 'monoBERT' model [41].

**bi-encoder** architecture. It is computationally far more efficient than the cross-encoder architecture as creating the document embeddings can be done offline [41].

It has been found that cross-encoders generally perform better than bi-encoders. However, the latter is computationally far more efficient [41]. Something to note is that these dense retrieval techniques are often ineffective when directly applied to texts of different domains and different types of queries than the models were trained on. These types of queries and documents are considered '**out of distribution**". In such situations, BM25 performs better overall [41]. Lastly, it was found that **hybrid** models, which combine sparse and dense methods, usually exhibit higher effectiveness than their individual components [41].

Choosing the right model for your IR needs can be difficult as it also depends on the problem you are facing. In the case you want to implement a retrieval pipeline where your query and your documents are of similar lengths, you are dealing with *symmetric* search. However, if you have a short query and longer documents, the search is usually defined as *asymmetric* [50]. For the latter, one should pick a model trained on asymmetric search datasets like MS MARCO [51][50]. For the former, one should aim for models trained on symmetric search datasets. As we have mentioned before (see Section 2.1.3): extrapolation to unseen sequence lengths is a challenge for transformer-based models. In addition, most transformer-based models have a maximum input length of 512; longer input usually being truncated [41]. Consequently, it is crucial to analyze the **sequence lengths** of our legal questions and the legal sources to be retrieved. One should be aware that sequence lengths greater than a model's maximum input length can be truncated and that the average respective sequence lengths of queries and documents are important for choosing the right dense retrieval model.

*One new development is the rise of generative retrieval methods [52] (2022). However, their retrieval performance is yet to be proven superior on big datasets [53].*

### 2.2.3 Legal information retrieval

The application of IR to the legal domain is nothing new. Several IR tasks can be distinguished within legal IR, typically **case law retrieval** and **Statutory Article Retrieval** (SAR). Along with 'textual entailment', these tasks are part of the **COLIEE** curriculum; the annual Competition on Legal Information Extraction/Entailment. SAR involves the task of retrieving statute law articles relevant to a legal question [54], while case law retrieval involves finding past cases relevant to a query case [55].

In these problems lie extra difficulties for BERT-like retrieval models due to the specific **features of typical legal documents**, such as: "*long texts, mixing of formal abstract concepts with casual verbatims in layman language, or dealing with an unclear formulation of relevance*" [55]. A model that has been (pre-)trained only on generic corpora might not adapt well to legal IR tasks due to the "*difference between the generic language and the legalese language, a difference in word usage and meaning, information structuring, or sentence complexity*" [56][55].

As legal documents often consist of long texts, they usually exceed the standard maximum input length of 512 tokens in BERT-based models. To deal with this issue, previous work [57][58] has looked at the use of **Longformers** [22] in processing legal texts. Longformers are BERT-based models that use *sparse* self-attention mechanisms, as opposed to *full* self-attention mechanisms [19]. This modification reduces the computational complexity of the model; allowing for longer input sequences. One other notable approach to having

transformer-based architectures deal with long legal texts is the use **hierarchical transformer models** [54]. In hierarchical encoders, long input sequences are divided into segments. These segments are then passed through an encoder independently. Afterwards, the resulting embeddings are aggregated and sent through another encoder to get a single output representation. This approach is an example of a typical divide-and-conquer strategy [19].

Many recent works in legal IR use BERT-based models [59], which are typically pre-trained on general domain corpora [60]. This becomes a problem when these models have to adapt to very specific domains like the legal domain [60]. One problem is the difference in frequently used words between the specific and the general domain. Previous work [61] has tried to illustrate this difference by measuring the overlap between a generically pre-trained BERT model's vocabulary and the most commonly found tokens in the target corpora. To deal with this issue, one can choose to completely pre-train BERT from scratch or to further pre-train BERT from a checkpoint, using domain-specific corpora, to significantly improve the performance of the model in domain-specific tasks [60][62]. An example of such a model for the legal domain is **LEGAL-BERT** (English) [60].

As pre-training BERT from scratch is computationally very expensive [62], further pre-training can be preferred. Subsequently, such models also have to be fine-tuned for downstream tasks; requiring a lot of labeled data [41]. As labeled data for natural language processing tasks in the legal domain is notoriously hard to get [63][64], one can use **data augmentation** to acquire more data, e.g. by using 'synthetic query generation' [54].

So far, we have only discussed IR techniques that were based on leveraging the semantic similarity and/or lexical similarity between texts. However, it might be worth to also look at the other types of relationships between texts. Previous work in SAR involved the use of Graph Neural Networks (GNNs) to leverage the '**topological structure of legislation**' for retrieving statutes. In this paper, they first created dense statutory article embeddings that were later enriched with structural relationships; forming 'knowledge-rich embeddings' [54].

When implementing an IR system, one can optimize it using different objectives; one notable design choice is about balancing recall and precision (see Section 4.4.1). This choice should be based on how tolerant your system should be of false positives and false negatives. Historically, the focus on IR was on high recall [65]; i.e. all relevant documents are returned to the user. Currently, there are also sounds in legal IR that actually call for a high precision [66].

## 2.3 Legal question-answering

In the field of legal Question-Answering (QA), we are concerned with finding one or more answer(s) to a given legal question. Historically, two major paradigms exist for the task of question-answering called **open domain QA** and **knowledge-based QA** [1]. The latter uses *structured* data (*knowledge bases*) to derive its answers from. We will focus on the former; open domain QA, also referred to as **IR-based QA**.

In this paradigm, the dominant approach is to use a **retrieve and read** model where a search engine is used for the retrieve step, and a reading comprehension algorithm for the second step [1]. This reading comprehension algorithm is usually implemented in two ways: either using an *extractive* QA approach or using an *abstractive* QA approach. In the former, the answer is *in the span of text* in a retrieved passage. In the latter, the answer is not drawn *exactly* from the passage [1]; it could for example be reformulated. Using LLMs to generate answers to questions does not strictly fall under either umbrella. However, we do plan to implement a retriever in the LLM pipeline to retrieve relevant statutory norms we can give to an LLM as extra context, but this approach is called **Retrieval-Augmented Generation** (RAG) [67].

As we are dealing with legal QA, the environment is **closed**; only legal questions and legal sources are potentially retrieved and used. Moreover, we expect to primarily encounter questions in our dataset where **legal reasoning** over statutory articles is required. Thus, we have to assume that an LLM has this information *stored* in its parameters and in addition, that they are able to *correctly* reason according to that information.

Students of the law typically: follow lectures, read textbooks, analyze case law, and complete homework assignments, in order to prepare for exams. LLMs however, are typically (pre-)trained on extensive datasets comprising of books, web pages, Reddit data, Wikipedia, code (e.g. StackOverflow, Github) and other data [6]. A benefit of pre-training on data of a conversational nature (e.g. Reddit, StackOverflow) is that it has been speculated to potentially improve the performance of the model in question-answering tasks [6]. However, as LLMs are primarily trained on general corpora [68], their exposure to domain-specific data, such as Dutch law, is limited.

### 2.3.1 Legal domain adaptation

To deal with this issue for the Chinese legal domain, Chinese researchers have created the 'Lawyer LLaMa' model [68] which is a Chinese LLaMa model that has been continually pre-trained on Chinese legal data and then fine-tuned on both general tasks and 'legal' tasks. In their work, they have also experimented with RAG to retrieve relevant statutory articles in their LLM pipeline. One notable observation they have made might be very relevant to our research:

> "When we directly concatenate the retrieved legal articles and the user's question as new input, the model tends to quote all the provided legal articles in its response, without distinguishing whether they are truly relevant to the current scenario."

To deal with their retriever being imperfect, they added the following phrase to the end of their prompts:

> "There may be irrelevant articles in the reference articles, so please avoid quoting those unrelated ones when replying".

*This suggests that in realistic RAG implementations, prompt engineering (see Section 2.4.3) is something that remains necessary.*

### 2.3.2 Benchmarking legal capabilities

One of the most extensive works on assessing the legal capabilities of LLMs is the evaluation benchmark '**LawBench**' [69]. It consists of 20 different types of legal-related tasks under the Chinese civil law system. The tasks are classified into 3 different skill levels: **legal knowledge memorization**; **legal knowledge understanding**; and **legal knowledge applying**. They tested 20 multilingual LLMs, 22 Chinese-oriented LLMs and 9 (Chinese-oriented) legal-specific LLMs (including 'Lawyer LLaMa') in order to conclude that GPT-4 performs the best by a significant margin. This conclusion might not be very interesting, but their general findings are:

- They have found that most LLMs can not efficiently leverage retrieved article content in a RAG setting.
- The tested legal-specific LLMs do not necessarily outperform general LLMs.
- Increasing model size typically helps improve performance in one-shot settings, but in zero-shot settings, the results are mixed.
- (Legal) supervised fine-tuning may improve performance, but RLHF leads to a higher abstention rate (model abstains to answer).

They end their paper by stating that current LLMs (Sep 2023) are still **unable to give meaningful judicial aid** [69]. Similar extensive legal benchmarks have emerged for the English language, e.g.: 'LegalBench' [70] and 'LexGLUE' [71]. The LegalBench paper also affirms the superiority of the GPT-4 in legal tasks. In contrast to LawBench, they classify tasks based on: issue-spotting, rule-recall, rule-application, interpretation, and rhetorical understanding [70]. It seems this 'rule-recall' corresponds to the 'legal knowledge memorization' aspect in LawBench tasks. In both studies, it is the worst-performing aspect of LLMs; hinting that RAG approaches might help in overcoming this weakness. Something to note is that a problem of benchmarking LLMs using legal data is that due to regularly updated laws and new precedents, the data becomes outdated frequently [10].

Furthermore, previous work [72] has analyzed the ability of the older GPT-3 model to reason over simple *synthetic statutes*. This is an interesting approach as it is guaranteed that the LLM has not seen the statutory articles before. Other work [73] has found that legal multiple-choice questions require complex logical reasoning, *affirming* that models with a larger number of parameters usually perform better.

From the extensive 'LawBENCH' [69] paper, we observe that current open-source LLMs fine-tuned on legal-related tasks do perform better than their base models. However, they do not necessarily outperform the big commercial models like GPT-4 in such tasks; suggesting that we (performance-wise) should not use our limited labeled data to fine-tune our own open-source LLM. However, an alternative approach, which has gained prominence, is the '**pre-train, prompt and predict**' [74] paradigm, where LLMs are applied to new problems without fine-tuning the weights on the task [75].

## 2.4 Large language models

Research has found that **scaling** pre-trained language models (see Section 2.1.5) often leads to an increased model capacity [6]. Scaling in this context can refer to scaling the: number of model parameters, training dataset size, and the amount of training computations. This scaling gave rise to something called the "**emergent abilities**" of LLMs; formally defined as "abilities that are not present in small models but arise in large models" [6]. Such abilities were first discovered in the GPT-3 family [4], and they transition seemingly instantaneously from not present to present at seemingly unforeseeable model scales [76]. Three typical emergent abilities are: 'in-context learning', 'instruction following', and 'step-by-step reasoning'.

### 2.4.1 Emergent abilities

**In-context learning** revolves around giving an LLM a task description and/or a few task examples as demonstrations, and having the LLM be able to recognize and perform the new task without *explicit* gradient updates [6]. Its exact working mechanism is still unknown [77]. Subsequently, **instruction following** refers to LLMs being able to follow task instructions for new tasks without using explicit examples. This is achieved by the *instruction tuning* of models of a large enough scale ($> 8B$ parameters), and it involves fine-tuning such models with a mixture of multi-task datasets [6]. Lastly, **step-by-step reasoning** refers to a model being able to perform well on complex reasoning tasks by making use of *intermediate* reasoning steps before coming up with the final answer. It is speculated that the training on code is the reason for models obtaining this ability [78]. Moreover, the `Chain of Thought` (CoT) [79] prompting strategy is a method that can be used to elicit step-by-step reasoning in LLMs [6].

### 2.4.2 Hallucinations

Despite the amazing capabilities of current LLM models, some issues still persist. One notable problem in LLMs is their tendency to produce **hallucinations**: "texts that seem plausible but may be factually incorrect" [19]. The term 'confabulation' is occasionally used as well [80]. Current literature distinguishes between two types of hallucinations; **intrinsic** hallucinations and **extrinsic** hallucinations. In the former type, the generated output *contradicts* the source content, while the latter refers to generated outputs that cannot be *verified* from the source content. Something to note: these extrinsic hallucinations are not necessarily erroneous [7].

Both types of these hallucinations are unwanted, especially in the legal domain. For instance, early on in this research, we have found that an LLM would refer to non-existent statutory articles in its output when prompted with a legal question; an extrinsic hallucination. Several methods exist that aim to mitigate this problem of hallucination. Notably, the techniques of **prompt engineering** (see Section 2.4.3) and **retrieval-augmented generation** (see Section 2.3) can be used for this. Thus, by formatting the prompts in the most effective way possible, and including relevant legal sources into them, we hope to see if that improves the performance of LLMs in the legal QA problem. However, retrieving the relevant legal sources for a legal question is a problem on its own (see Section 2.2).

Lastly, LLMs tend to generate **harmful, biased, or toxic responses** [6], which just like the hallucination problem, is undesirable behaviour. The issue stems from LLMs capturing

unwanted characteristics from their pre-training data. Consequently, to more align the model's behaviour with human values: `Reinforcement Learning from Human Feedback` (RLHF) [8] has been the dominant approach to avert these issues [6].

### 2.4.3 Prompt engineering

When using LLMs that are not run locally, one has to make use of the APIs provided by the distributors of LLM services to make inference calls. Within the vast landscape of diverse APIs offered by different companies, several frameworks have emerged that have tried to provide a unified interface to all different LLMs. These *Language Model Programming* (LMP) frameworks as they are called also include several prompt templates and many other tools intended to create LLM-powered applications more easily. Examples of such frameworks include: LangChain[9], LlamaIndex[10], Microsoft's Semantic Kernel[11], and Stanford's DSPy [81]. Furthermore, most LMP frameworks introduce something called **prompt chaining**; "the principle of chaining LLM steps together, where the output of one step becomes the input for the next, thus aggregating the gains per step" [82]. With the use of such advanced techniques, one is able to create LLM-based **autonomous agents** [83]. At the moment, research into such agents is very active but the implementation is out of scope for this thesis.

Creating the right prompt is difficult, and there are several ways to create one. The most simple method of prompting is **zero-shot** prompting where a question simply goes in and an answer comes out. As we have seen from Section 2.4, we can also make use of a model's in-context learning ability by first providing some task demonstrations/examples to a model before inputting the actual question. This is called **few-shot** learning [4]. Neither of these methods have to be of a conversational nature, but depending on the model, multi-turn conversations can be implemented. Another common method is **Chain-of-Thought** (CoT) prompting [79]. The two major settings to use CoT in are **few-shot CoT** and **zero-shot CoT** [6]. In few-shot CoT, the task demonstrations/examples are accompanied by a chain of thought; a "series of intermediate natural language reasoning steps". Thus, few-shot CoT depends on providing examples with specific answer formats per task. In contrast, zero-shot CoT [84] is task agnostic. It simply consists of adding a phrase similar to "Let's think step by step" before each answer and using a second prompt that is similar to "Therefore, the answer is" to extract the answer in the correct format [84]. Previous work showed that CoT only has a positive effect on sufficiently large models (typically $> 10B$ parameter) and that it is only beneficial for tasks that require complex, or step-by-step, reasoning [6]. Moreover, it might actually be counterproductive for tasks that do not rely on complex reasoning.

### 2.4.4 Prompt enhancement techniques

Legal scholars are taught to answer legal questions using legal reasoning techniques such as **IRAC** (Issue, Rule, Application, Conclusion). It would be natural, to elicit LLMs to take the same approach. Hence, previous work has experimented with prompting techniques that are based on this. It has been found that prompts that are derived from such specific legal reasoning techniques, can outperform CoT prompting and fine-tuning approaches [85].

[9] `https://github.com/langchain-ai/langchain`
[10] `https://github.com/jerryjliu/llama_index`
[11] `https://github.com/microsoft/semantic-kernel`

Many more different prompting techniques have been explored in literature. We will give a brief overview of some of the most notable ones that we have found. Many **iterative prompting** techniques have been explored such as self-critique, self-refinement, self-feedback, self-reflection, and self-improvement [86]. The key idea of these techniques is to provide an initial prompt and to have the LLM iteratively prompt itself in order to improve his answer. One could also stack multiple LLMs together and have them debate, or 'iterative prompt', each other [87]. In addition, we also see prompting techniques that leverage the non-deterministic nature of LLMs by eliciting multiple answers per input and picking the final answer with a **majority vote** (e.g. self-consistency [88]). Lastly, we would like to mention the existence of techniques that expand upon, and/or use an ensemble of existing techniques; e.g. domain-specific CoTs [89], Tree of Thoughts (ToT) [90], and Ensemble Refinement (ER) [91].

Furthermore, it has been found that assigning a **role** to the LLM by adding a prefix "You are an expert on this task (or in this domain)" can boost the performance of LLMs [6], as well as specifying the **tone** the LLM should answer in at the end of a prompt, e.g. "Academic tone" [92]. In addition, we have found a plethora of other general prompting tips in the aforementioned literature; some contradicting one another, some very task-specific, and many of the same tenor. For general tips on prompting we refer the reader to Table 12 of the 'A Survey of Large Language Models' paper [6]. For tips specific to legal prompt engineering, we refer the reader to Section V of the 'ChatGPT Goes to Law School' paper[92].

As LLMs are shown to be sensitive to the prompt format, such that semantically similar prompts can have drastically different performances, previous work exists on how to automatically create and optimize the right prompt formats [93][94]. One specific method that we plan to explore in this thesis, is the Optimization by PROmpting (**OPRO**) [93] framework where one leverages LLMs themselves to optimize prompts (see Section 4.3).

*All in all, after reviewing literature focused on prompting, we feel to have gained a better understanding of what works and what does not. However, one thing to note is that all of the listed techniques are based on the English language. It is unclear how well they translate to the Dutch language. We know that adding the standard 'Let's think step by step' sentence to a question helps LLMs in answering English questions, but we do not know if its exact translation in Dutch has the same effect on LLMs in answering Dutch questions.*

# 3 Data

In this section, we will describe the data that we are working with. In light of our research questions, we will be working with **Dutch statutory laws** and a dataset of **Dutch legal questions**.

## 3.1 Legal questions

In this thesis, we define a legal question as a question that is related to the legal domain. Specifically, this means that the answer to the question should be derivable from applying the law to the facts in the question.

The legal questions in our dataset were extracted from approximately 60 final, mid-term, and practice exams[12] from the courses: '*Inleiding Burgerlijk Recht*', '*Verbintenissenrecht*', '*Arbeidsrecht*', '*Personen-familie- en erfrecht*', '*Goederenrecht*' and '*Onderneming en recht*'; all taught at *Leiden University*. These are all courses from the undergraduate Law degree programme (LL.B.) in Leiden and they all relate to the domain of **Dutch Civil Law**. The choice for this particular field of law is partly based on the availability of exams regarding that domain and partly based on the fact that the questions in those exams often require reasoning over statutory articles (enabling statutory article retrieval experiments). The exams were given to students over the years 2012-2023. Only those exams were picked for which the model answers were available. We naively assume the material in the exams to still be relevant and not outdated.

### 3.1.1 Pre-processing

The exams contain both open-ended questions and multiple-choice questions. Out of the 66 exams we have extracted a total of 404 open-ended questions and 223 multiple-choice questions. Only questions were included that were accompanied by a model answer. We excluded all questions and answers: involving **visual images**, that were **duplicates**, that were **incomplete**, that were **not about the law**, or that contained **tables**. As all collected exams were in PDF format, we had to transform the questions and answers into a more machine-readable format; i.e. CSV. However, during this extraction process, we encountered multiple issues that we had to deal with. We will go over them briefly.

During the extraction of questions and answers from the PDF files to CSV, we performed the following pre-processing steps (in no particular order) to create a more uniform dataset.

1. Removed white lines from questions and answers.
2. Stripping trailing and leading whitespaces.
3. Removed redundant text, e.g.: point indicators, question numbers, mentioned textbook sources, etc.
4. Converted all text to UTF-8 encoding.
5. Removed decoding/parsing errors, e.g. unusual hyphens due to line breaks.

[12] Downloaded from `https://www.studeersnel.nl/nl`

We have generally tried to keep manual adjustments to a minimum. For some exams, we had to first use Optical Character Recognition (OCR) software to extract the text. We acknowledge that the removal of white lines in questions and answers could affect the performance. In addition, cursive and bold texts have been normalized during extraction; also leading to a slight loss of information.

Lastly, we sometimes see *multiple* correct model answers for a particular question. As evaluating these types of questions would require the use of more complicated evaluation metrics, combined with the fact that they are relatively scarce, we have decided to exclude them. In addition, we exclude open-ended questions for which the model answer explicitly states that multiple explanations can be correct depending on the soundness of the arguments.

### 3.1.2 Open-ended questions

Specific to the open-ended questions, there were some extra issues we had to decide on. In line with the in-exact nature of the legal field, it is common to see questions or answers explicitly mentioning what articles, or other types of grounds, should *not* be used for the case at hand. As this information is not incorrect, we still tried to exclude such phrases from the model answers because it might skew the evaluation metrics we intend on using. Furthermore, we have seen very long questions, often involving a piece of a specific court case or a company's general terms. Given the maximum context lengths of default transformer encoders, we believed it would be best to remove the extremely long questions (questions with $> 1000$ words) from our data. Additionally, occasionally the 'model answers' are made up of loose terms or short bullet points. As this contrasts with how questions should be answered; in full and complete sentences, we decided to exclude the questions accompanied by such model answers from our QA experiments. As we would like to compare the performance of the RAG setting against other settings, we have excluded questions not referring to the Dutch Civil Code *at least once*. Subsequently, we have removed questions referring to articles outside the Dutch Civil Code, as we intend on only retrieving articles from the Dutch Civil Code.

This leaves us with **310** out of 404 questions for the QA experiments. An overview of the distribution of questions over courses can be seen in Table 1. The distribution over the open-ended question lengths can be seen in Figure 3.[13]

Another observation that we have made, is that some model answers were written as guidelines directed to a corrector; containing sentences like "student should mention this" or "student should not name that". As such phrases might skew our evaluation metrics, we have decided to **manually** remove such phrases in a way that did not interrupt the flow of the answer. In the cases it did, we simply excluded the question. In addition, we noticed that some model answers use arbitrary abbreviations without explicitly explaining them, e.g.: 'wg = werkgever' or 'aok = arbeidsovereenkomst'; respectively meaning employer and labor contract. We have tried to fully write out any unofficial abbreviation we encountered.

Lastly, we have found that a substantial number of questions within the legal exams are not independent; there are many **follow-up questions**. For such questions, the context provided in the preceding question is of relevance to the question at issue. The most obvious way to

[13] A visualization of the model answer lengths, can be seen in Figure 12 of Appendix C.1

deal with this is to just concatenate the preceding context with the follow-up question and to leave out the sentence with the actual question in the first context; which is exactly what we did. This did raise two slight issues; for some follow-up questions, the inclusion of the preceding context would also lead to the inclusion of some non-relevant context (e.g. context only relevant to the original question). The second issue is that some questions were actually follow-up questions of follow-up questions and so on; leading to very long texts. N.B.: For some questions, we deleted an extra one or two sentences at the end of the original question in order to more naturally concatenate the contexts.

*We see that most of the model answers in our dataset are described using a step-by-step approach. Using these questions and answers as in-context demonstrations/examples, it would likely elicit LLMs to incorporate the same step-by-step methodology in the formulation of their response. Thus, we believe that most few-shot settings in our open-ended QA experiment are, by default, types of few-shot CoT prompting (see Section 2.4.3).*

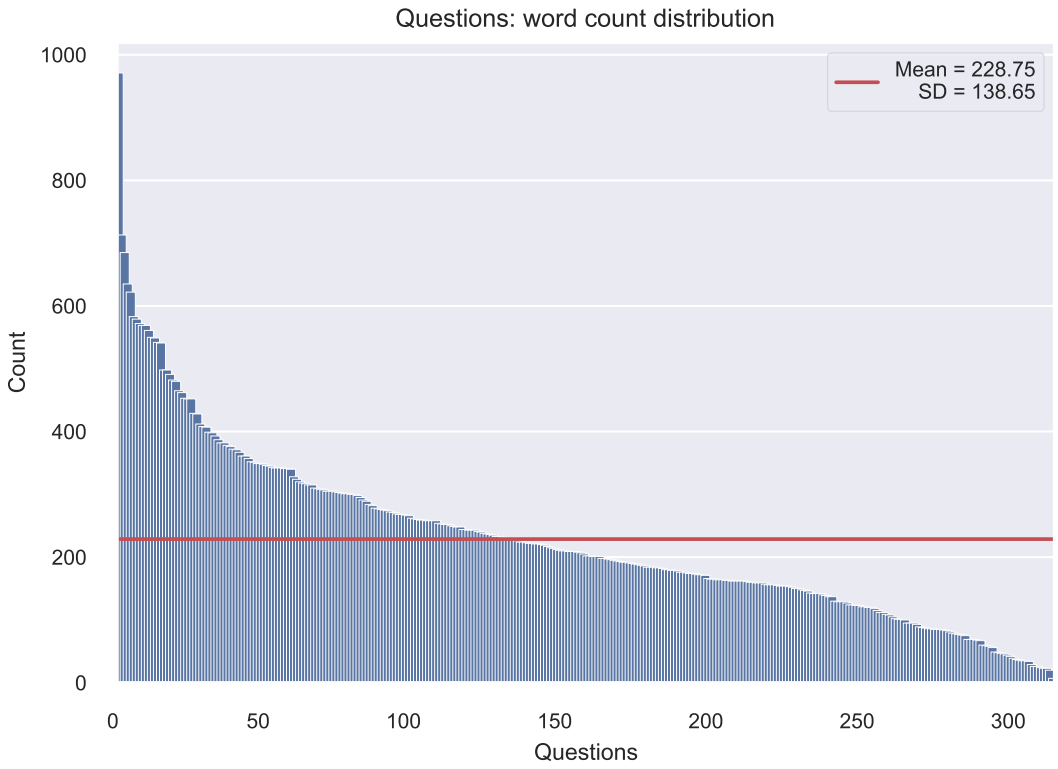


Figure 3: Word count distribution for the questions.

Table 1: Distribution of questions over courses.

| Courses | Open-ended questions | Multiple-choice questions |
|---|---|---|
| Inleiding burgerlijk recht | 78 | 191 |
| Verbintenissenrecht | 75 | 0 |
| Arbeidsrecht | 74 | 0 |
| Personen-familie- en erfrecht | 34 | 0 |
| Goederenrecht | 28 | 2 |
| Onderneming en recht | 21 | 30 |
| Total: | **310** | 223 |

#### 3.1.3 Multiple-choice questions

Multiple-choice questions occur in exams for the courses: '*Inleiding Burgerlijk Recht*', '*Goederenrecht*', and '*Onderneming en recht*'. As we only selected exams for which answers were available, all extracted questions are accompanied by their ground truth answer option. After carefully reviewing all questions, we noticed that for some questions, besides the answer options and the correct answer, an *explanation* is given. These explanations often include the relevant statutory articles. However, for most questions, *no explanation* was given. Thus, most of the time, we do not know what statutory articles are relevant to the questions; deeming retrieval-augmented generation impossible for these questions. As the amount of multiple-choice questions that are accompanied by explanations is limited, we will *not* run RAG experiments for multiple-choice questions.

### 3.2 Data for statutory article retrieval

Statutory laws, statutory articles, or legal norms, are all different notions of the same written rules that constitute the legislation in a Civil law-based system. As a first step in our Statutory Article Retrieval (SAR) experiments, we had to analyze the references made to these statutory articles in the model answers.

#### 3.2.1 Article references

As we chose the exams of the courses based on the assumptive relevance to Dutch Civil Law, it was in our line of expectations that most referenced articles in the model answers are from the Dutch Civil Code; specifically books 1, 2, 3, 4, 5, 6, 7, and 7A. We have not encountered any statutory articles from books 8 and 10 from the Dutch Civil Code, hence we will not take them into account. Moreover, we encountered multiple approaches to refer to articles of the Dutch Civil Code. The most common way to refer to them is as follows: "art. 6:217 BW". In this example, 'BW' stands for 'Burgerlijk Wetboek'; which is the Dutch Civil Code. The '6' before the colon refers to the *book* the article belongs to; a book being a subset of articles relating to a similar topic within the Civil Code. In this case, book 6 is the collection of articles that constitute the 'law of obligations'. Books can be further divided into sections[14] and subsequently into departments[15]; each concerning an even narrower subset of

[14] 'Secties'
[15] 'Afdelingen'

legal articles (see Figure 13 of Appendix C.1). Within an article, there may exist multiple separate elements; e.g. 'leden', 'subleden', 'onderdelen'. It often happens that one refers to these specific parts of an article. For example, by referring to 'art. 6:217 lid 1 BW' or 'art. 15d lid 1 sub e onderdeel 1'. Moreover, occasionally the model answers refer to, more broadly, whole sections of articles (e.g., "zie ook afdeling 2 boek 7 titel 10, tegenprestatie") or a series of articles (e.g., "7:400 BW e.v."; article 7:400 BW and subsequent articles). These kinds of references lead to some ambiguity about what to retrieve as opposed to references to a single article. Furthermore, we sometimes see references to aliases of statutory articles without the article itself explicitly being mentioned e.g. 'redelijkheid en billijkheid'; referring to article 6:248 BW. One last thing to note is that sometimes an article has to be seen in conjunction with another article; leading to the insertion of the word 'juncto' (jo.) between articles, e.g. "art. 6:230b jo. 6:230c lid 2 jo. 6:234 lid 1 BW".

*Concluding, it may not be immediately straightforward how statutory articles are referred to. Thus, we need to make some decisions about how we define the relevance of statutory articles in our IR experiments.*

*N.B. in our model answers we also see references to doctrines in Civil law e.g., 'ontvangsttheorie' and 'droit de suite' as well as references to precedents in case law e.g., 'Haviltex' (case law alias) or 'ECLI:NL:HR:2011:BU7412' (official case law identifier). We acknowledge the relevance of these doctrines and precedents in answering legal questions but, we had to make certain choices regarding the scope of our IR experiments.*

### 3.2.2 Article relevance

To retrieve relevant statutory articles to a legal question, we first had to define our notion of 'relevance'. As relevance in the legal domain has been a long-studied and complex subject [95], our notion of relevance can only be thought of as crude. To keep it short, if the model answer for a legal question in our dataset refers to a statutory article, we consider the referred article to be relevant to the question. Our notion of relevance is **binary**: a statutory article in our collection is either relevant or not relevant, to a question. When multiple articles are mentioned in the answer, we define the **set** of them as relevant. This means that our notion of relevance does not change if an answer refers to the same article one time or ten times. In Section 3.2.1, we mentioned that model answers sometimes refer to a section or a series of articles; leading to ambiguity about which exact articles are referred to. Thus, as 'section references' were relatively rare in our data, we chose to ignore them. In the case of the series references, we considered only the starting article to be relevant.

Furthermore, each individual article can consist of multiple different elements dispersed over several layers. The heterogeneity of articles regarding this aspect led us to wonder how to best deal with this. *Do we retrieve articles in their entirety, or do we retrieve per element?* We settled on **retrieving articles in their entirety**. We believe this was the most uniform approach to deal with statutory articles, but we acknowledge that this may negatively affect extremely long articles and articles containing many semantically different elements.

### 3.2.3 Pre-processing the retrieval collection

Statutory articles in the Netherlands are publicly available. They can be viewed and downloaded from multiple online sources. The official online repository for Dutch law is `https://wetten.overheid.nl`; a website by the Dutch government. On this website, one is able to download any desired Dutch law. Given our needs, we downloaded books 1-7 and 7A from the Dutch Civil Code.[16] As these books were of the XML file format, we first converted them to the tabular CSV format in order to make the data (pre-)processing easier. As most articles belong to specific sections and departments, we also stored that information in the columns of the CSV file. To avoid garbled text, the UTF-8 encoding has to be used to store the articles in CSV.

Furthermore, we noticed three types of irregularities within the downloaded articles. These were articles that were: *expired*, *not yet in force*, or *reserved for future implementation*. As these articles were without content, we removed them from our dataset leaving us with $3579 - 522 = 3057$ articles. An overview of the length of each article can be seen in Figure 4 below.

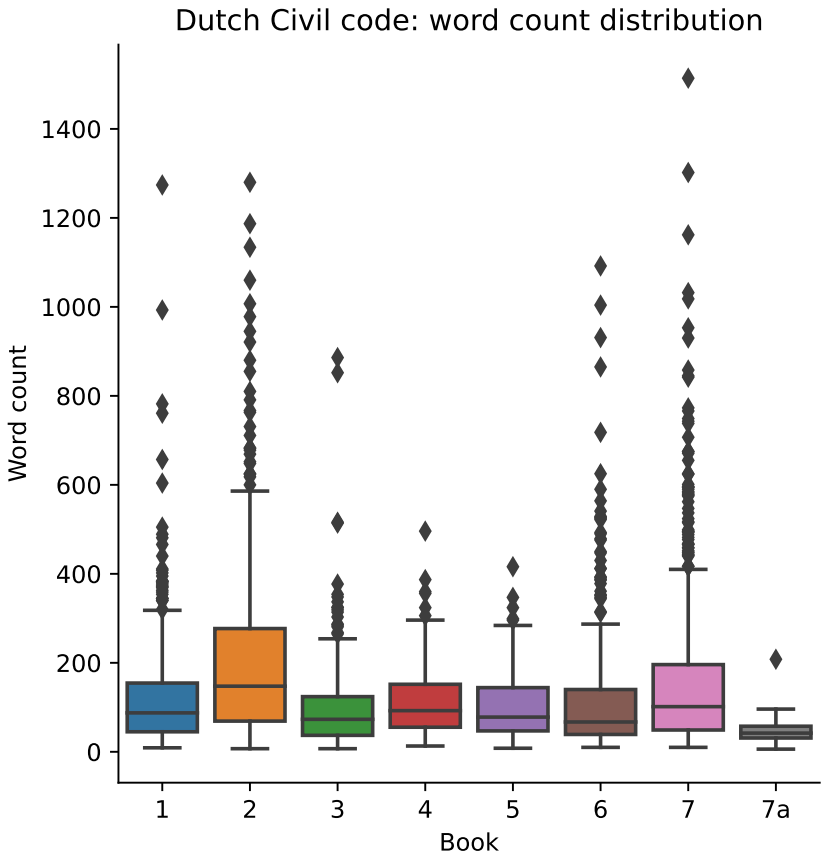


Figure 4: Word count distribution for the statutory articles.

Less than $15.47\%$ of the articles are longer than 250 words, and less than $3.43\%$ of the articles are longer than 500 words.

[16] The versions of the Dutch Civil Code books used, can be seen in Table C.3 of Appendix C.3.

# 4 Methods

In this section, we will be going over the methods that we are using in our experiments. First, we will describe how we have implemented the retrieval component in our RAG approach. This includes going over its workings; the software/packages that we have used; methodologies; and tunable parameters. Subsequently, we will describe how we plan to use LLMs for tackling QA. This includes going over our choice of models; tunable parameters; and prompting techniques used. Lastly, we will describe the evaluation metrics that we will use.

## 4.1 Statutory article retrieval

In Section 2.2.2, we have discussed many dense retrieval techniques. We use a **bi-encoder** architecture to create the embeddings of our statutory articles offline. Subsequently, during run time, we only have to embed the query after which we can compare the query embedding with our offline embeddings. To create, store, and handle these embeddings, we can take several approaches. One popular framework for running IR experiments in the last couple of years has been the PyTerrier framework [96]. However, in the modern day, we also see many '**vector databases**' emerging; specifically created to manage embeddings/vectors. Popular examples of vector databases include the commercial Pinecone[17] database, as well as open-source databases like Milvus [97], Weaviate[18], and Chroma[19]. Given the production-grade quality of most of these, and their ability to integrate with other technologies (e.g. LLM APIs, user interface frameworks), we decided to use a vector database to run our experiment.

### 4.1.1 Embeddings

We selected the open-source **Weaviate** vector database to manage our embeddings. Alongside the vector embeddings, Weaviate also indexes, stores, and provides access to our raw text data.[20] Given a query in our QA pipeline, we would want the vector database to retrieve the relevant data that it has stored. It achieves this by comparing the vector embedding of a query with all vectors stored. **Approximate Nearest Neighbor (ANN)** algorithms are the standard approach in dense retrieval. The specific ANN algorithm used by Weaviate is the **HNSW** algorithm, or the Hierarchical Navigable Small World algorithm [98]. HNSW has been found to often be the fastest algorithm, while also outperforming most other approaches [99].

Different similarity measures to calculate distances between vectors exist. Weaviate supports the following: Squared Euclidean, Manhattan, Cosine similarity, Dot product, and Hamming distance. While Weaviate recommends using the same similarity measure as the ML model used, we will display a quick comparison of their performance in the experiments section (see Section 5).

[17] https://www.pinecone.io/
[18] https://github.com/weaviate/weaviate
[19] https://github.com/chroma-core/chroma
[20] https://weaviate.io/blog/what-is-a-vector-database?

### 4.1.2 Vectorization

Creating embeddings from text data is also called vectorization and the 'vectorizer' (encoder-only) model tasked with this in the Weaviate database, can be chosen in accordance to the use case. The vectorizer that we are using, is the multilingual model '**setu4993/LEALLA-small**' [100]; *a bi-encoder model* (see Section 2.2.2). It is a model with 69M parameters, and the sentence embeddings use 128 dimensions. It is a lightweight model that is based on the computationally heavy language-agnostic LaBSE model [101]. The LEALLA-small model has been trained on the same data as the LaBSE using a maximum sequence length of 128. The similarity function used during *training* is cosine similarity. *We will use our vectorizer 'out-of-the-box'. As the amount of data that we have is limited, we do not perform any fine-tuning.*

As statutory articles are stored as data objects, other relevant information can be included in the vectorization process as seen in Figure 5.[21]

For example, this data object,

```
Article = {
  summary: "Cows lose their jobs as milk prices drop",
  text: "As his 100 diary cows lumbered over for their Monday..."
}
```

will be vectorized as:

```
article cows lose their jobs as milk prices drop as his diary cows lumbered over
for their monday...
```

Figure 5: From data object to vector (example from Weaviate.io)

Thus, statutory articles usually also belong to different 'boeken', 'titels' and 'afdelingen' (see Figure 13). That information can be concatenated with the main text of an article during vectorization. During indexing, the article data objects can also be labeled with 'class names', and the different properties can be labeled with 'property names', which in turn can also be concatenated and vectorized.

*In Section 5.1.3, we will perform an ablation experiment in which we exclude the aforementioned properties one by one, to see how the ablation affects the IR performance.*

### 4.1.3 Hybrid search

In this research, we also want to test the effectiveness of hybrid retrieval in retrieving statutory articles to a legal question. Hybrid retrieval is a combination of lexical retrieval and dense retrieval. The most popular algorithm for lexical retrieval is BM25 (see Section 2.2.1). We will use a slightly modified version called BM25F [102] which extends the original BM25

[21] https://weaviate.io/developers/weaviate/config-refs/schema

with 'multiple weighted fields'. It treats texts as a combination of multiple fields (e.g.: 'title' and 'main text') and as such treats fields differently in the retrieval process.

Hybrid search works by combining the results of both BM25F search and dense retrieval. The sum of the inverse of both rankings is taken, to determine a final score for each item.[22] A parameter called $alpha$, determines the weighting of both rankings. $Alpha = 0$ is a pure BM25F search (lexical retrieval) while $alpha = 1$ is a pure embeddings search (dense retrieval).

Tokenization is something that is done to go from text to tokens. For the vectorizer, this is handled by the model itself. In 'lexical' and 'hybrid' retrieval, we are able to specify how we want to tokenize our text. We use the default tokenization method in Weaviate, which consists of keeping only alpha-numeric characters, lowercasing them, and splitting them by whitespace.

BM25F has the tunable free parameters $k1$ and $b$. Weaviate suggests that the default $k1 = 1.2$ and $b = 0.75$ work well for most cases.[23] We decided to go with these values in our experiments but, we do acknowledge that in previous legal IR work [54] [103], it has been argued that the BM25 performance is "highly dependent on adequately choosing the ($k1$, $b$) values for the task".

## 4.2 Legal answer generation

For our answer generation purposes, we can select between both open-source and commercial models. However, from Section 2.4 we have seen that emergent abilities only emerge in models of large enough scale. As answering legal questions generally requires step-by-step reasoning ability, we have a higher expectation of seeing legal reasoning in the bigger models. As we do not have the computational resources to run the bigger models locally, we have decided to stick with the bigger commercial models.

LLMs are naturally **non-deterministic** and therefore, we will start by running our experiments using multiple repetitions. Subsequently, we will analyze the performance differences between repetitions.

As mentioned in Section 2.3.2, the OpenAI models have performed best in previous work on legal QA using LLMs; outperforming open-source models that have been 'further pre-trained' and fine-tuned on domain-specific data.

The last thing to be considered is the context windows of LLMs. The **context window** of an LLM determines the input length of the models [6]. It directly affects the amount of examples you could provide to a model in the few-shot learning setting, and more generally, the amount of information you could give it as extra context. In the past year, the context window of many models has been drastically increased [69]. We see that this is accomplished by making several architectural adjustments to the original transformer architecture, such as implementing different types of positional embeddings or different types of attention [6]. In addition, the context window is also important to us as it also affects the amount of possibly relevant statutory articles we can pass to a model in a RAG setting.

[22] `https://weaviate.io/developers/academy/zero_to_mvp/queries_2/hybrid`
[23] `https://weaviate.io/developers/weaviate/config-refs/schema`

### 4.2.1 OpenAI models

The company `OpenAI` provides several AI models that we can access through their API. Their LLMs for text generation can be divided into two types of models: completion (legacy) and chat completion models. The latter extends the former (input-output format) with conversational dialogue capabilities. However, when using 'chat completion' models, one is also able to use them in a 'completion' manner.

OpenAI allows the user to tune several parameters when using their text generation models; notably 'temperature' and 'max_tokens'.[24] A high temperature will lead to the output being more random, while a low temperature will lead to the model being more deterministic. Max_tokens simply determines the maximum number of tokens in its output, as longer outputs will be truncated. In our experiments, we will start with a preliminary test to show how tuning these parameters affects the performance. In similar previous work, we see a diverse range of temperature values being used (LegalBench [70]: temperature$= 0.0$, LawBench [69]: temperature $= 0.7$). It is not specified why they have decided on those values.

The OpenAI models that we will be using are: **gpt-3.5-Turbo-0613**, **gpt-3.5-Turbo-1106**, **gpt-4-1106-preview** and **gpt-4-0613**. The context windows of these models are respectively: 4,096; 16,385; 128,000; and 8,192 tokens. The gpt-4-1106-preview model has been trained using data from up to April 2023, while the other models have been trained using data from up to September 2021. N.B.: It should be noted that the number of tokens in a text is not equal to the number of *words* in a text. A rough baseline is: 100 tokens $\approx$ 75 words.[25]

*Given the length of our legal questions and answers, these context windows should allow us enough room for performing our few-shot and RAG experiments.*

## 4.3 Prompting techniques

As a first step, we will test the performance of our generative models in the **zero-shot** performance. Subsequently, we will also try **zero-shot chain-of-thought** prompting and **few-shot** prompting to see how those methods affect performance. Lastly, we will try passing the relevant statutory articles along with the question, to see if the LLMs can correctly incorporate these articles in their answer. This is referred to as the '**oracle**' setting. Out of our 310 legal questions, we will reserve 10 of them for demonstration purposes in our few-shot experiments. To compare across experiments, we have excluded the 10 questions from all other experiments as well. Thus, we will benchmark all methods using **300** legal questions. During the early stages of this research, we found that the Language Model Programming (LMP) frameworks we intended to use (see Section 2.4.3), specifically LangChain and LLamaIndex, were not suited to our needs. There was a problem where we wanted to use a new feature available in the OpenAI API, that was not yet accessible through the LMP frameworks. To avoid similar issues in the future, we opted to write code that directly accessed OpenAI's API.

[24] Other tunable hyperparameters exist in OpenAI's models such as 'top_p'. However, OpenAI recommends only altering either 'top_p' or 'temperature' but not both (`https://platform.openai.com/docs/api-reference/chat/create`).

[25] `https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them`

### 4.3.1 Zero-shot

Zero-shot prompting is the most basic form of QA prompting; no examples are passed along, the model sees the task only once and it does not have a reference on how to answer. The prompt simply constitutes the question. In a single prompt setting, we define this as the *direct* prompt. In multiple-choice question answering, we also test zero-shot prompting in a 2-step prompting format in which we prompt the model a second time to elicit a single-letter response (A, B, C, or D). The general format of the zero-shot prompt is as follows:

**Zero-shot prompt:**

<QUESTION>

*The experiments performed in the zero-shot setting shall generally constitute the baseline for the other experiments.*

### 4.3.2 Zero-shot chain-of-thought

In almost all exams, a description on how the legal questions should be answered is shown on the front page. An example of such a description (nr. 6) can be seen below:

> "Beantwoord de vragen kort en bondig. U dient uw antwoorden altijd te motiveren. Bij enkele vragen dient u dit te doen aan de hand van het zogenaamde stappenplan: 1. Wat zijn de relevante feiten in de casus 2. Wat is het probleem ofwel de rechtsvraag 3. Welke rechtsregel(s) zijn hiervoor van belang 4. Analyse van de rechtsregel(s) en toetsing van de feiten aan de rechtsregel(s) 5. Daaruit vervolgens een conclusie trekken."

The format for a zero-shot CoT prompt, using description nr. $6$, would be as follows:

**Zero-shot CoT prompt:**

<QUESTION>
<description nr. $6$>

As opposed to prompting with a standard CoT strategy like appending the phrase: 'let's think step for step' to a prompt [84], we *append* official exam descriptions to prompts to see how that affects performance. Such extra instructions can be seen as 'domain-specific CoT cues' [89]. All exam descriptions used can be found in Appendix A.

### 4.3.3 Automated prompt optimization

Furthermore, we hope to perform automated prompting optimization experiments in order to find the most optimal CoT cue (instruction) for our QA experiments. We will use the existing OPRO [93] framework to experiment with this. The results of these experiments can be seen in Section 5.2.7. We only test it for the multiple-choice questions as we need a stable optimization target; the accuracy metric being well suited for this.

The authors of OPRO describe the framework as "*large language models are leveraged as optimizer, where the optimization task is described in natural language*" [93]. They define its goal as finding an instruction that maximizes the task accuracy. Two LLMs are used here:

one optimizer LLM and one scorer LLM. The prompt to the optimizer is called a **meta-prompt** and the optimizer is tasked with finding prompt instructions. The scorer LLM is tasked with evaluating the newly found instruction on a test dataset. A high-level overview of the framework can be seen in Figure 6.

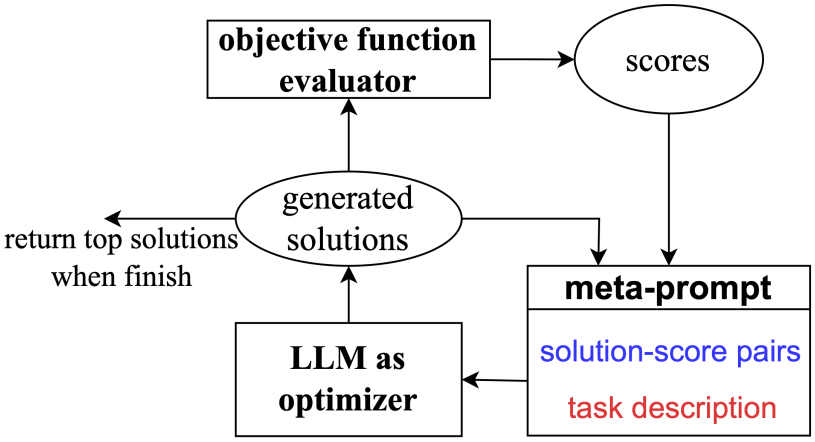


Figure 6: High-level overview of OPRO. Directly copied from the OPRO paper [93].

Effectively, the optimizer generates $x$ new instructions per step, for an $y$ amount of steps. During every step, the $x$ instructions are evaluated on a training set. The best scoring instruction of that step is then added to the meta-prompt along with its score. After a few iterations, the meta-prompt contains multiple examples of previous instruction-score pairs. **At the core of this framework lies the assumption that these previous instruction-score pairs can assist the optimizer LLM in finding newer better instructions.** After the last optimization step, the best prompt can be evaluated on the questions in the test set.

Within the setup, one can specify a couple of settings. This includes but is not limited to: the training/test/validation set ratio, the step budget, the number of instructions created and evaluated per step, the two LLMs to use, and the instruction position in the prompt. The instruction position can be as follows: '*before the question*', '*after the question*', and '*at the start of the response*'.

As legal scholars typically use legal reasoning techniques such as IRAC (see Section 2.4.3), we hypothesize that instructions that encourage the use of IRAC in LLMs might be most effective. Consequently, we defined the meta-prompt to the optimizer LLM as follows:

**Meta-prompt:**

> "Je taak is om een instructie <INS> te genereren. Hieronder staan eerder geteste instructies met hun score. The score varieert van 0 tot 100.
>
> <previous instruction(s) + score>
>
> ...
>
> Genereer een instructie die anders is dan de instructies <INS> hierboven, en die een hogere score haalt dan de instructies <INS> hierboven. De instructie moet beginnen met <INS> en eindigen met </INS>. De instructie moet beknopt, effectief, en over het algemeen toepasbaar zijn op alle bovenstaande problemen. De instructie moet zich ook baseren op de IRAC methode."

Consequently, an example of a newly found instruction could be as follows:

> "Identificeer de specifieke juridische kwestie, analyseer zorgvuldig de feiten en wetgeving, bespreek de argumenten en tegenargumenten, en kom tot een weloverwogen conclusie met concrete aanbevelingen voor verdere stappen."

As we intend on putting the instruction *after the question*, the format of the prompt to the scorer LLM in OPRO is similar to the zero-shot CoT prompting format:

**Prompt to scorer LLM:**

> <QUESTION>
> <newly found instruction>

*The goal of this experiment is to simply demonstrate the workings of a prompt optimization framework. As the methodology of prompt optimization is relatively new, the choice to use OPRO was somewhat arbitrary. During the writing of this thesis, we did not identify one dominant approach to prompt optimization and as such, one might opt to use other frameworks, e.g. ones based on evolutionary algorithms [94].*

### 4.3.4 Few-shot chain-of-thought

When setting up a few-shot learning experiment, one has to make certain decisions regarding the format of in-context learning demonstrations; notably the demonstration **selection**, demonstration **format**, and demonstration **order**. For a recap on few-shot prompting, see Section 2.4.3.

In similar previous work on legal QA, we have seen that *zero to eight* samples were used for in-context demonstrations/examples [70]. The authors stated that the number of samples had been chosen in regard to the availability of the data, and the sequence length of the samples (context window). In contrast, we have seen that the 'LawBench' [69] paper only evaluates their legal QA tasks in a zero-shot and a one-shot setting. N.B.: In line with findings on general-domain tasks, it has been observed that LLMs on legal tasks are also highly sensitive to the **choice of these in-context samples** [69].

Furthermore, as most multiple-choice questions are not accompanied by a step-by-step explanation, few-shot prompting the multiple-choice questions would not be considered few-shot *CoT*. As most open-ended questions are accompanied by a thorough explanation, our few-shot learning experiments using open-ended questions should be classified as few-shot CoT prompting. In our few-shot experiments, we *prepend* up to three questions and accompanying model answers as demonstrations. The three examples used can be seen in Appendix D. Given $n$ demonstrations, the format of the prompt would be as follows:

**Few-shot CoT prompt:**

> <example question $1$>
> <answer to example question $1$>
> ...
> <example question $n$>
> <answer to example question $n$>
> <QUESTION>

*The goal of this experiment, is to test how using a varying amount of task demonstrations affects performance.*

### 4.3.5 Retrieval-augmented generation

As our retrieval augmented generations experiments depend on the successful retrieval of articles, we plan to test RAG in both the full pipeline and the oracle setting. In the oracle setting, we pass the legal question and the relevant statutory articles as context in the prompt to the models.

We use a simple format for our RAG prompts, in which we simply *prepend* the relevant articles to the question. Given relevant articles $x$, $y$, ..., $n$ for a question, it would look as follows:

**RAG prompt:**

> Article $x$
> <content article $x$>
> Article $y$
> <content article $x$>
> ...
> Article $n$
> <content article $n$ >
> <QUESTION>

*In this experiment, we hope to test if passing the relevant statutory articles as context in the prompt, leads to a better performance. The primary question in this is about how well the LLM is able to utilize this extra context.*

## 4.4 Evaluation

To answer our research question, we need to evaluate both the performance of the IR component and the question-answering component, and in order to evaluate them objectively, we need to use the right evaluation metrics.

### 4.4.1 Statutory article retrieval

Historically, the following metrics have commonly been used to evaluate the performance of IR systems [36]:

- **Precision**: the fraction of retrieved documents that are relevant.

$$Precision = \frac{\#(relevant\ retrieved\ documents)}{\#(retrieved\ documents)}$$

- **Recall**: the fraction of relevant documents that are retrieved.

$$Recall = \frac{\#(relevant\ retrieved\ documents)}{\#(relevant\ documents)}$$

- $F_1$: weighted harmonic mean of the precision and recall.

Something to note is that IR metrics do not necessarily have to be evaluated over *all* retrieved documents. We can use a **cut-off point** $k$ to specify that the metric is calculated over the top-$k$ retrieved documents. E.g.: to calculate `precision@100`, we mean to calculate the precision over the top-100 retrieved documents.

Still, precision and recall are set-based measures and as such, they do not take the ranking of the retrieved documents into account. This means that they do not pay attention to where the *relevant* retrieved documents are located in the list of retrieved documents; they could be at the bottom, at the top, it does not matter. Fortunately, several measures exist that are more suited for ranked retrieval.

One of the most standard metrics used in the IR community is `mean Average Precision` (**mAP**) [36] [1]. Its appeal comes from the fact that it is a single metric that allows one to easily compare the performance of different IR systems.

To explain mAP, we first need to describe the metric `Average Precision` (AP) [1]. Given: a query $q$, a set of relevant documents $R$, and a list of retrieved documents, we can calculate the AP for this query by averaging the precision values calculated at every relevant document's rank in the retrieved document list. If we were to retrieve 1000 documents for a query, with 2 documents being relevant, located at rank 2 and 500, then the AP for this query would be:

$$AP_q = \frac{1}{|R|} * \sum_{i \in R} Precision(i) = \frac{1}{2} * [\frac{1}{2} + \frac{2}{500}]$$

.

As mAP is just the mean AP over all queries $q \in Q$, we get the following formula for mAP [1]:

$$mAP = \frac{1}{|Q|} * \sum_{q \in Q} AP_q$$

.
In the following chapters, we will primarily be using **recall@100**, **recall@1000**, and **mAP** as evaluation metrics for our experiments. We do not expect our retrieval to be $100\%$ perfect, therefore we chose to use a recall cut-off point that is higher than in a production-grade IR pipeline. The mAP metric will be used to showcase the IR performance independent of an arbitrarily chosen cut-off point.

*For calculating all mentioned metrics in this subsection, we have used the 'pytrec_eval' [104] package in Python.*

### 4.4.2 Legal question-answering

For evaluating the legal QA task, we will be using a variety of metrics to compare the quality of the LLM answers to that of the ground truth answers. For the open-ended questions, we will be using automated metrics such as BLEU [105], ROUGE [106], and BERTScore [107]. In addition, we will also use precision and recall to check the correctness of referenced statutory articles in answers.

**BLEU** and **ROUGE** are both examples of **n-gram** based metrics. An n-gram is a sequence of $n$ words [1]. As a uni-gram (1-gram) is a sequence of one word, the sentence "I study law" can be decomposed into the uni-grams: "I", "study", and "law". Similarly, the same sentence can be decomposed into the bi-grams (2-grams): "I study" and "study law". The key idea behind BLEU and ROUGE is that they look at the overlap of n-grams between a candidate sentence (LLM output) and a reference sentence (ground truth) [1]. However, BLEU is focused on evaluating *precision* with this overlap: the number of overlapping n-grams divided by the total number of n-grams in the *candidate* sentence. In contrast, ROUGE is focused on evaluating *recall* with this overlap: the number of overlapping n-grams divided by the total number of n-grams in the *reference* sentence.

If we calculate ROUGE-1, this means that we look at the overlap of 1-grams; for ROUGE-2, it is the overlap of 2-grams, and so on. The same principle applies to the different BLEU metrics. However, the default BLEU metric is defined as the geometric mean of BLEU-1, BLEU-2, BLEU-3, and BLEU-4 [1]. Moreover, there exists a metric called **ROUGE-L** [108], which is a scoring based on the 'longest common subsequence'.[26] Specifically, it is defined as the length of the longest common subsequence between the candidate sentence and the reference sentence, divided by the number of uni-grams in the candidate sentence. We have seen that in similar experiments on legal QA [69], Rouge-L was used as the standard metric for generative tasks.

Two common pitfalls in these n-gram-based metrics are that they depend on the exact matching of terms (see 'vocabulary mismatch problem' Section 2.2.1), and that they fail to capture distant dependencies and crucial ordering in n-grams [107]. Consequently, the **BERTScore** metric was developed to overcome these pitfalls. It makes use of BERT embeddings to compare candidate sentences to reference sentences. With BERTScore, the candidate and reference sentences are passed through a BERT model to turn each token in both sentences into embeddings. Each pair of 'token embeddings' between sentences is then scored by their cosine similarity. BERTScore provides a precision and a recall score [1]. Taking the harmonic mean of the precision and recall gives us the $F_1$ BERTScore; the overall recommended metric to use by the authors of BERTScore [107].

[26] This subsequence does not have to be consecutive, but it has to be in order.

Moreover, we are using regular expressions to extract referenced statutory articles in LLM outputs. As all questions are accompanied by a model answer containing the references to the relevant statutory articles, we can compare these sets of referenced articles in the model answer and the LLM output. We do this by calculating the aforementioned precision and recall of the referenced articles in the model answer and the LLM output.

All these automated metrics can give us an *indication* of how good an answer is. However, they still lack the finesse of a human evaluation. Thus, we will also periodically check the quality of the answers throughout the experiments ourselves; sharing any notable *empirical* observations.

For multiple-choice questions, we have removed all (erroneous) questions in which more than one answer was marked as correct in pre-processing. Thus it is an instance of a single-label classification problem and we are able to simply use '**accuracy**' to evaluate the performance.

$$Accuracy = \frac{\#(correct\ answers)}{\#(total\ number\ of\ questions)}$$

*There are two approaches for evaluating LLMs: extrinsic and intrinsic evaluation. So far, we have only discussed extrinsic evaluation; the LLM performance on a downstream task like legal QA. The evaluation of the LLM itself: the 'task-agnostic' performance is called intrinsic evaluation [109]. One commonly used metric for this is 'perplexity' [110]. Having that said, we are only concerned with extrinsic evaluation in this research.*

# 5 Experiments

For our experiments, we will first describe the findings of our statutory article retrieval task, after which we will continue with our question-answering experiments.

## 5.1 Statutory article retrieval results

In the following subsections, we will show our findings regarding our `Statutory Article Retrieval` (SAR) experiments. We will test out multiple hyperparameter configurations and vectorization approaches to see what works best.

### 5.1.1 Distance measures

In a quick preliminary experiment, we have researched the effect of using different similarity measures for our dense retrieval. We use the mAP metric as it gives a single value indication for the performance of our dense retrieval.

Table 2: mAP scores for different distance measures using dense retrieval.

| | Cosine | Dot | Hamming | L2-squared | Manhattan |
|---|---|---|---|---|---|
| mAP | **0.0027** | 0.0016 | 0.0025 | 0.0015 | 0.0016 |

From Figure 2, we see that cosine similarity performs best. We will continue to use this similarity measure for our next experiments.

### 5.1.2 Hybrid search

We start our IR experiments with a hybrid search over our 300 legal questions. As mentioned before: alpha $= 0.0$ indicates a pure lexical retrieval while alpha $= 1.0$ indicates a pure dense retrieval. We see from Figure 7 that for $recall@100$: an alpha value of $0.1$ performs best and that in the $recall@1000$ setting: an alpha value of $0.0$ performs best. If we look at the mAP in Figure 8, we see that an alpha value of $0.0$ performs best. At this point, we feel the need to draw some conclusions from this experiment as we believe that they affect the viability of our planned RAG experiments.

We believe that in a full RAG pipeline a $recall@100$ of $\approx 0.44$ in the IR step, would most likely not improve the performance in QA; as most *relevant* articles are not retrieved. In addition, if we pass the retrieved articles to the LLM as context to a legal question: all retrieved relevant articles would be intertwined with the $\gg$ retrieved non-relevant articles; opening up the question if LLMs are able to distinguish the relevant from the non-relevant articles. As this strays too far from our main research question, combined with the fact that it is not even clear if current LLMs are even able to reason with relevant statutory articles, we have decided to only implement RAG using the oracle setting; in which *only relevant articles are passed to the LLM*.

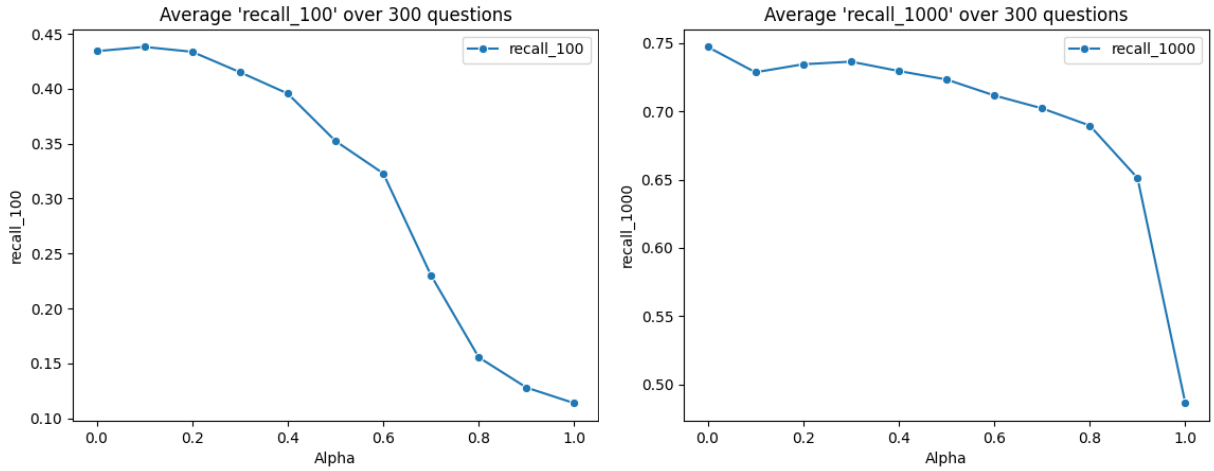


Figure 7: Recall@100 and recall@1000 for different alpha values.

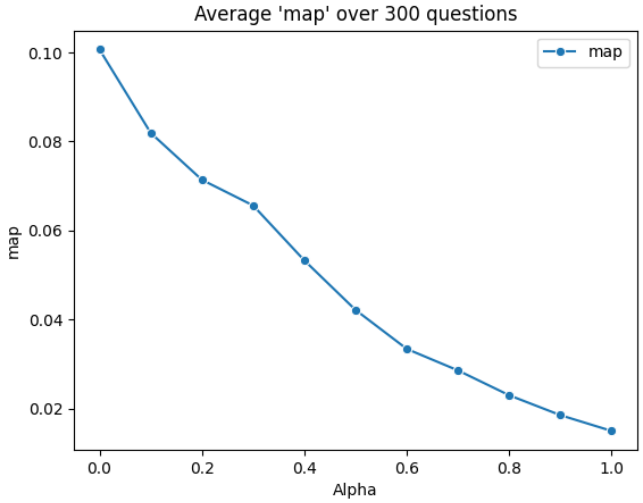


Figure 8: mAP for different alpha values.

### 5.1.3 Properties to vectorize (ablation study)

As mentioned in Section 4, different properties of an article can be included in the vectorization process. In this experiment, we exclude these properties one by one to see how they will affect performance.

As this ablation study only affects dense retrieval, and not lexical retrieval, all configurations have the same performance at alpha $= 0.0$. The '*Wetsartikel_baseline*' plot represents the configuration in which no property is excluded. In the other configurations, the '*x*' in '*Wetsartikel_x*' indicates the property that is excluded.

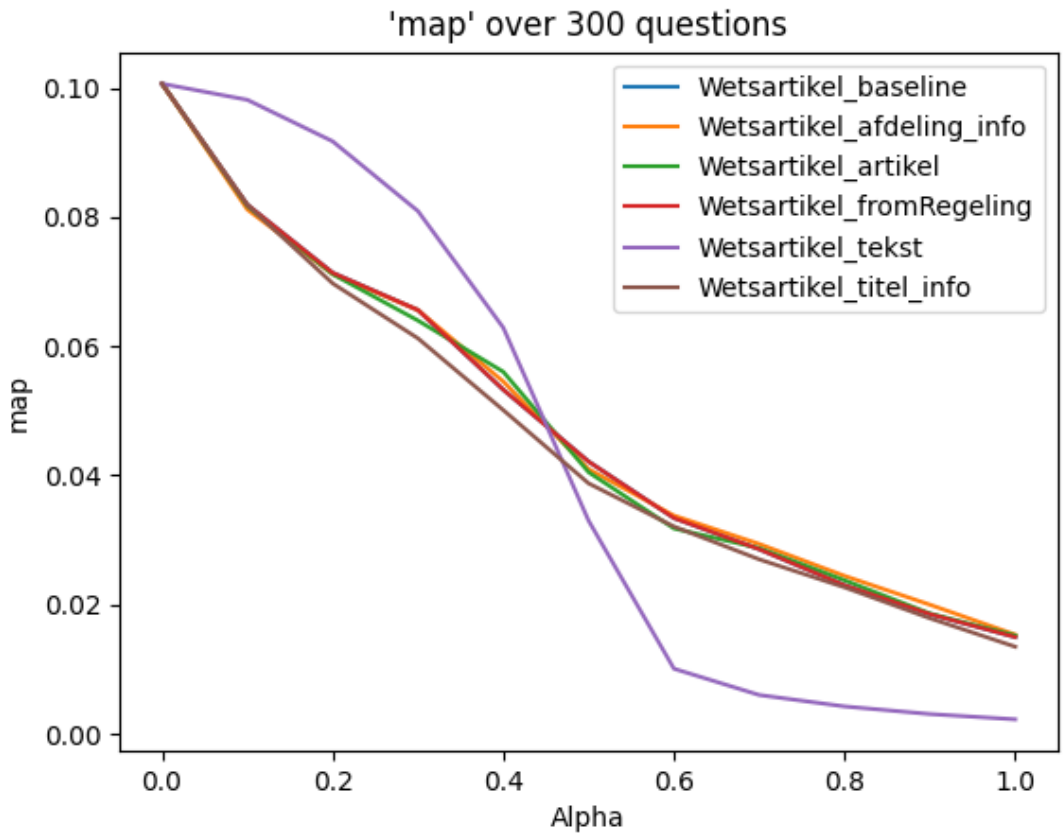


Figure 9: mAP for different excluded properties.

From Figure 9, we see that an alpha value of $0.0$ performs best. This is in line with the findings of earlier experiments (see Figure 8) in which we also use mAP to evaluate a hybrid search. It is interesting to note that the exclusion of the article contents ('*Wetsartikel_tekst*') in the vectorization process, leads to a better score when the hybrid search is predominantly lexical retrieval, while excluding it in a predominantly dense retrieval worsens the score. Without the contents of an article, the vectorized articles exclusively represent the properties/sections of an article.

We believe that the sigmoid-like curve seen is caused by the fact that the model has been trained on sentences up to 128 tokens. Therefore, we hypothesize it is not able to deal well with longer sequences; i.e. when the article contents are also vectorized. It seems that in a predominantly dense retrieval hybrid search, the retrieval of solely vectorized properties/sections is insufficient for a good retrieval; while in a predominantly lexical hybrid search, the retrieval of vectorized properties/sections better complements the lexical retrieval.

*All in all, all hybrid searches in this experiment seem to be outperformed by sole lexical retrieval; affirming that BM25 variants remain a strong baseline in IR.*

## 5.2 Legal question-answering results

In this question-answering section, we will go over the different: prompting methods, prompts, hyperparameters, and models that we have tested.

### 5.2.1 Open-ended questions: zero-shot

We have started by simply providing our questions as prompts to OpenAI's *gpt-3.5-turbo-0613* model without any modification to its default hyperparameters. The default hyperparameter values are defined as follows:

$$Temperature = 1.0 \quad Max\_tokens = 512 \quad Repetitions = 5$$

Given the non-determinant nature of LLMs, we have the model provide five answers for each question. After manually inspecting these answers in the zero-shot settings, we immediately notice a few things:

1. Most of the time, the model does not mention any statutory article in its response.
2. The model sometimes covers its bases by incorporating certain (safety) phrases in its response. An example that was translated from Dutch, is as follows: "*however, it is important to note that this is a hypothetical legal question and the outcome may depend on the specific circumstances of each case and the applicable law in the relevant jurisdiction.*".

Not mentioning any statutory article is a clear problem in terms of explainability. The second observation might skew the automated evaluation metrics, as the model answers do not include such messages.

Within zero-shot prompting, we have experimented with changing the following hyperparameter settings of the model: 'temperature' and 'max_tokens'. The results can be seen in Figures 10 and 11.

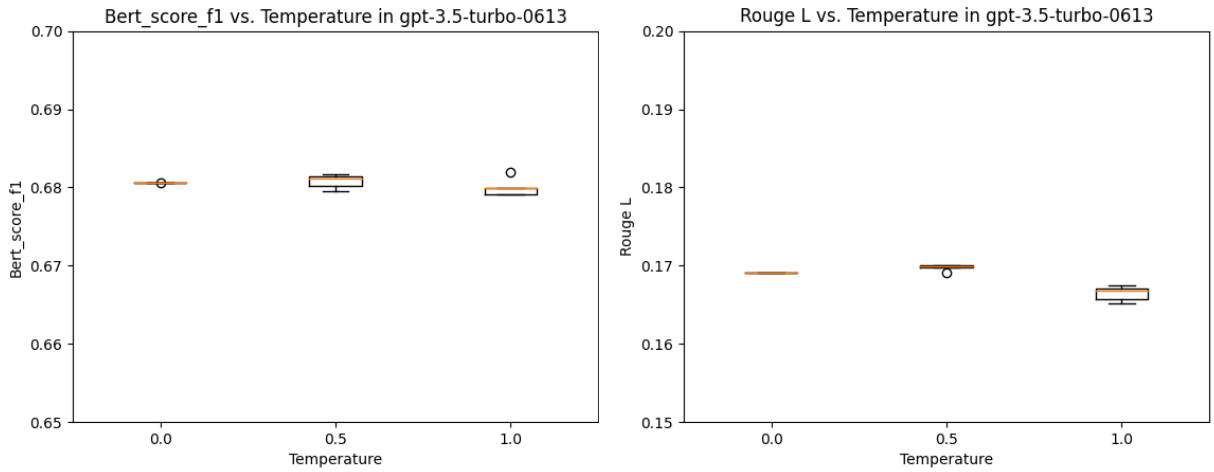


Figure 10: Rouge-L and $F_1$ BERTScores for different temperatures over 5 iterations.

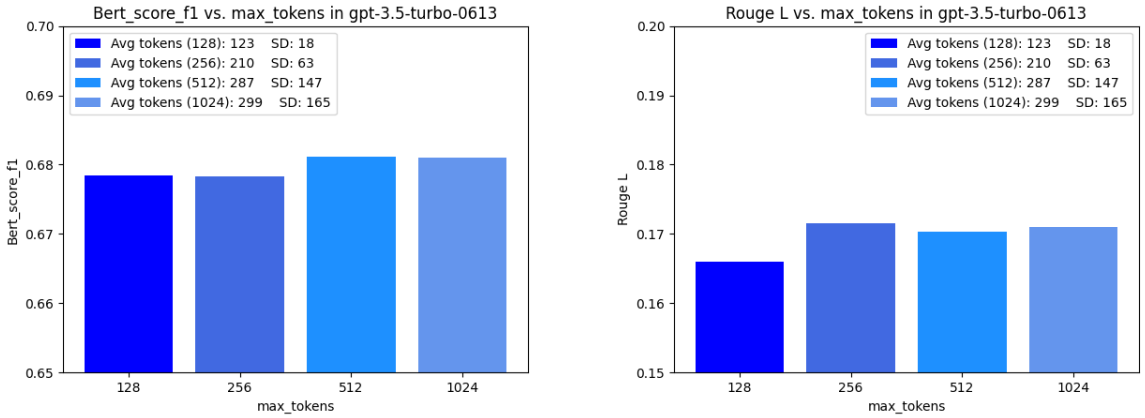


Figure 11: Rouge-L and $F_1$ BERTScores for different max_tokens over 5 iterations.

We see from the output that when using temperature $= 0.0$, the model becomes **deterministic**. This means that when we ask it the same question five times, it will give five identical answers. The identical Rouge-L and $F_1$ BERTScores at temperature $= 0.0$ in Figure 10 reflect this. Moreover, we do not see a substantial variance within the box plots of the Rouge-L and $F_1$ BERTScores in Figure 10. This could either indicate that the output over multiple repetitions does not differ a lot, or that the metrics used are not suited for capturing the output differences. If we look at the average scores across the tested temperature values, we also do not see a substantial variance.

*Therefore, we do not see the reason to use a temperature value other than* $0.0$*; as using a deterministic model would reduce the need to run the experiments over multiple repetitions.*

Furthermore, we see that with a maximum token limit of 128, some answers get cut off prematurely; negatively affecting the performance. This applies to a lesser extent to maximum token limits of 256 and 512. From empirical observations, we see that setting the maximum token limit to 1024 removes all risk of cutting off answers prematurely.

*Going into our next experiments, we have set the temperature to* $0.0$*, the max_tokens to* $1024$*, doing only a single run per configuration.*

### 5.2.2 Open-ended questions: zero-shot, different models

For this next experiment, we test different GPT models in the zero-shot setting. We use the following hyperparameters for all models:

$$Temperature = 0.0 \quad Max\_tokens = 1024 \quad Repetitions = 1$$

From Table 3, we observe that most Rouge-L and $F_1$ BERTScores across the tested models do not substantially differ, except for the unexplained worse Rouge-L performance in the gpt-4-1106-preview model.

Looking at the article-precision and article-recall metrics, we *do* see substantial differences between the models. We see that the gpt-4-1106-preview model has the highest precision and recall; thus including more relevant statutory articles in its output. As the tested models are commercial models, the exact details about their architectures and training methods are unknown. In line with online rumours, we speculate that the newer GPT-4 models could: be larger in size, be trained more rigorously, or be constituted of multiple models; each factor

Table 3: Mean performance metrics for different LLMs (zero-shot). The gpt-3.5-turbo-0613 model that was used in the previous experiment is included.

| | Avg Rouge-L | Avg BERTScore $F_1$ | Avg article precision | Avg article recall |
|---|---|---|---|---|
| gpt-3.5-turbo-0613 | 0.17 | 0.68 | 0.08 | 0.07 |
| gpt-3.5-turbo-1106 | **0.18** | **0.69** | 0.07 | 0.06 |
| gpt-4-0613 | 0.17 | 0.68 | 0.19 | 0.16 |
| gpt-4-1106-preview | 0.14 | 0.68 | **0.25** | **0.24** |

possibly contributing to the measured difference in article-precision and article-recall across models.

One non-performance-related difference between these models is that the average lengths of their responses substantially differ (see Figure 14 of Appendix C.1).

*Based on the better article-precision and article-recall scores of the GPT-4 models, we would recommend using them over the GPT-3.5-turbo models for legal QA.*

#### 5.2.3 Open-ended questions: zero-shot, CoT cues

Legal exams usually contain a description, or specification, on how the questions in them should be answered (see Section 4.3.2). In this experiment we append these descriptions to the end of our prompt, to see how it affects the performance. The descriptions 1–7 can be found in Appendix A. The following model is tested in this experiment:
**gpt-3.5-turbo-1106**, using the following hyperparameters:

$$Temperature = 0.0 \quad Max\_tokens = 1024 \quad Repetitions = 1$$

Table 4: Performance different zero-shot CoT cues (see Appendix A).

| Exam description # | Avg Rouge-L | Avg BERTScore $F_1$ | Avg article precision | Avg article recall |
|---|---|---|---|---|
| <no description> | **0.18** | **0.69** | 0.07 | 0.06 |
| 1 | **0.18** | 0.68 | 0.08 | 0.07 |
| 2 | 0.17 | **0.69** | 0.38 | 0.30 |
| 3 | 0.16 | 0.68 | **0.39** | **0.33** |
| 4 | 0.17 | 0.68 | 0.38 | 0.31 |
| 5 | **0.18** | 0.68 | 0.10 | 0.08 |
| 6 | 0.17 | 0.67 | 0.17 | 0.13 |
| 7 | **0.18** | **0.69** | 0.38 | **0.33** |

From Table 4, we can see that the difference in Rouge-L and $F_1$ BERTScores is small, but that the difference in article-precision and article-recall is quite big.

*This suggests that appending these types of cues to the prompt could substantially help the models in citing relevant statutory articles.*

However, we acknowledge that these metrics are limited, as they do *not* capture how well the models are able to correctly reason with the referenced articles.

#### 5.2.4 Open-ended questions: few-shot

In this experiment, we perform a small experiment where we add either: 1, 2, or 3 examples to the prompt in order to see how the examples affect the model's QA performance. The examples used can be seen in Appendix D. In regards to the prompting format, we simply prepend the examples to all questions before prompting. The following model is tested in this experiment: **gpt-3.5-turbo-1106**, using the following hyperparameters:

$$Temperature = 0.0 \quad Max\_tokens = 1024 \quad Repetitions = 1$$

Table 5: Performance few-shot learning (1, 2, and 3 examples).

| # examples | Avg Rouge-L | Avg BERTScore $F_1$ | Avg article precision | Avg article recall |
|---|---|---|---|---|
| <No examples> | 0.18 | 0.69 | 0.07 | 0.06 |
| 1 | **0.19** | **0.70** | 0.28 | 0.22 |
| 2 | **0.19** | **0.70** | 0.31 | **0.26** |
| 3 | **0.19** | **0.70** | **0.33** | 0.24 |

We can see that the Rouge-L and $F_1$ BERTScores are slightly raised when adding the examples to the prompts. Moreover, we see a substantial increase in article-precision and article-recall when adding the examples.

*This could suggest that the added examples make the model more aware of the fact that the legal questions require references to relevant statutory articles in the answers. Consequently, we would recommend the use of few-shot examples in legal QA.*

*In this experiment we have researched if the inclusion (and number) of examples, had any effect on the performance of LLMs. We note that we have not extensively researched how the: selection, order, and format of examples affect performance. We acknowledge that these factors could affect our results.*

#### 5.2.5 Open-ended questions: retrieval-augmented generation

As mentioned in Section 5.1.2, we have decided to only test RAG in the oracle setting; as we have deemed the quality of our IR component insufficient.

The model we use for this experiment is: **gpt-3.5-turbo-1106**, and the hyperparameters used for this experiment are as follows:

$$Temperature = 0.0 \quad Max\_tokens = 1024 \quad Repetitions = 1$$

As we initially did not explicitly indicate in the prompt what the model should do with the prepended articles in the context, we also decided to test a configuration in which we append a cue to the end of the prompt specifying that the model should use all listed articles in formulating its response. The results can be seen in Table 6.

Table 6: Performance oracle RAG (cue = "Gebruik alle bovenstaande artikelen in je antwoord.".)

| Setting | Avg Rouge-L | Avg BERTScore $F_1$ | Avg article precision | Avg article recall |
|---|---|---|---|---|
| <No RAG> | 0.18 | 0.69 | 0.07 | 0.06 |
| RAG | **0.21** | **0.71** | 0.82 | 0.66 |
| RAG + cue | **0.21** | **0.71** | **0.96** | **0.88** |

We can see that in the oracle RAG setting the Rouge-L and $F_1$ BERTScores are increased; although the difference is small. We also see that when passing the articles as context, it does not automatically use *all* articles in its answer. Appending the aforementioned cue helps to increase the article-precision and article-recall scores, but it does not affect the Rouge-L and $F_1$ BERTScores.

*Based on these results, we would recommend the use of RAG in the case of a perfect IR system.*

#### 5.2.6 Multiple-choice questions: zero-shot

In this experiment, we test the model: **gpt-3.5-turbo-0613** on the **223 multiple-choice questions** in our dataset. We have set the max_tokens to the default value of 512 as we are expecting short responses.

We start off by simply prompting the LLM with the question and answer options; the *direct* prompt setting. An example of the format that we are using can be seen in Appendix B. In this direct prompt setting, we immediately notice a few issues (examples listed in Appendix B). We summarize the most notable ones below:

1. The model does not specify an answer option, instead treating the multiple-choice question as an open-ended question.
2. The model responds with his own answer option (e).
3. The model marks multiple answer options as correct.
4. The model abstains from answering.
5. The right answer option is embedded somewhere in the middle of the output.
6. The model goes through all the answers without specifying its exact choice.

All these issues made evaluating the multiple-choice questions harder than necessary. Thus, we experimented with methods to tackle these issues. We have tried appending the phrase: "U dient uit de vier keuzemogelijkheden het meest juiste antwoord te kiezen." to the prompt. This phrase constitutes the first part of the exam description shown in multiple-choice exams (see Appendix A.2), and appending it to the prompt substantially helped to reduce most issues. However, this did not *fully* guarantee a single answer option as a response, hence we still had to manually correct some of the answers.

In the setting where we append the aforementioned phrase to the prompt, we can see from Table 7 that a single run with a temperature of $0.0$ achieves the highest accuracy. It

outperforms the average accuracy of the $N = 5$ runs that use a temperature of $1.0$. Additionally, it outperforms the accuracy of $N = 5$ runs where a majority vote is used to decide the answer per question (e.g., ['a', 'b', 'c', 'b', 'b'] $\rightarrow$ 'b').

Table 7: (Average) accuracy over 223 multiple-choice questions.

| | (Average) accuracy | Majority vote accuracy |
|---|---|---|
| N = 1, temperature = 0.0 | **0.47** | - |
| N = 5, temperature = 1.0 | 0.40 | 0.41 |

*We saw from the open-ended QA that the temperature value did not seem to have a big effect on the Rouge-L and $F_1$ BERTScores. However, in this multiple-choice QA experiment where we use accuracy as a metric: we see that a temperature value of $0.0$ leads to substantially better scores over using a temperature of $1.0$ & taking the mean score over repetitions.*

The accuracy scores when no cue is appended to the prompt can be seen in Table 9 of Appendix C.3. When comparing the accuracy scores between Tables 7 and 9, we see that the differences are substantial; suggesting that adding a simple cue such as "U dient uit de vier keuzemogelijkheden het meest juiste antwoord te kiezen." is a helpful but not so straightforward addition.

To further analyze the effectiveness of appending such cues, we have experimented with automated prompt optimization in the next section.

#### 5.2.7 Multiple-choice questions: automated prompt optimization

In this section, we describe the experiment performed as discussed in Section 4.3.3. In the setup, we set the training and test set ratio to respectively $22/223$ and $201/223$ questions. We do not use a validation set. We run the experiment for $20$ steps, evaluating $8$ instructions per step. We use the **gpt-3.5-turbo-1106** model as the scorer and optimizer. The instruction is added to the *beginning* of a question. In line with OPRO paper recommendations, we have set the temperature of the optimizer LLM to $1.0$ in order to generate 'diverse and creative' instructions [93]. We use a temperature of $0.0$ for the scorer LLM and start with an *empty* initial instruction.

After first running OPRO, we quickly encountered problems similar to the ones listed in Section 5.2.6. As manually correcting the answers in the 1-step prompting approach was infeasible in automated prompt optimization, we experimented with other ways to tackle these issues. We solved them by using a 2-step prompting approach where we prompt the LLM a second time with the prompt: "Dus het meest juiste antwoord is:". This led to the LLM giving a single answer option most of the time.

After running OPRO for $20$ steps, using a maximum of $10$ previous instructions in the meta-prompt, we got the meta-prompt seen in Appendix E. The test set performance of the best-found instruction can be seen in Table 10 of Appendix E. From this table, we see that using OPRO, we were able to find an instruction that substantially outperforms the setting with no instruction; reaching a test set accuracy of $43\%$ as opposed to $33\%$. We do see a substantial difference between the *training set accuracy* and the test set accuracy of the

best-found instruction; $73\%$ vs. $43\%$. This indicates the presence of overfitting, which could be tackled by increasing the training set size.

*N.B.: The results of this experiment are not comparable to the results in Section 5.2.6 due to the use of different prompting setups, different models, different evaluation set sizes, and a fully automated evaluation.*

## 5.3 Statutory article retrieval observations

If we take a closer look at the lexical retrieval results, we see that some articles are harder to retrieve than others. To understand why, we will showcase the question with the worst AP score in the lexical retrieval experiments. For this question, its ground truth answer refers to a single relevant article.

**Question**

> "Geef aan of de stelling JUIST of ONJUIST is en motiveer uw antwoord in maximaal 5 regels. Stelling 1: De reële overeenkomst vormt een uitzondering op het aan het overeenkomstenrecht ten grondslag liggende beginsel van het consensualisme."

**Relevant article**

> 3:33 BW - "Een rechtshandeling vereist een op een rechtsgevolg gerichte wil die zich door een verklaring heeft geopenbaard."

In this example, it is clear that lexical retrieval would not be successful as there is marginal overlap between the texts. Alternatively, in dense retrieval, the example does not lead to a substantially better performance (the article was ranked 1755/3057). This can partly be explained by the fact that the first sentence of the question is not semantically related to the relevant article. Up next we would like to showcase the question which had the second-worst AP in dense retrieval. We have chosen the second worst AP question as opposed to the worst AP question (very long text) to exclude the possibility of the sequence length being the bottleneck. The ground truth answer to this question also refers to a single relevant article.

**Question**

> "Bank B heeft een vordering van € 800.000 op A. A heeft tot zekerheid van betaling een stil pandrecht op zijn (A's) dure Stradivariusviool gevestigd ten behoeve van B. Wat gebeurt er met het pandrecht van B indien B zijn vordering op A verkoopt en levert aan C?"

**Relevant article**

> 3:82 BW - "Afhankelijke rechten volgen het recht waaraan zij verbonden zijn."

In dense retrieval, a bad AP score means that the cosine similarity between the vector representations of the query and relevant document(s) had to be smaller than between the query and *irrelevant* documents. This means that the vector representations of the query and relevant document(s) were relatively *far* apart in the vector space. As such, the dense retriever did not succeed at mapping this question and article close to each other; affirming the apparent necessity to fine-tune the model on domain-specific data to create better embeddings.

N.B.: We notice that book 7A of the Dutch Civil Code contains articles in old Dutch. This could also influence the results of the IR experiments.

## 5.4 Legal (open-ended) question-answering observations

From our QA experiments, we see that the *some* metrics are able to clearly measure the relative quality between different prompting approaches. For instance, looking at the article-precision and article-recall metrics, we saw a great variety of scores over the experiments. This, however, contrasts with the $F_1$ BERTScores and Rouge-L metrics, whose scores hardly varied at all. But relative quality differences aside, it is hard to say how well all these metrics reflect the absolute quality of the responses.

One thing that we would like to discuss here is the **bias** present in these metrics. We see from Figure 15 in Appendix C that the text length of the ground truth answers is positively correlated with many of the automated metrics (e.g. BLEU, ROUGE scores) used in the **zero-shot prompting** setting with **gpt-3.5-turbo-0613**. Moreover, if we then look at the ratio between the response lengths and the ground truth lengths, we see that this ratio is negatively correlated with the automated metrics. This shows that there is a bias in these metrics; notably, the metrics tend to 1) favour questions with long (ground truth) answers and 2) disfavour questions where the response length of the model is longer than the ground truth answer. These biases are apparent in all the tested GPT models using the zero-shot prompting setting.

To mitigate the effects of the latter bias, we hypothesize that one could try to specify tailored response length indications in the prompt, to elicit the models to give responses that are similar in length to the corresponding ground truth answers.

Another problem occurs when LLMs incorporate the 'protect' phrases that we mentioned in Section 5.2.1. As such phrases do not occur in the ground truth answers, they negatively affect the automated performance metrics used. We speculate that this tendency to cover its base was learned in the RLHF training step (see Section 2.4). Once again, we hypothesize that one could try to tune the prompt to avoid this LLM behaviour.

When we dive into the questions across the LLM experiments where the article-precision is lower than the article-recall scores, we see that sometimes the LLM refers to statutory articles that are (somewhat) relevant, but which do not occur in the ground truth answers. In addition, sometimes, for some reason, an article is mentioned in the question but not in the ground truth answer. Thus, if an LLM repeats such articles in its answer, it unjustly leads to a worse article-precision score. Consequently, we would recommend the use of **article-recall** over article-precision in evaluating legal QA.

We deem a full manual self-evaluation of the LLM responses as impractical as the grading would not be objective. Especially in the case of incomplete or slightly wrong answers, deciding on the number of partial credits given introduces subjectivity. Due to the sheer amount of LLM responses in our experiments, it would also take a tremendous effort. However, we do want to give the reader an idea about the types of errors to expect. Therefore, we will list some notable examples in the next section.

*Lastly, from empirical observations, we believe that the answers produced by the GPT-4 models are of a generally higher quality than the answers produced by the GPT-3.5-turbo models; at least in the zero-shot setting. We believe this improvement is correlated with the higher article-recall measured in the GPT-4 experiments. Consequently, we hypothesize that the article-recall metric can act as a reasonable proxy for indicating the quality of an answer.*

## 5.5 Error analysis of (open-ended) legal question-answering

**Open norms in statutory articles**: occasionally we see references to statutory articles that have a very broad interpretation. An example of this is, is article 6:75 BW which roughly translated states:

> "A non-performance cannot be attributed to the debtor if he is not to blame for it nor accountable for it by virtue of law, a juridical act or generally accepted principles (common opinion)."

For one particular question, the conclusion in the answer should be based on these 'generally accepted principles'. In the respective LLM responses across the experiments, we did not encounter an instance where the LLM correctly reasons according to these generally accepted principles.

*Hence, we speculate that current state-of-the-art LLMs are not able to fully grasp and apply such 'open norms'.*

**Citing precedents**: a second observation we have made, is that references to precedents are vulnerable to mistakes. In the following question, a certain legal rule is asked, as well as the rule's origin in case law:

> "Stel een vennootschap onder firma wordt door opzegging ontbonden. Wat is volgens de Hoge Raad het peilmoment voor het bepalen van de waardeverandering van de goederen waarvan het economisch eigendom is ingebracht? Noem in uw antwoord ook de naam van het arrest."

In the following LLM response, the legal rule is correctly recalled, but its supposed origin is wrong:

> "Het peilmoment voor het bepalen van de waardeverandering van de goederen waarvan het economisch eigendom is ingebracht, is het tijdstip van de ontbinding van de vennootschap onder firma. Dit is bepaald in het arrest van de Hoge Raad van 10 januari 1997, NJ 1997/360 (Bruil/Kluin)."

We see two mistakes in this example. In the first place, the case 'HR 10 januari 1997, NJ 1997/360' has the alias 'Staleman/Van de Ven' as opposed to 'Bruil/Kluin'. Second, the correct origin of the legal rule is 'Faanse Veehouderijmaatschap', which is the alias of the case 'ECLI:NL:HR:2010:BL1127'.

*We see multiple mistakes like these across the experiments. We hypothesize that these hallucinations could be tackled by a RAG approach that also includes relevant precedents into the prompt.*

**Wrong article interpretation**: especially in the experiments resulting in high article-recall scores, the relevant articles are mentioned but they are incorrectly interpreted/applied. As correctly answering the questions in our dataset often requires multiple reasoning steps, there are many moments for LLMs to generate an incorrect reasoning.

**Wrong/missing articles**: relatively often across experiments, we do not see any statutory article mentioned and whenever they are, they are not always relevant.

### 5.5.1 Article-recall distribution over articles

Across the 300 legal questions we have included in our experiments, we see 267 unique relevant articles. We have visualized the article-recall distribution over these articles for several tested prompting methods in Figures: 16, 17, 18, 19, 20, 21, 22, and 23 of Appendix C.2. The tested prompting methods include the baseline zero-shot setting, the three few-shot settings of Table 5, the seven CoT settings of Table 4, and both RAG settings from Table 6. The model used is **gpt-3.5-turbo-1106**.

We can see from the heatmaps in the figures that some statutory articles are substantially harder to recall for LLMs than others. We hypothesize that the differences in article-recall scores over articles are related to the characteristics of the GPT (pre-)training data.

*The heatmaps show that a RAG system might especially help LLMs in recalling statutory articles that would otherwise not have been recalled using other prompting approaches, e.g. zero-shot, few-shot, or CoT prompting.*

# 6 Discussion

In line with previous findings in literature[27] [41, p.142, p.157], we can affirm that using pre-trained bi-encoder models without task-specific fine-tuning leads to bad results. Moreover, we believe that using such pre-trained models, that have been trained on general corpora, on very specific domains like Dutch Civil Law, exacerbates this outcome. In addition, the dense retriever used was only trained on sequences up to 128 tokens but the statutory articles and legal questions in our IR pipeline were of generally longer lengths; contributing to the bad performance. Chunking the questions and articles into smaller segments before indexing could tackle this issue of long text lengths, but we speculate that the 'somewhat arbitrary' cut-off points in chunking might cause different performance issues. Likewise, lexical retrieval has also performed worse than expected. The **vocabulary mismatch problem**, extensively described in Section 2.2.1, undeniably plays a role in this.

As the first-stage retrieval did not perform as expected (low $recall\_100$ and $recall\_1000$ scores), implementing a cross-encoder-based re-ranker was not deemed useful. As cross-encoders take in concatenated questions and articles, the limited maximum sequence lengths of such models would also pose a problem; given the large text lengths of questions and articles combined.

Regarding the evaluation of open-ended legal questions, in line with previous work [69], we conclude that neither the Rouge-L metric nor the BERTScore metrics can fully reflect human judgement. During the data extraction process, we have seen that *some* model answers are more like evaluation guidelines than answers while others are closer to being (one of many) examples of good answers. In addition, we have found that across questions these evaluation guides may substantially vary. Hence, a uniform approach to compare LLM answers to these model answers might never be optimal. For the multiple-choice legal questions, evaluation using the accuracy metric is not inherently an issue.

In spite of this, in the multiple-choice question-answering experiments, we struggled to elicit single-answer responses from LLMs. This turned out to be a problem in the automated prompt optimization experiments using OPRO, but it could mostly be solved by using a 2-step prompting approach. Another problem in OPRO was the presence of overfitting. This can be tackled by increasing the training set size however, one should be cautious with this as it would lead to substantially higher computational efforts. The number of computations would increase according to $O(xyq)$ with $x$ being the number of optimization steps, $y$ being the number of generated instructions per step, and $q$ being the number of questions in the training set. Furthermore, in setting up OPRO, there are many design choices to be made e.g.: optimization steps, training/test set ratio, the maximum number of previous instructions in the prompt, and the location of the instruction in the prompt. All of these choices can influence the performance of OPRO prompt optimization.

Lastly, we have encountered many difficulties in creating and curating the dataset with legal questions. There was a substantial effort needed to pre-process the questions as it required a lot of manual pre-processing. Some notable issues included: the diverse formats of exams, handling (long) follow-up questions, and fixing decoding errors.

[27] "... there is existing experimental evidence demonstrating that dense retrieval techniques are often ineffective in a zero-shot transfer setting to texts in different domains, different types of queries, etc. "

# 7 Conclusion

Before we answer our main research question, we will first go over the five sub-questions that we have asked ourselves at the start of this research.

> *What methods can be used to at forehand retrieve the relevant statutory articles to legal exam questions?*

To retrieve statutory articles, we have experimented with lexical retrieval, dense retrieval, and a hybrid combination of these. We implemented these methods in the Weaviate vector database. As previously discussed in Section 6, we have found the performance of all tested methods to be underwhelming. Therefore, we would specifically recommend the **fine-tuning** of a task-specific bi-encoder for the dense retrieval of statutory articles in future related research. Additionally, one could try to use techniques like query expansion and document expansion (see Section 2.2.1) to improve upon the lexical retrieval.

> *Given the relevant statutory articles to a legal question as context, to what extent are large language models able to provide the correct answer?*

We have found that when using an LLM in the RAG oracle setting, all evaluation metrics are improved as compared to the setting without RAG. From empirical observations, we have noticed that in RAG the responses tend to incorporate phrases from the included statutory articles in its answer. This corresponds to stating the *rule* in the IRAC methodology (see Section 2.4.4). Generally, the ground truth answers to legal exam questions are constructed according to this methodology; likely explaining the improved metrics. However, it is unclear if the improved metrics also correspond to a correct *application* of the legal rules.

> *What prompting techniques can be used to improve their ability to answer these questions?*

We have experimented with several prompting techniques in this work; notably **few-shot**, **zero-shot Chain-of-Thought**, and **RAG** prompting. We have found **positive effects** when using these methods over zero-shot prompting as they led to improvements in almost all used metrics. Furthermore, we have also experimented with using a majority vote system in the multiple-choice QA using a temperature of $1.0$. This did not seem to perform better over doing a single run with a temperature of $0.0$. In general, we would recommend a temperature of $0.0$ for both open-ended QA and multiple-choice QA.

> *What aspects should be evaluated, and what evaluation metrics should be used, in order to objectively evaluate their performance?*

The automated metrics that we use, evaluate different aspects of the responses. The BLEU and ROUGE metrics are based on word overlap, while semantic similarity is measured using the BERTScore metrics. In addition, we use the metrics article-precision and article-recall to measure the LLM's ability to refer to relevant statutory articles in their answer. Consequently, we generally do not see a big difference in performance between the different models and prompting techniques if we solely go off by the BLEU, ROUGE and $F_1$ BERTScores, but we *do* see a big difference in article-precision and article-recall scores. These latter metrics lend themself well to measure the **explainability** of the answers.

However, they do not fully reflect the *quality* of an answer; only relevance to some degree. Therefore, we believe that **human evaluation** by an independent party is the best and most objective way to evaluate answers in open-ended legal QA. For multiple-choice question-answering, **accuracy** seems to be the most logical metric. However, one thing to note is that LLMs are not guaranteed to treat all single-option multiple-choice questions correctly (e.g. picking multiple options or treating them as open-ended questions). We hypothesize that this behaviour may be mitigated by using few-shot prompting.

With all of this in mind, we will now hope to answer our main research question:

> *How successful are current large language models at answering questions regarding Dutch Civil Law?*

We believe that current state-of-the-art LLMs are not yet ready for legal QA as we still see quite a few errors when we manually inspect some of the answers (see Section 5.5). Nevertheless, we have found a big performance difference between the GPT-3.5-turbo and GPT-4 models regarding referencing relevant statutory articles in their answers. Therefore, in light of these quick advances between subsequent GPT generations, we have high hopes for the next generation of LLMs. In addition, we believe that the evolution of novel prompting techniques may help in that regard. Lastly, we believe it is also important for the scientific community to focus on developing the right benchmarks and evaluation methods. Without them, it is hard to measure LLM-performance in legal QA.

## 7.1 Future work

Regarding future work in legal QA using LLMs, it would be interesting to test a more diverse range of models. Given the closed-source nature of the LLMs used in this research, we can not objectively match the outcomes of this paper to any specific: architectural component, training data, or general characteristic of the models used. This would be different if open-sourced models had been used. The Dutch open-source LLM '**GEITje**' [111], released at the end of 2023, could prove to be a suitable candidate for any research related to Dutch QA in the future.

In addition, one might consider **expanding the dataset** with exam questions from courses outside the civil law domain; possibly by also looking at courses from other universities teaching law. The creation of a more comprehensive open-source legal QA benchmark could help accelerate the development of LLM use cases in the legal field. As discussed in Section 2.4.4, assigning a role to the LLM can improve the performance. The OpenAI models have a setting in their API called '**system role**'. Using this setting, one can assign a certain role/identity to the LLM when asking it a question. It would be interesting to research how one could best define such a role for legal QA, and how it would affect the performance of the system. In addition, this research primarily focused on testing several prompting techniques individually. It would be interesting to see how a **mixture** of these techniques would affect performance. Furthermore, future research could look at a more **fine-grained evaluation** of statutory articles, e.g. 'art. 1:5 lid 5 sub b BW' instead of 'art. 1:5 BW'. In addition, the RAG pipeline could be extended by incorporating **case law** in the IR components.

Lastly, one promising new approach to evaluating answers in QA is by using LLMs themselves to judge the quality of the answers; removing the need for expensive human evaluation. For more information, we refer the reader to recent work [112] on this.

# References


[1] D. Jurafsky and J. Martin, "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition 3rd ed. draft," 01 2023.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

[3] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., "Improving language understanding by generative pre-training," 2018.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language models are few-shot learners," Advances in neural information processing systems, vol. 33, pp. 1877–1901, 2020.

[5] R. Bommasani, D. A. Hudson, E. Adeli, R. B. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. S. Chatterji, A. S. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. D. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. S. Krass, R. Krishna, R. Kuditipudi, and et al., "On the Opportunities and Risks of Foundation Models," CoRR, vol. abs/2108.07258, 2021.

[6] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A Survey of Large Language Models," 2023.

[7] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," ACM Computing Surveys, vol. 55, no. 12, pp. 1–38, 2023.

[8] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al., "Training language models to follow instructions with human feedback," Advances in Neural Information Processing Systems, vol. 35, pp. 27730–27744, 2022.

[9] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT," 2023.

[10] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy, "Challenges and Applications of Large Language Models," 2023.

[11] S. Dhuliawala, M. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz, and J. Weston, "Chain-of-verification reduces hallucination in large language models," arXiv preprint arXiv:2309.11495, 2023.

[12] OpenAI, "GPT-4 Technical Report," 2023.

[13] D. M. Katz, M. J. Bommarito, S. Gao, and P. Arredondo, "Gpt-4 passes the bar exam," Available at SSRN 4389233, 2023.

[14] L. Tunstall, L. von Werra, and T. Wolf, "Natural Language Processing with Transformers: Building Language Applications with Hugging Face," 2022.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," Advances in neural information processing systems, vol. 27, 2014.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

[17] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," 2016.

[18] J. Alammar, "The illustrated transformer," The Illustrated Transformer–Jay Alammar–Visualizing Machine Learning One Concept at a Time, vol. 27, 2018.

[19] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," AI Open, 2022.

[20] B. Newman, J. Hewitt, P. Liang, and C. D. Manning, "The EOS Decision and Length Extrapolation," in Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, pp. 276–291, 2020.

[21] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating Long Sequences with Sparse Transformers," CoRR, vol. abs/1904.10509, 2019.

[22] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The Long-Document Transformer," CoRR, vol. abs/2004.05150, 2020.

[23] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The Efficient Transformer," CoRR, vol. abs/2001.04451, 2020.

[24] O. Press, N. Smith, and M. Lewis, "Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation," in International Conference on Learning Representations, 2021.

[25] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in 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) (J. Burstein, C. Doran, and T. Solorio, eds.), pp. 4171–4186, Association for Computational Linguistics, 2019.

[26] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, H. Peng, J. Li, J. Wu, Z. Liu, P. Xie, C. Xiong, J. Pei, P. S. Yu, and L. Sun, "A Comprehensive Survey on Pretrained Foundation Models: A History from BERT to ChatGPT," 2023.

[27] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," CoRR, vol. abs/1907.11692, 2019.

[28] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," CoRR, vol. abs/1909.11942, 2019.

[29] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth, "Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey," ACM Comput. Surv., vol. 56, sep 2023.

[30] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., "Language models are unsupervised multitask learners," OpenAI blog, vol. 1, no. 8, p. 9, 2019.

[31] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al., "Palm: Scaling language modeling with pathways," Journal of Machine Learning Research, vol. 24, no. 240, pp. 1–113, 2023.

[32] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., "Llama: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971, 2023.

[33] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," Journal of machine learning research, vol. 21, no. 140, pp. 1–67, 2020.

[34] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2020.

[35] Y. Tay, M. Dehghani, V. Q. Tran, X. Garcia, J. Wei, X. Wang, H. W. Chung, D. Bahri, T. Schuster, S. Zheng, et al., "UL2: Unifying Language Learning Paradigms," in The Eleventh International Conference on Learning Representations, 2022.

[36] C. D. Manning, An introduction to information retrieval. Cambridge university press, 2009.

[37] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models," in Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021.

[38] B. Miutra and N. Craswell, "An Introduction to Neural Information Retrieval ," Found. Trends Inf. Retr., vol. 13, p. 1–126, dec 2018.

[39] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, "The Vocabulary Problem in Human-System Communication," Commun. ACM, vol. 30, p. 964–971, nov 1987.

[40] V. Balakrishnan and L.-Y. Ethel, "Stemming and Lemmatization: A Comparison of Retrieval Performances," Lecture Notes on Software Engineering, vol. 2, pp. 262–267, 01 2014.

[41] J. Lin, R. Nogueira, and A. Yates,
Pretrained transformers for text ranking: Bert and beyond. Springer Nature, 2022.

[42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," Advances in neural information processing systems, vol. 26, 2013.

[43] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543, 2014.

[44] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep Contextualized Word Representations," in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (M. Walker, H. Ji, and A. Stent, eds.), (New Orleans, Louisiana), pp. 2227–2237, Association for Computational Linguistics, June 2018.

[45] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," Transactions of the association for computational linguistics, vol. 8, pp. 64–77, 2020.

[46] R. F. Nogueira and K. Cho, "Passage Re-ranking with BERT," CoRR, vol. abs/1901.04085, 2019.

[47] J. Guo, Y. Cai, Y. Fan, F. Sun, R. Zhang, and X. Cheng, "Semantic models for the first-stage retrieval: A comprehensive review," ACM Transactions on Information Systems (TOIS), vol. 40, no. 4, pp. 1–42, 2022.

[48] R. Nogueira, W. Yang, K. Cho, and J. Lin, "Multi-stage document ranking with BERT," arXiv preprint arXiv:1910.14424, 2019.

[49] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, 2019.

[50] N. Muennighoff, "SGPT: GPT Sentence Embeddings for Semantic Search," 2022.

[51] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "MS MARCO: A human generated machine reading comprehension dataset," choice, vol. 2640, p. 660, 2016.

[52] M. Bevilacqua, G. Ottaviano, P. Lewis, S. Yih, S. Riedel, and F. Petroni, "Autoregressive search engines: Generating substrings as document identifiers," Advances in Neural Information Processing Systems, vol. 35, pp. 31668–31683, 2022.

[53] G. Bénédict, R. Zhang, and D. Metzler, "Gen-ir@ sigir 2023: The first workshop on generative information retrieval," in Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 3460–3463, 2023.

[54] A. Louis, G. van Dijck, and G. Spanakis, "Finding the Law: Enhancing Statutory Article Retrieval via Graph Neural Networks," in Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (A. Vlachos and I. Augenstein, eds.), (Dubrovnik, Croatia), pp. 2761–2776, Association for Computational Linguistics, May 2023.

[55] J. Rossi and E. Kanoulas, "Legal Search in Case Law and Statute Law.," in JURIX, pp. 83–92, 2019.

[56] P. Tiersma, The nature of legal language, vol. 5. John Benjamins Publishing Amsterdam/Philadelphia, 2008.

[57] A. Askari, S. Verberne, O. Alonso, S. Marchesin, M. Najork, and G. Silvello, "Combining Lexical and Neural Retrieval with Longformer-based Summarization for Effective Case Law Retrieval.," in DESIRES, pp. 162–170, 2021.

[58] A. Askari, M. Aliannejadi, A. Abolghasemi, E. Kanoulas, and S. Verberne, "CLosER: Conversational Legal Longformer with Expertise-Aware Passage Response Ranker for Long Contexts," in Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, pp. 25–35, 2023.

[59] C. Sansone and G. Sperlí, "Legal Information Retrieval systems: State-of-the-art and open issues," Information Systems, vol. 106, p. 101967, 2022.

[60] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "LEGAL-BERT: The Muppets straight out of Law School," in Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 2898–2904, 2020.

[61] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A Pretrained Language Model for Scientific Text," in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3615–3620, 2019.

[62] C. Sanchez and Z. Zhang, "The Effects of In-domain Corpus Size on pre-training BERT," 2022.

[63] A. Louis and G. Spanakis, "A Statutory Article Retrieval Dataset in French," in Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Volume 1: Long Papers, pp. 6789–6803, Association for Computational Linguistics, 2022.

[64] A. Chen, F. Yao, X. Zhao, Y. Zhang, C. Sun, Y. Liu, and W. Shen, "EQUALS: A real-world dataset for legal question answering via reading chinese laws," in Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law, pp. 71–80, 2023.

[65] A. Bock, "Gütezeichen als Qualitätsaussage im digitalen Informationsmarkt: dargestellt am Beispiel elektronischer Rechtsdatenbanken," 2000.

[66] G. Wiggers, S. Verberne, W. van Loon, and G.-J. Zwenne, "Bibliometric-enhanced legal information retrieval: Combining usage and citations as flavors of impact relevance," Journal of the Association for Information Science and Technology, vol. 74, no. 8, pp. 1010–1025, 2023.

[67] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks," Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.

[68] Q. Huang, M. Tao, C. Zhang, Z. An, C. Jiang, Z. Chen, Z. Wu, and Y. Feng, "Lawyer LLaMA Technical Report," 2023.

[69] Z. Fei, X. Shen, D. Zhu, F. Zhou, Z. Han, S. Zhang, K. Chen, Z. Shen, and J. Ge, "LawBench: Benchmarking Legal Knowledge of Large Language Models," arXiv preprint arXiv:2309.16289, 2023.

[70] N. Guha, J. Nyarko, D. Ho, C. Ré, A. Chilton, A. Chohlas-Wood, A. Peters, B. Waldon, D. Rockmore, D. Zambrano, et al., "Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models," Advances in Neural Information Processing Systems, vol. 36, 2024.

[71] I. Chalkidis, A. Jana, D. Hartung, M. Bommarito, I. Androutsopoulos, D. Katz, and N. Aletras, "LexGLUE: A Benchmark Dataset for Legal Language Understanding in English," in Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 4310–4330, 2022.

[72] A. Blair-Stanek, N. Holzenberger, and B. Van Durme, "Can GPT-3 Perform Statutory Reasoning?," in Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law, ICAIL '23, (New York, NY, USA), p. 22–31, Association for Computing Machinery, 2023.

[73] J. Cui, Z. Li, Y. Yan, B. Chen, and L. Yuan, "Chatlaw: Open-source legal large language model with integrated external knowledge bases," arXiv preprint arXiv:2306.16092, 2023.

[74] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," ACM Computing Surveys, vol. 55, no. 9, pp. 1–35, 2023.

[75] V. Liévin, C. E. Hother, and O. Winther, "Can large language models reason about medical questions?," arXiv preprint arXiv:2207.08143, 2022.

[76] R. Schaeffer, B. Miranda, and S. Koyejo, "Are emergent abilities of large language models a mirage?," Advances in Neural Information Processing Systems, vol. 36, 2024.

[77] D. Dai, Y. Sun, L. Dong, Y. Hao, S. Ma, Z. Sui, and F. Wei, "Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers," in Findings of the Association for Computational Linguistics: ACL 2023, pp. 4005–4019, 2023.

[78] Y. Fu, H. Peng, L. Ou, A. Sabharwal, and T. Khot, "Specializing Smaller Language Models towards Multi-Step Reasoning," in Proceedings of the 40th International Conference on Machine Learning, ICML'23, JMLR.org, 2023.

[79] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., "Chain-of-thought prompting elicits reasoning in large language models," Advances in Neural Information Processing Systems, vol. 35, pp. 24824–24837, 2022.

[80] R. Ali, O. Y. Tang, I. D. Connolly, J. S. Fridley, J. H. Shin, P. L. Z. Sullivan, D. Cielo, A. A. Oyelese, C. E. Doberstein, A. E. Telfeian, et al., "Performance of ChatGPT, GPT-4, and Google bard on a neurosurgery oral boards preparation question bank," Neurosurgery, pp. 10–1227, 2022.

[81] O. Khattab, K. Santhanam, X. L. Li, D. Hall, P. Liang, C. Potts, and M. Zaharia, "Demonstrate-Search-Predict: Composing Retrieval and Language Models for Knowledge-Intensive NLP," arXiv preprint arXiv:2212.14024, 2022.

[82] T. Wu, M. Terry, and C. J. Cai, "Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts," in Proceedings of the 2022 CHI conference on human factors in computing systems, pp. 1–22, 2022.

[83] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J.-R. Wen, "A Survey on Large Language Model based Autonomous Agents," 2023.

[84] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," Advances in neural information processing systems, vol. 35, pp. 22199–22213, 2022.

[85] F. Yu, L. Quartey, and F. Schilder, "Legal Prompting: Teaching a Language Model to Think Like a Lawyer," 2022.

[86] S. J. Zhang, S. Florin, A. N. Lee, E. Niknafs, A. Marginean, A. Wang, K. Tyser, Z. Chin, Y. Hicke, N. Singh, M. Udell, Y. Kim, T. Buonassisi, A. Solar-Lezama, and I. Drori, "Exploring the MIT Mathematics and EECS Curriculum Using Large Language Models," 2023.

[87] M. Zhuge, H. Liu, F. Faccio, D. R. Ashley, R. Csordás, A. Gopalakrishnan, A. Hamdi, H. A. A. K. Hammoud, V. Herrmann, K. Irie, L. Kirsch, B. Li, G. Li, S. Liu, J. Mai, P. Piekos, A. Ramesh, I. Schlag, W. Shi, A. Stanić, W. Wang, Y. Wang, M. Xu, D.-P. Fan, B. Ghanem, and J. Schmidhuber, "Mindstorms in Natural Language-Based Societies of Mind," 2023.

[88] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-Consistency Improves Chain of Thought Reasoning in Language Models," in The Eleventh International Conference on Learning Representations, 2023.

[89] V. Liévin, C. E. Hother, and O. Winther, "Can large language models reason about medical questions?," 2023.

[90] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," Advances in Neural Information Processing Systems, vol. 36, 2024.

[91] K. Singhal, T. Tu, J. Gottweis, R. Sayres, E. Wulczyn, L. Hou, K. Clark, S. Pfohl, H. Cole-Lewis, D. Neal, M. Schaekermann, A. Wang, M. Amin, S. Lachgar, P. Mansfield, S. Prakash, B. Green, E. Dominowska, B. A. y Arcas, N. Tomasev, Y. Liu, R. Wong, C. Semturs, S. S. Mahdavi, J. Barral, D. Webster, G. S. Corrado, Y. Matias, S. Azizi, A. Karthikesalingam, and V. Natarajan, "Towards Expert-Level Medical Question Answering with Large Language Models," 2023.

[92] J. H. Choi, K. E. Hickman, A. Monahan, and D. Schwarcz, "Chatgpt goes to law school," Available at SSRN, 2023.

[93] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, "Large Language Models as Optimizers," in The Twelfth International Conference on Learning Representations, 2024.

[94] Q. Guo, R. Wang, J. Guo, B. Li, K. Song, X. Tan, G. Liu, J. Bian, and Y. Yang, "Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers," in The Twelfth International Conference on Learning Representations, 2024.

[95] M. Van Opijnen and C. Santos, "On the concept of relevance in legal information retrieval," Artificial Intelligence and Law, vol. 25, pp. 65–87, 2017.

[96] C. Macdonald and N. Tonellotto, "Declarative Experimentation inInformation Retrieval using PyTerrier," in Proceedings of ICTIR 2020, 2020.

[97] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, et al., "Milvus: A Purpose-Built Vector Data Management System," in Proceedings of the 2021 International Conference on Management of Data, pp. 2614–2627, 2021.

[98] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," IEEE transactions on pattern analysis and machine intelligence, vol. 42, no. 4, pp. 824–836, 2018.

[99] M. Aumüller, E. Bernhardsson, and A. Faithfull, "ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms," Information Systems, vol. 87, p. 101374, 2020.

[100] Z. Mao and T. Nakagawa, "LEALLA: Learning Lightweight Language-agnostic Sentence Embeddings with Knowledge Distillation," in Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, (Dubrovnik, Croatia), pp. 1886–1894, Association for Computational Linguistics, May 2023.

[101] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic BERT Sentence Embedding," in Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 878–891, 2022.

[102] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, vol. 3, pp. 333–389, 01 2009.

[103] I. Chalkidis, M. Fergadiotis, N. Manginas, E. Katakalou, and P. Malakasiotis, "Regulatory Compliance through Doc2Doc Information Retrieval: A case study in EU/UK legislation where text similarity has limitations," in Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 3498–3511, 2021.

[104] C. Van Gysel and M. de Rijke, "Pytrec_eval: An Extremely Fast Python Interface to trec_eval," in SIGIR, ACM, 2018.

[105] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a Method for Automatic Evaluation of Machine Translation," in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (P. Isabelle, E. Charniak, and D. Lin, eds.), (Philadelphia, Pennsylvania, USA), pp. 311–318, Association for Computational Linguistics, July 2002.

[106] C. Lin, "Recall-oriented understudy for gisting evaluation (rouge)," Retrieved August, vol. 20, p. 2005, 2005.

[107] T. Zhang*, V. Kishore*, F. Wu*, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," in International Conference on Learning Representations, 2020.

[108] C.-Y. Lin and F. J. Och, "Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics," in Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04), pp. 605–612, 2004.

[109] L. Chen, Y. Deng, Y. Bian, Z. Qin, B. Wu, T.-S. Chua, and K.-F. Wong, "Beyond Factuality: A Comprehensive Evaluation of Large Language Models as Knowledge Generators," in Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 6325–6341, 2023.

[110] C. Wei, Y.-C. Wang, B. Wang, and C. C. J. Kuo, "An Overview on Language Models: Recent Developments and Outlook," 2023.

[111] E. Rijgersberg and B. Lucassen, "GEITje: een groot open Nederlands taalmodel," Dec. 2023.

[112] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al., "Judging llm-as-a-judge with mt-bench and chatbot arena," Advances in Neural Information Processing Systems, vol. 36, 2024.

# Appendix A Domain-specific CoT cues

These descriptions, usually on the front page of an exam, are instructions for the students on how to answer the exam questions. We see that some instructions mainly see to the student maintaining correct grammar and spelling, while others see to students using the correct answer methodology. The results of appending these instructions to the prompt in our QA pipeline can be seen in Section 5.2.3.

*N.B.: The crossed out text has been left out in the experiments, to make the instruction more general or because we deem it redundant.*

## A.1 Open-ended questions

1. NB: Bij de beoordeling speelt taalvaardigheid een rol. Zowel de structuur van het stuk, als correcte spelling en zinsbouw wordt meegewogen. ~~Taal- en spelfouten kunnen leiden tot een aftrek van maximaal 2 punten op het cijfer van de tussentoets.~~

2. De motivering van uw antwoord zal worden meegewogen bij de beoordeling. Motiveer uw antwoorden dus zorgvuldig; verwijs waar mogelijk naar de relevante wetartikelen en jurisprudentie.

3. De motivering van uw antwoord zal~~, juist vanwege het open-boek karakter van het tentamen,~~ een centrale rol spelen bij de beoordeling. Lees de vraag goed en motiveer uw antwoorden zorgvuldig. Dit betekent dat u goed inzicht in uw gedachtegang moet geven en waar mogelijk moet verwijzen naar de relevante wetartikelen en jurisprudentie. Er is geen maximum aantal woorden voor de beantwoording. Dit is geen uitnodiging tot het ongebreideld spuien van informatie en kennis. Overbodige of ongestructureerde informatie kan leiden tot puntenaftrek.

4. De beantwoording van de open vragen zal ~~dan ook op de bij IBR bekende wijze~~ gemotiveerd moeten plaatsvinden onder verwijzing naar wetsartikelen, wettelijke vereisten en rechtsregels die voortvloeien uit arresten van de Hoge Raad. In het onderwijs is ook duidelijk gewezen op de specifieke toetsing van de feiten aan dit juridisch kader. Deze toetsing zal nauwkeurig, uitgebreid en zelfstandig moeten plaatsvinden om de desbetreffende punten te kunnen behalen. Uw antwoord zal moeten resulteren in goed lopend coherent verhaal.

5. Alle antwoorden dienen te worden gemotiveerd. Ongemotiveerde antwoorden (bijvoorbeeld alleen "ja, nee, hoger, lager") worden gewaardeerd met nul punten.

6. Beantwoord de vragen kort en bondig. U dient uw antwoorden altijd te motiveren. Bij enkele vragen dient u dit te doen aan de hand van het zogenaamde stappenplan: 1. Wat zijn de relevante feiten in de casus 2. Wat is het probleem ofwel de rechtsvraag 3. Welke rechtsregel(s) zijn hiervoor van belang 4. Analyse van de rechtsregel(s) en toetsing van de feiten aan de rechtsregel(s) 5. Daaruit vervolgens een conclusie trekken.

7. 1. De casus en de vragen dienen zeer zorgvuldig te worden gelezen. Afwijking of verwaarlozing van het in de casus en vragen gestelde wordt als fout aangerekend. ~~Bij elk onderdeel is vermeld hoeveel punten u maximaal kunt scoren.~~ 2. Beantwoord de vragen gemotiveerd, duidelijk en rechtstreeks, en onder vermelding van relevante wetsartikelen

en jurisprudentie; bij de beoordeling wordt rekening gehouden met een juist gebruik van de Nederlandse taal. ~~Een onleesbaar handschrift komt voor eigen rekening.~~

## A.2 Multiple-choice questions

- U dient uit de vier keuzemogelijkheden het meest juiste antwoord te kiezen. Bij de beantwoording van de meerkeuzevragen ~~Ondernemingsrecht met een waarde van 2 punten~~ raden wij u aan het stappenplan te gebruiken om uw antwoord te controleren: 1. Wat zijn de relevante feiten in de casus; 2. Wat is het probleem ofwel de rechtsvraag; 3. Welke rechtsregel(s) zijn hiervoor van belang; 4. Analyse van de rechtsregel(s) en toetsing van de feiten aan de rechtsregel(s); 5. Daaruit vervolgens een conclusie trekken.

# Appendix B LLM zero-shot multiple-choice issues

## B.1 Abstention

An example of an LLM abstaining from answering, can be seen below:

**Question:**

> In een catalogus van C&A staat in een advertentie voor damesjurken de volgende tekst: "Deze jurk heeft net zo'n chique uitstraling als de jurken van het trendy label La Dress, maar is een stuk goedkoper!" Als IE-specialist kwalificeert u de uitlating als:
> (a) Ongerechtvaardigd voordeel trekken uit de bekendheid van een merk.
>
> (b) Slaafse nabootsing.
>
> (c) Een verveelvoudiging.
>
> (d) Verwarringwekkend.

**LLM response:**

> Als AI taalmodel kan ik u helpen met het begrijpen van de verschillende termen. In dit geval lijkt de uitlating te verwijzen naar een vergelijking met het trendy label La Dress, waarbij wordt beweerd dat de jurk een vergelijkbare uitstraling heeft maar goedkoper is. Gebaseerd op deze beschrijving zou ik de uitlating echter niet direct kunnen classificeren als een van de genoemde opties. Het lijkt erop dat de uitlating gebaseerd is op een vergelijking in termen van uiterlijk en prijs, maar zonder verdere context kan ik geen definitieve beoordeling geven. Het kan bijvoorbeeld zijn dat er sprake is van oneerlijke concurrentie of misleidende reclame, maar dit kan niet worden vastgesteld op basis van de verstrekte informatie.

## B.2 Other issues

An example of an LLM not going by single-label classification can be seen below:

**Question:**

> Welke van de onderstaande stellingen met betrekking tot levering, bezit en houderschap is onjuist?
> (a) Joost verliest het bezit van zijn gesigneerde voetbalshirt wanneer Martijn het bezit van dat voetbalshirt verkrijgt.
>
> (b) Wanneer Mieke een grasmaaier steelt, is zij bezitter van de grasmaaier geworden doordat zij zich daarover de feitelijke macht heeft verschaft.
>
> (c) Arjan, die onmiddellijk houder is van de spelcomputer van Pieter, kan zichzelf onmiddellijk bezitter maken door de spelcomputer voor zichzelf te gaan houden.
>
> (d) Het overhandigen van de sleutel bij de overdracht van een mountainbike is een voorbeeld van een traditio symbolica.

**LLM response:**

> Alle stellingen zijn juist, er is geen onjuiste stelling tussen deze opties.

# Appendix C Additional figures and tables

## C.1 Figures

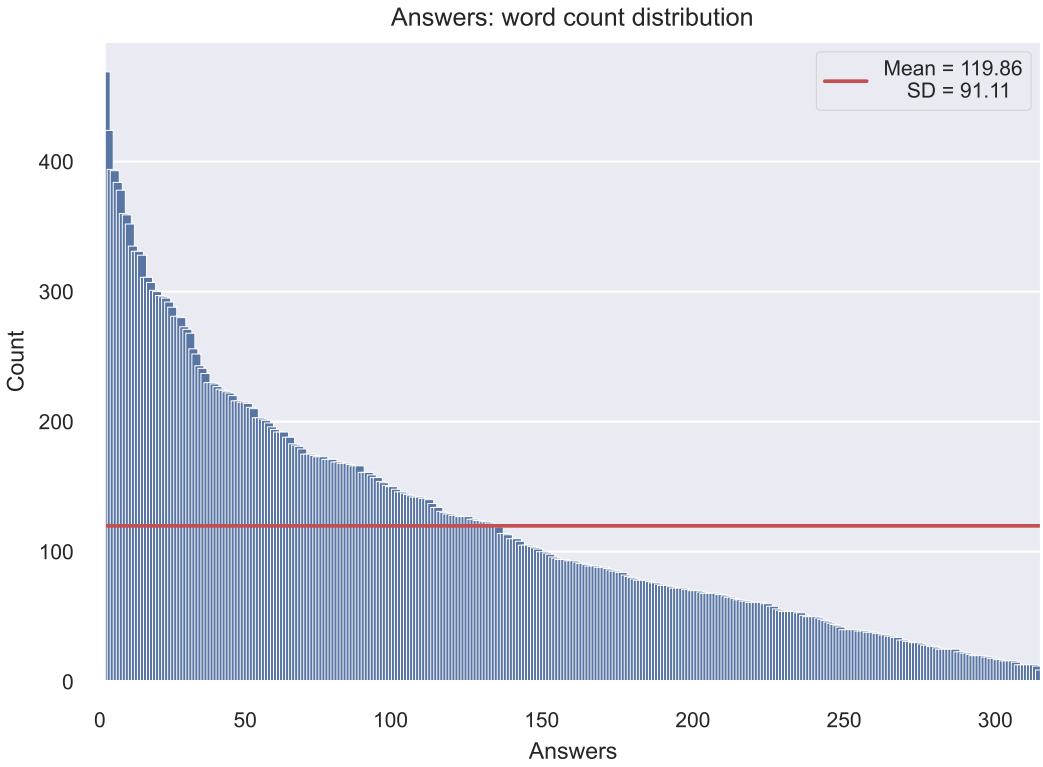


Figure 12: Word count distribution for the model answers.

**Boek 3. Vermogensrecht in het algemeen**

**Titel 1. Algemene bepalingen**

**Afdeling 1. Begripsbepalingen**

**Artikel 1**

Goederen zijn alle zaken en alle vermogensrechten.

Figure 13: Contents article 3:1 BW[28]

[28] `https://wetten.overheid.nl/BWBR0002656/2023-07-01`)

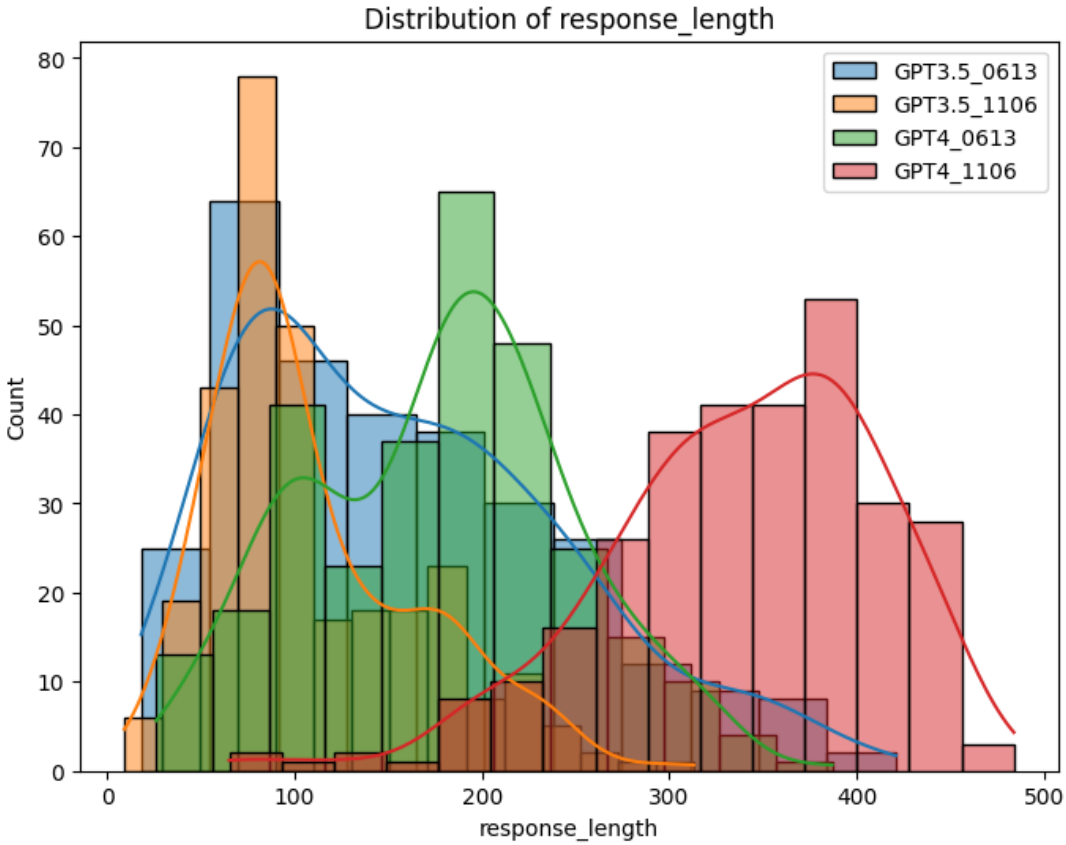


Figure 14: Response length differences between different models in zero-shot prompting.

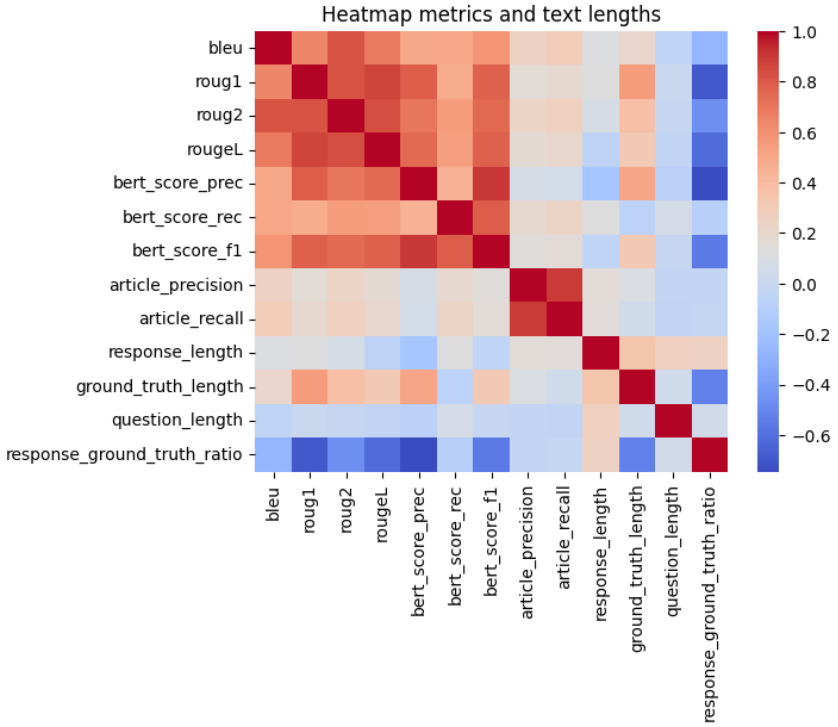


Figure 15: Heatmap of metrics and response/question/ground_truth text lengths for zero-shot QA using gpt-3.5-turbo-0613. The 'response_ground_truth_ratio' refers to the response length divided by the ground truth answer length per question.

## C.2 Article-recall across prompting techniques, books 1-7A

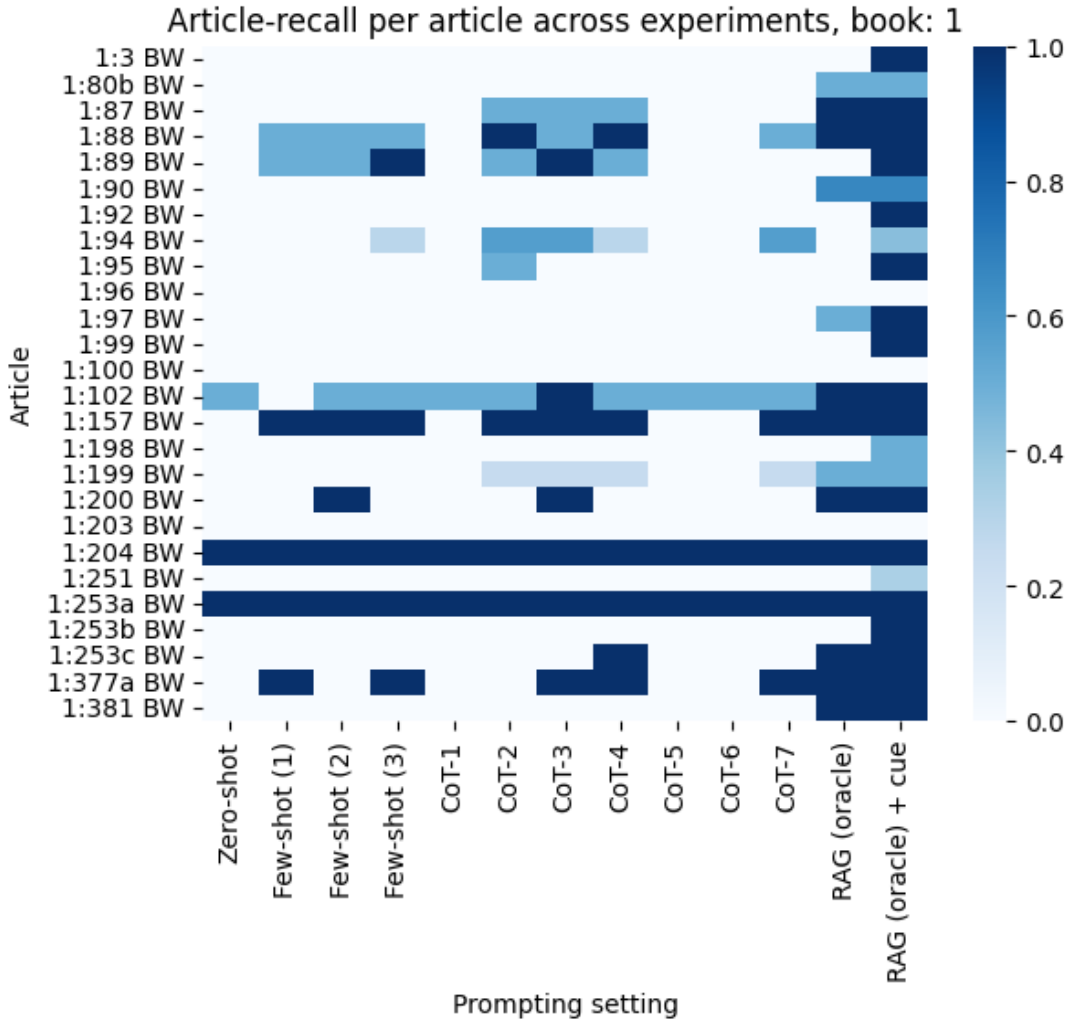


Figure 16: Article-recall per article across experiments, book 1

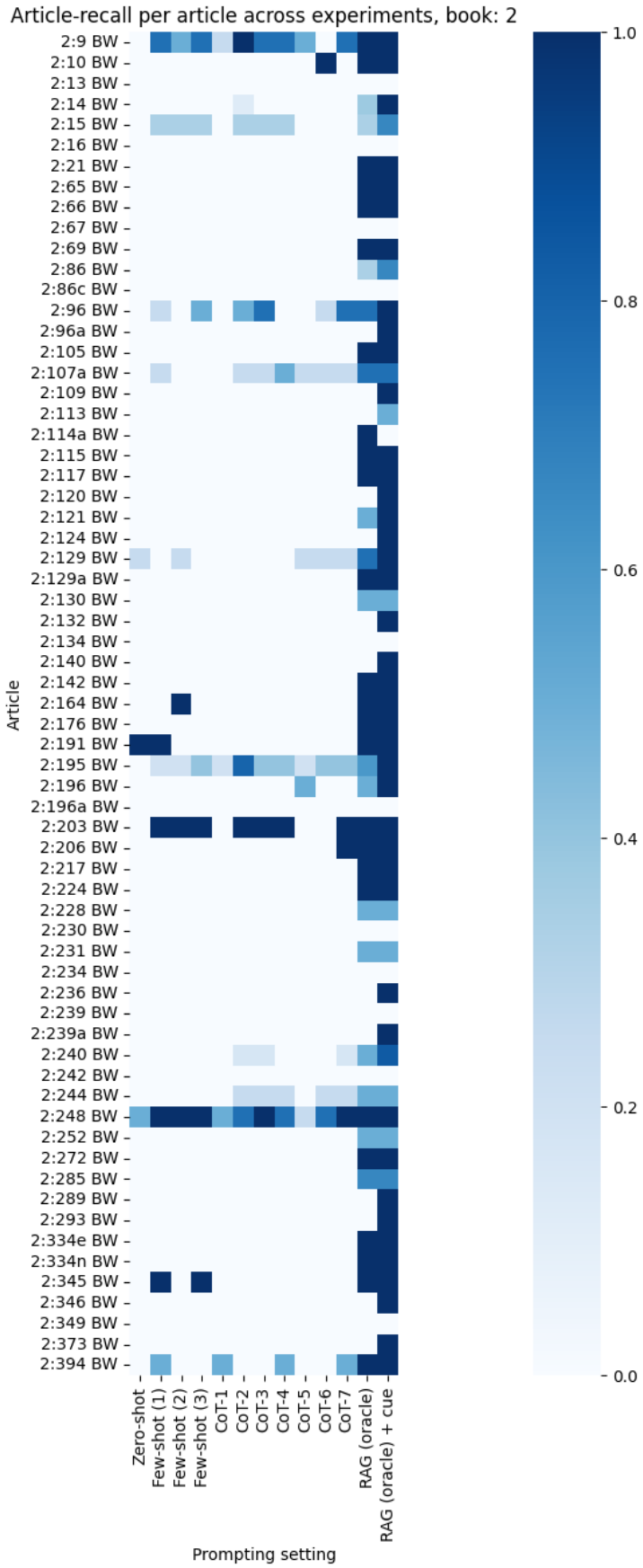


Figure 17: Article-recall per article across experiments, book 2

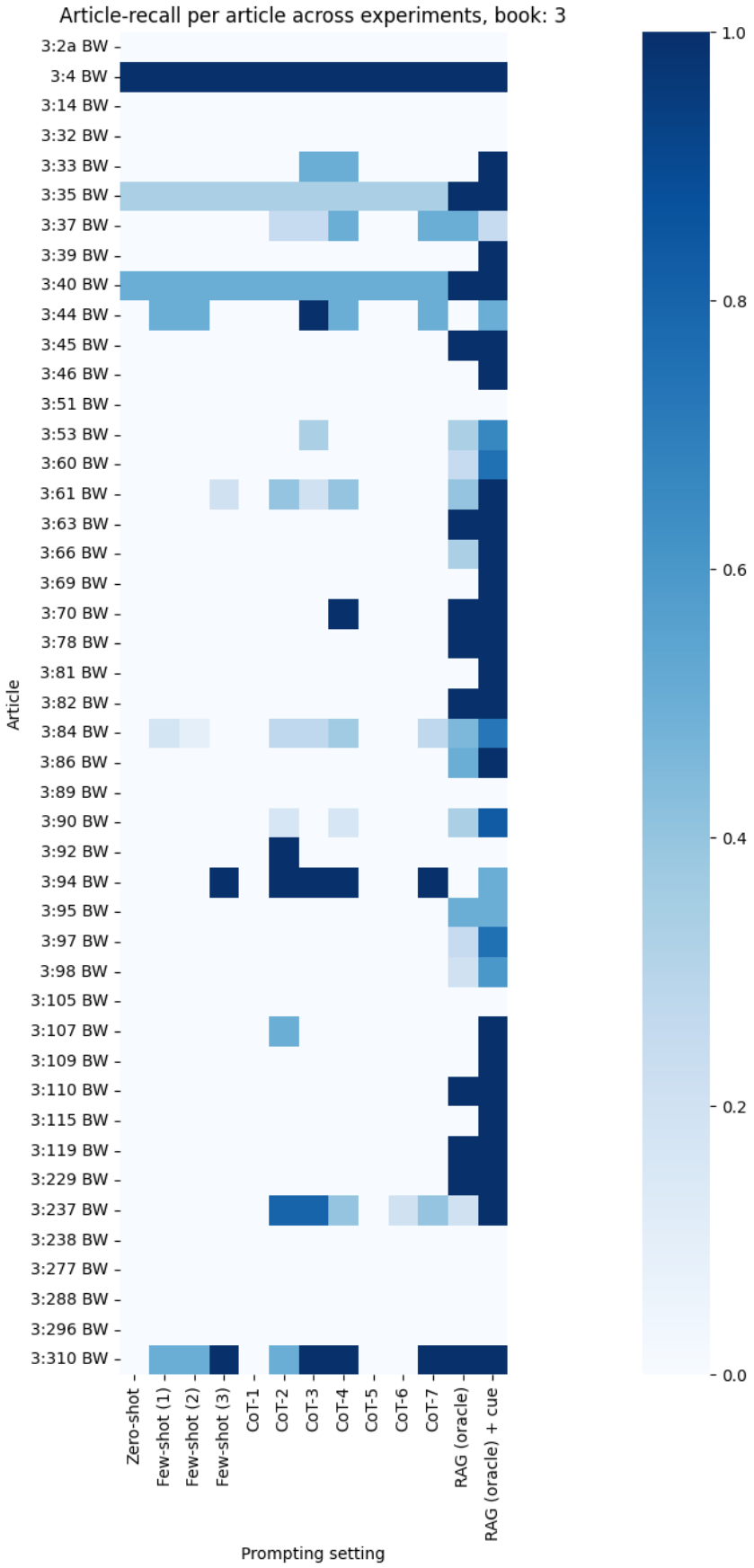


Figure 18: Article-recall per article across experiments, book 3

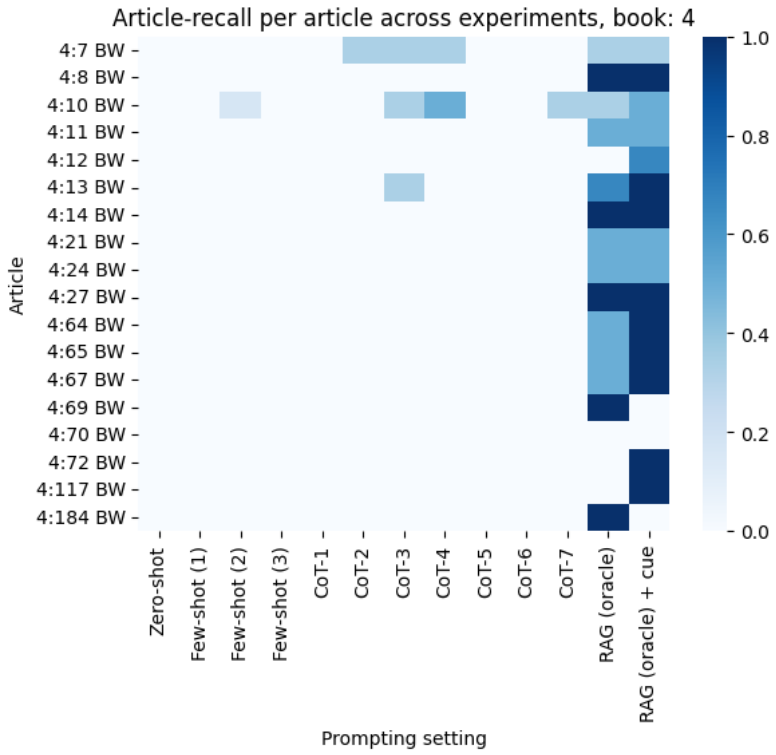


Figure 19: Article-recall per article across experiments, book 4

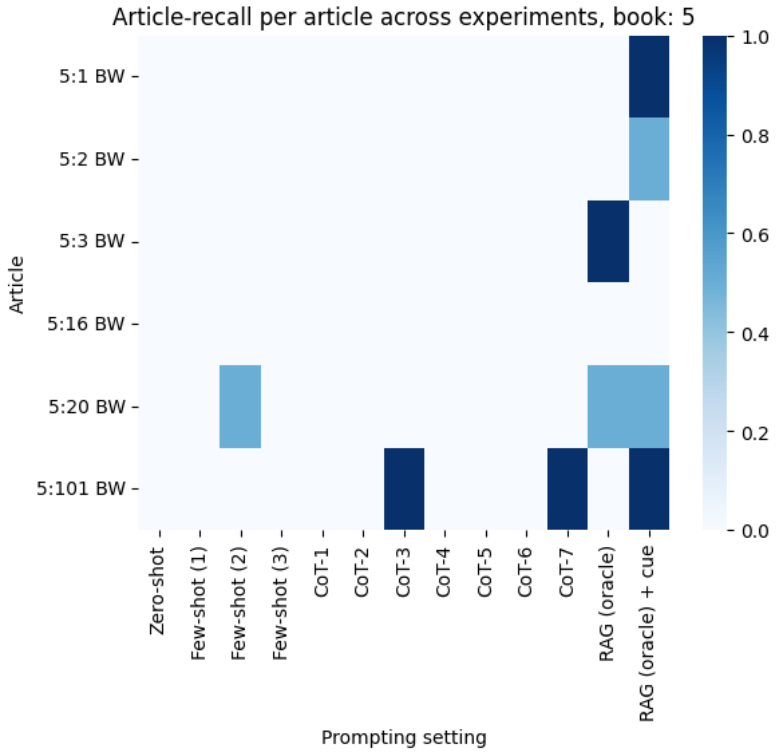


Figure 20: Article-recall per article across experiments, book 5

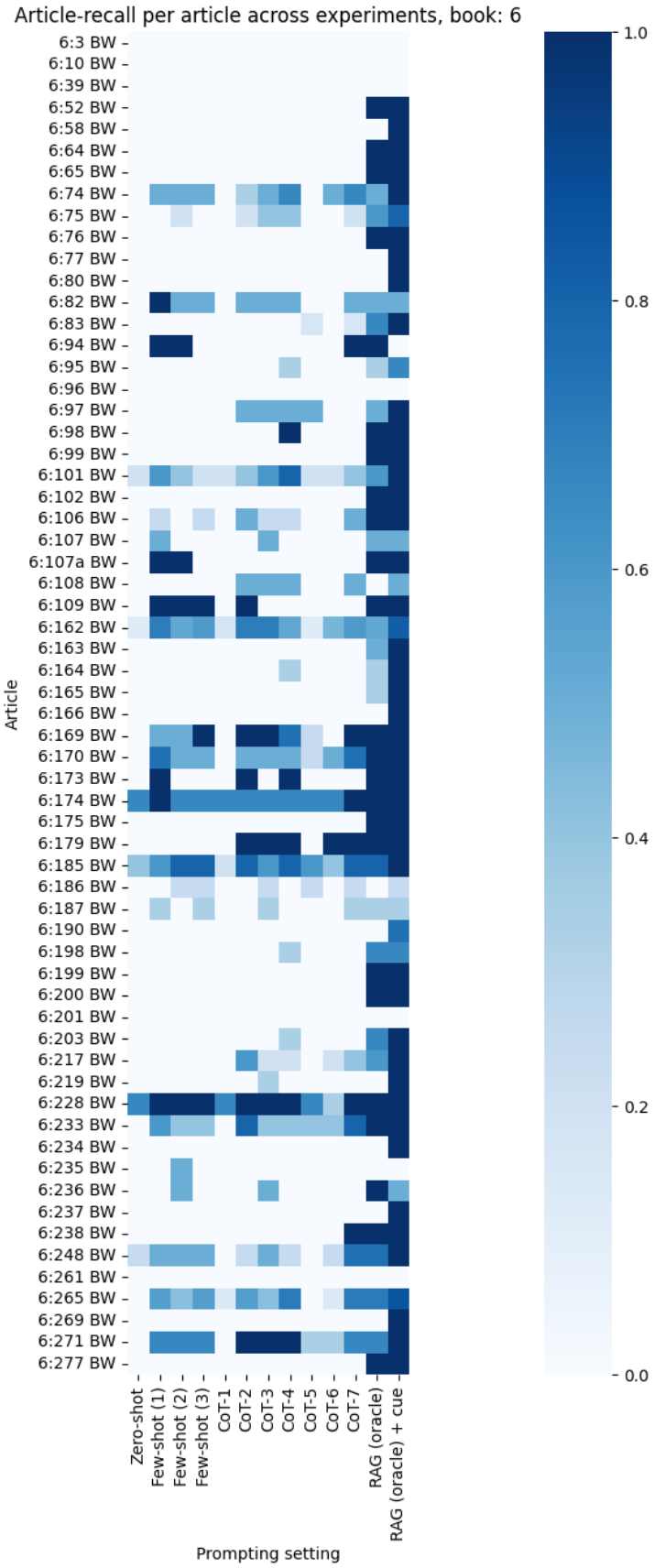


Figure 21: Article-recall per article across experiments, book 6

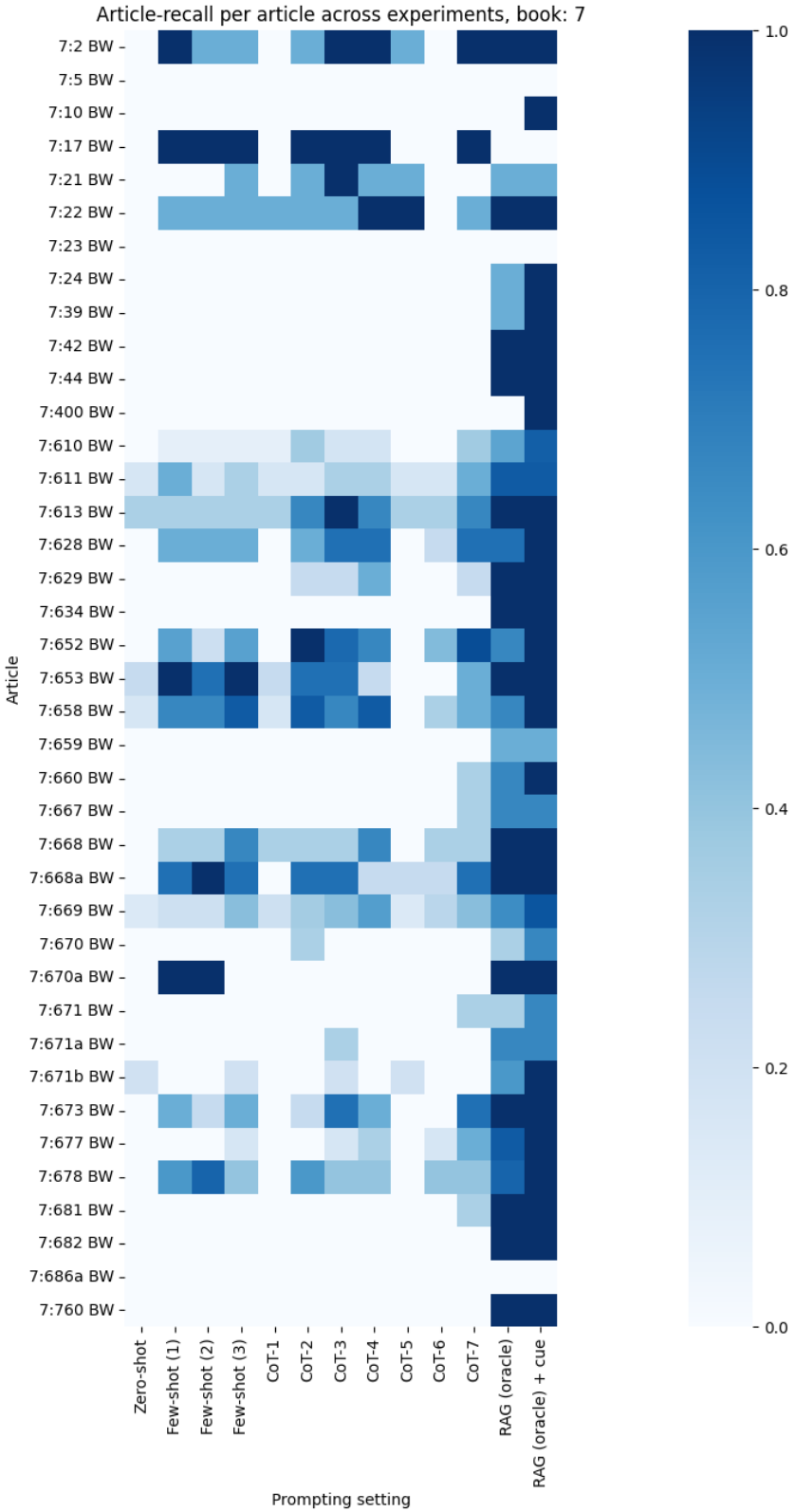


Figure 22: Article-recall per article across experiments, book 7

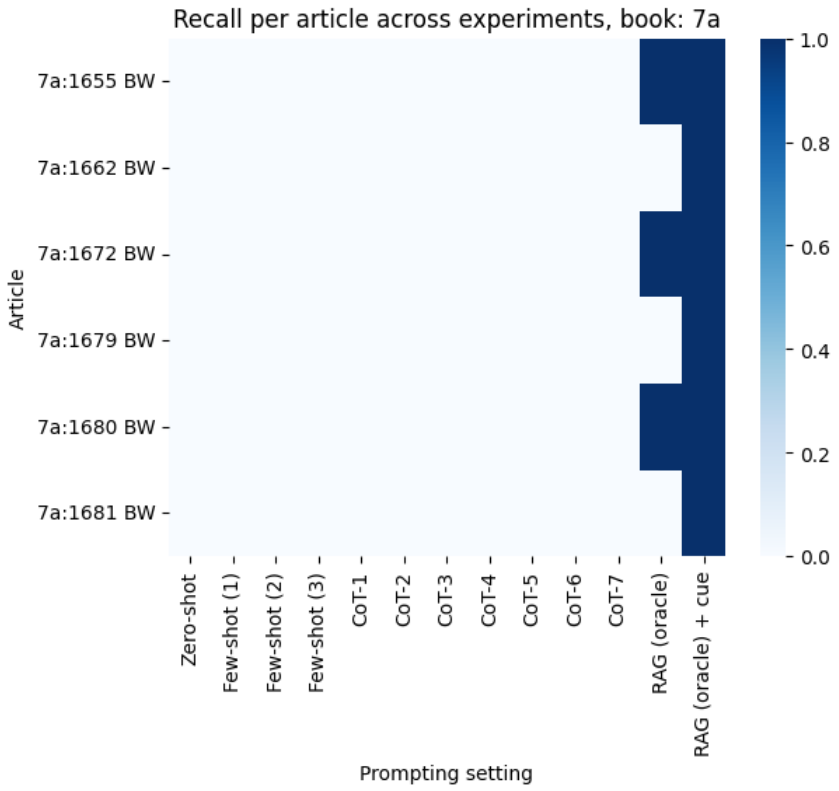


Figure 23: Article-recall per article across experiments, book 7A

## C.3 Tables

| | Valid from |
|---|---|
| Book 1 | 01-07-2023 to present |
| Book 2 | 15-11-2023 to present |
| Book 3 | 25-06-2023 to present |
| Book 4 | 01-05-2023 to present |
| Book 5 | 01-05-2023 to present |
| Book 6 | 01-07-2023 to present |
| Book 7 | 01-07-2023 to present |
| Book 7A | 01-01-2019 to present |

Table 8: Dutch Civil Code versions.

Table 9: (Average) accuracy over 223 multiple-choice questions.

| | (Average) accuracy | Majority vote accuracy |
|---|---|---|
| N = 1, temperature = 0.0 | **0.39** | - |
| N = 5, temperature = 1.0 | 0.34 | **0.39** |

# Appendix D Few-shot examples

Below, we see the three examples used in our few-shot prompting experiments (see Section 5.2.4). We prepend these examples to the prompt in order to create a 'few-shot prompt'.

**Example 1:**

> Voorbeeldvraag: Nadat hij een optreden van uTube in de Ziggo Dome heeft bijgewoond, bestelt Mario Aquilero, stervoetballer bij Newcastle United, bij Taxicentrale Vliegensvlug BV een taxi om hem van de concerthal naar Schiphol te laten vervoeren. Na enig onderhandelen spreken partijen een vaste vervoersprijs af. Even later rijdt Dirk de Wilde, in dienst van de Taxicentrale, voor om Aquilero op te halen. Nadat Aquilero is ingestapt, rijdt De Wilde met 160 km/u naar Schiphol. Onderweg raakt hij de macht over het stuur kwijt en rijdt met hoge snelheid tegen een lantaarnpaal. De Wilde komt met de schrik vrij, maar Aquilero loopt een aantal gebroken ribben en een zware hersenschudding op. Hij moet enkele weken in een ziekenhuis in Amsterdam verblijven en kan enkele maanden niet voetballen. Is De Wilde aansprakelijk voor de door Aquilero geleden schade? Geef bij uw antwoord de grondslag aan.
>
> Antwoord: Dirk de Wilde is aansprakelijk op grond van een onrechtmatige daad, art. 6:162 jo. 6:163 BW. Met het veroorzaken van het ongeluk door het overschrijden van de maximumsnelheid, waarmee hij een veiligheidsnorm heeft overschreden, handelt hij jegens Aquilero onrechtmatig in de zin van 'strijd met een wettelijke plicht' (art. 6:162 lid 1 jo lid 2 BW). De onrechtmatige daad is aan De Wilde toe te rekenen op grond van schuld in de zin van verwijtbaarheid (art. 6:162 lid 1 jo lid 3 BW). Aquilero lijdt letselschade (met daaruit voortvloeiende kosten) die hij niet zou hebben geleden indien de onrechtmatige daad niet had plaatsgevonden (causaal verband). De geschonden norm beschermt tegen de schade zoals door Aquilero geleden, aangezien het een veiligheidsnorm betreft strekkende tot bescherming van alle aan het verkeer deelnemende personen (art. 6:163 BW).

**Example 2:**

> Voorbeeldvraag: Piet, recreatief fietser, huurt in juli een racefiets bij Rent-a-bike BV. Nog voor het afrekenen van de borg van EUR 150,- en de huurprijs van EUR 20,- in de winkel heeft een kassamedewerker van Rent a-bike BV Piet gewezen op het feit dat sinds kort de "Voorwaarden Rent-a-bike BV" op alle transacties van toepassing zijn. Op de balie staat een duidelijk zichtbaar bakje met losse folders waarin de volgende tekst is opgenomen: "Voorwaarden Rent-a-bike BV (. . . ) Op al onze overeenkomsten zijn deze voorwaarden van toepassing. Indien u de borg en huurprijs heeft betaald bent u als huurder gebonden aan de huurovereenkomst. Met de betaling van de borg en de huurprijs verklaart u tevens dat Rent-a-bike BV aan al haar verplichtingen onder de huurovereenkomst heeft voldaan. Bij gebreken aan de fiets die kenbaar worden na het verlaten van de winkel is Rent-a-bike BV altijd gerechtigd de totale borg in te houden als vergoeding van de schade. Na totstandkoming van de overeenkomst en verlaten van de winkel wordt namelijk als vaststaand aangenomen dat de gebreken zijn veroorzaakt door de huurder. (. . . )" Piet kiest ervoor om geen exemplaar van de

voorwaarden mee te nemen. "Die raak ik toch kwijt", denkt hij. Piet is de straat nog niet uit gefietst of de schroef in de voorvork van de fiets breekt af. Piets vertrouwen in Rent-a-bike BV is volledig weg: hij wil zo snel mogelijk zijn borg terug. De kassamedewerker vertelt hem dat hij zijn borg niet terug krijgt nu blijkens de voorwaarden er vanuit mag worden gegaan dat Piet het gebrek zelf heeft veroorzaakt. Piet vraagt zich af of hij, gezien het beding, inderdaad geen recht meer heeft op teruggave van zijn borg. Maken de algemene voorwaarden onderdeel uit van de overeenkomst tussen Piet en Rent-a-bike en zo ja, op welke grond(en) zou Piet het beding in de algemene voorwaarden kunnen aantasten?

Antwoord: De algemene voorwaarden – in de zin van art. 6:231 sub a BW – , waaronder het betreffende beding, zijn onderdeel van de overeenkomst tussen Piet en Rent-a-bike BV. Op grond van art. 6:217 BW is deze overeenkomst tot stand gekomen door aanbod en aanvaarding daarvan. Een wederpartij (Piet) is ook dan aan de algemene voorwaarden gebonden als bij het sluiten van de overeenkomst de gebruiker (Rent-a-bike) begreep of moest begrijpen dat zij de inhoud daarvan niet kende, art. 6:232 BW. Hoewel Piet heeft geweigerd de algemene voorwaarden te lezen, is hij wel op de hoogte geweest van het bestaan ervan en heeft hij deze door het sluiten van de overeenkomst geaccepteerd. De algemene voorwaarden maken dus deel uit van de overeenkomst. Het betreffende beding kan evenwel door Piet – een consument – worden aangetast. Piet zou het beding kunnen vernietigen op grond van art. 6:233 sub a BW, nu de inhoud ervan onredelijk bezwarend is / wordt vermoed onredelijk bezwarend te zijn. Het betreft namelijk een beding in de zin van art. 6:236 lid 1 sub k BW (zwarte lijst) / art. 6:237 sub b BW (grijze lijst), omdat het Piets bevoegdheid om bewijs te leveren van wanprestatie door Rent-a-bike BV uitsluit of beperkt. Nu het beding op de zwarte/grijze lijst staat is het onredelijk bezwarend / wordt het vermoed onredelijk bezwarend te zijn, en daarmee is het vernietigbaar door Piet. (Hij kan dus verklaren dat hij het beding wil vernietigen ex art. 3:50 lid 1 BW). NB: In casu zijn de algemene voorwaarden op een juiste wijze ter hand gesteld (op grond van art. 6:230b jo. 6:230c lid 2 jo. 6:234 lid 1 BW), zodat aantasting op grond van art. 6:233 sub b niet mogelijk is.

**Example 3:**

Voorbeeldvraag: Annemiek (28 jaar) heeft haar zojuist aangeschafte, gloednieuwe racefiets in haar voortuin gezet en rijdt vervolgens met haar auto naar de supermarkt. Enkele seconden later begint het zwaar te onweren. Haar buurman Michael ziet de pannen van het dak van Annemiek waaien, rakelings langs de fiets van Annemiek. Hij belt een aantal keer bij Annemiek aan, maar zij geeft geen thuis. Michael besluit daarom om de fiets snel in zijn eigen garage te zetten. Hij zet de fiets tegen een hoge stapel met boeken gevulde verhuisdozen en loopt vervolgens naar zijn woonkamer. De stapel dozen is echter niet stabiel en net wanneer Michael de huiskamer in loopt, vallen de bovenste dozen op de fiets van Annemiek. Het voorwiel van de fiets is hierdoor volledig kromgebogen en Annemiek moet de fiets voor een fiks bedrag laten repareren. Annemiek wil haar schade op Michael verhalen. Stel dat Michael de schade aan de fiets van Annemiek heeft vergoed. Annemiek kan met haar gerepareerde fiets eindelijk de weg op. Na een paar kilometer op de provinciale weg komt ze een groep

wielrenners tegen, die ze links inhaalt. Ze blijft vervolgens aan de linkerkant van de weg rijden. In de eerstvolgende bocht wordt zij geschept door de auto van André en loopt behoorlijke letselschade op ten bedrage van e 5.000,-. André was onderweg naar de crèche om zijn dochtertje op te halen, maar was te laat. Vaststaat dat hij te hard reed en aansprakelijk is voor door Annemiek geleden schade op grond van art. 185 van de Wegenverkeerswet (WVW). Ook staat vast dat hij voor 30% heeft bijgedragen aan de schade van Annemiek. André vindt het allemaal erg vervelend voor Annemiek, maar stelt dat hij maar 30% van haar schade hoeft te vergoeden. 'Zij fietste immers links!', zo meent hij. Heeft André gelijk? Zo ja, waarom? Zo nee, waarom niet?

Antwoord: De aansprakelijkheid van André op grond van artikel 185 WVW staat vast. De vraag rijst in welke mate André de schade van Annemiek moet vergoeden. In beginsel is André ertoe gehouden de schade van Annemiek integraal te vergoeden. In casu hebben zowel Annemiek als André bijgedragen aan de schade van Annemiek. Zodoende is er sprake van eigen schuld in de zin van artikel 6:101 BW aan de zijde van Annemiek, en kan André's vergoedingsplicht in evenredigheid worden verminderd. Op grond van de wederzijdse causaliteit, de primaire maatstaf, zou André daarom in dat geval maar 30% van de schade moeten vergoeden. Deze verdeling kan echter worden gecorrigeerd op grond van billijkheidsoverwegingen, de secundaire maatstaf. In het raam van de verkeersaansprakelijkheid van artikel 185 WVW geldt op grond van vaste jurisprudentie de zogenaamde 50%-regel, op grond waarvan André, als gemotoriseerde dader, de schade van Annemiek, als ongemotoriseerd slachtoffer ouder dan veertien jaar, voor ten minste 50% moet vergoeden, behalve als er bij Annemiek sprake is van opzet of aan opzet grenzende roekeloosheid. Laatstgenoemde uitzondering is in casu niet van toepassing. Verder biedt de casus geen aanknopingspunten voor een extra billijkheidscorrectie. Daarom heeft André geen gelijk: hij moet (minstens) 50% van de schade van Annemiek vergoeden.

# Appendix E OPRO meta-prompt

In this section, we list the resulting meta-prompt after 20 optimization iterations, as well as the test set performance of the best-found OPRO instruction. All configuration settings can be found in Section 5.2.7. The 'score' in the meta-prompt refers to the training set *accuracy* (in percentages) of the instructions. The training set consists of 22 questions.

**meta-prompt**:

> Je taak is om een instructie <INS> te genereren. Hieronder staan eerder geteste instructies met hun score. The score varieert van 0 tot 100.
>
> text: Identificeer de specifieke juridische kwestie, analyseer zorgvuldig de feiten en wetgeving, bespreek de argumenten en tegenargumenten, en kom tot een weloverwogen conclusie met concrete aanbevelingen voor verdere stappen.
> score: 59
>
> text: Identificeer allereerst de specifieke juridische kwestie die zich voordoet, analyseer zorgvuldig de feiten en relevante wet- en regelgeving, beoordeel de argumenten en tegenargumenten, en kom tot een weloverwogen conclusie met concrete en specifieke aanbevelingen voor verdere stappen.
> score: 59
>
> text: Identificeer eerst de specifieke juridische kwestie, analyseer zorgvuldig de feiten en relevante wet- en regelgeving, evalueer de mogelijke argumenten en jurisprudentie, en kom tot een weloverwogen conclusie met heldere en specifieke aanbevelingen voor verdere stappen.
> score: 59
>
> text: Identificeer eerst de specifieke juridische kwestie die zich voordoet, analyseer zorgvuldig de feiten en relevante wet- en regelgeving, evalueer de mogelijke argumenten en jurisprudentie, en kom tot een weloverwogen conclusie met heldere aanbevelingen voor verdere stappen.
> score: 59
>
> text: Identificeer eerst de specifieke juridische kwestie, analyseer zorgvuldig de feiten en relevante wet- en regelgeving, evalueer de argumenten en jurisprudentie, en kom tot een weloverwogen conclusie met concrete en specifieke aanbevelingen voor verdere stappen, inclusief mogelijke alternatieve oplossingen.
> score: 59
>
> text: Identificeer de specifieke juridische kwestie die zich voordoet, analyseer zorgvuldig de relevante feiten en wet- en regelgeving, evalueer de mogelijke argumenten en jurisprudentie, en kom tot een weloverwogen conclusie met concrete, specifieke en haalbare aanbevelingen voor verdere stappen en eventuele alternatieve oplossingen.
> score: 59
>
> text: Identificeer eerst de specifieke juridische kwestie die zich voordoet, analyseer zorgvuldig de feiten en relevante wet- en regelgeving, en kom tot een weloverwogen conclusie met duidelijke aanbevelingen voor verdere stappen.
> score: 64

text: Identificeer eerst de specifieke juridische kwestie, analyseer zorgvuldig de relevante feiten en wet- en regelgeving, evalueer de mogelijke argumenten en jurisprudentie, en kom tot een weloverwogen conclusie met concrete en specifieke aanbevelingen voor verdere stappen en eventuele alternatieve oplossingen.
score: 64

text: Identificeer eerst de specifieke juridische kwestie, analyseer zorgvuldig de relevante feiten en wet- en regelgeving, beoordeel de mogelijke argumenten en jurisprudentie, en kom tot een weloverwogen conclusie met concrete en specifieke aanbevelingen voor verdere stappen.
score: 68

text: Identificeer eerst de specifieke juridische kwestie, analyseer zorgvuldig de feiten en relevante wet- en regelgeving, evalueer de mogelijke argumenten en jurisprudentie, en kom tot een weloverwogen conclusie met concrete en specifieke aanbevelingen voor verdere stappen.
score: **73**

Genereer een instructie die anders is dan de instructies <INS> hierboven, en die een hogere score haalt dan de instructies <INS> hierboven. De instructie moet beginnen met <INS> en eindigen met </INS>. De instructie moet beknopt, effectief, en over het algemeen toepasbaar zijn op alle bovenstaande problemen. De instructie moet zich ook baseren op de IRAC methode.

**Evaluation on test set (201 questions)**

| Instruction | Accuracy |
|---|---|
| <no instruction> | 0.34 |
| "Identificeer eerst de specifieke juridische kwestie, analyseer zorgvuldig de feiten en relevante wet- en regelgeving, evalueer de mogelijke argumenten en jurisprudentie, en kom tot een weloverwogen conclusie met concrete en specifieke aanbevelingen voor verdere stappen" | **0.43** |

Table 10: Accuracy of the empty instruction vs. the best-found OPRO instruction.