Algorithm for Automatic Legislative Text Consolidation

Matias Etcheverry
Doctrine
Ecole Nationale Supérieure Paris-Saclay
matias.etcheverry@doctrine.fr

Thibaud Real
Doctrine
thibaud.real-del-sarte@doctrine.fr

Pauline Chavallard
Doctrine
pauline@doctrine.fr

Abstract

This study introduces a method for automating the consolidation process in a legal context, a time-consuming task traditionally performed by legal professionals. We present a generative approach that processes legislative texts to automatically apply amendments. Our method employs light quantized generative model, fine-tuned with LoRA, to generate accurate and reliable amended texts. To the authors knowledge, this is the first time generative models are used on legislative text consolidation. Our dataset is publicly available on HuggingFace[1]. Experimental results demonstrate a significant improvement in efficiency, offering faster updates to legal documents. A full automated pipeline of legislative text consolidation can be done in a few hours, with a success rate of more than 63% on a difficult bill.

1 Introduction

Every year in France, the *Projet de Loi Finance*[2] (PLF), annually introduces numerous modifications to the General Tax Code (484 in 2024). The objective of this study is to automate the process of legislative text consolidation, which is the act of combining modifications from a `modification section`, contained inside the PLF, to an `existing article` to generate a `modified article`. Example 1 illustrates a dummy consolidation, where the original text of a law is updated by incorporating amendments directly into it, resulting in a revised, coherent version.

Legislative text consolidation is a critical yet time-consuming task, traditionally performed manually by legal professionals. A sample of the PLF is presented in Example[3] 2. It modifies `article 1586 ter` and `article 1586 quater` of the General Tax Code.

[1]Link to dataset
[2]Link to *Projet de Loi Finance* for 2024
[3]All examples are translated from French to English.

Example 1: Illustration of legislative consolidation

`Existing article:` `Paris is the capital of France.`
`Modification section:`
`I.- Replace the word « is » with « has been ».`
`II.- Add « since the late 10th century » at the end of the sentence.`
`Modified article:` `Paris has been the capital of France since the late 10th century.`

The conventions represented on Figure 1 are adhered to:

- A legislative bill is composed of multiple articles[4].
- An article comprises several sections. A section is defined as a collection of paragraphs that enact modifications to a single article.
- A section may effectuate either a singular modification or multiple modifications. For instance, section `A.` – implements a single modification, whereas section `B.` – introduces four modifications.

Three primary modification categories are identified: a deletion, involving the removal of a word, sentence, or paragraph; an addition, encompassing the insertion of a word, sentence, or paragraph and a substitution, where a word, sentence, or paragraph is exchanged for another. Example 2 demonstrates one instance of addition and four instances of substitutions.

The automation of legislative text consolidation has the potential to significantly expedite

[4]An article would be the equivalent of a section in a bill in common-law countries.

Proceedings of the Natural Legal Language Processing Workshop 2024, pages 166–175

Example 2: Extract of article 79 of the PLF 2024

Modification sections:

I.-The General Tax Code is amended as follows:

A.-The following words are added to the first sentence of the second paragraph of 1 of II of Article 1586 ter: « , as it stood prior to Finance Act 2023-1322 of 29 December 2023 for 2024 »;

B.-Article 1586 quater is amended as follows:

1° I is amended as follows

a) The second paragraph of b and c is amended as follows:

-at the beginning, the rate: « 0.125% » is replaced by the rate: « 0.094% »;

-at the beginning, the rate: « 0.094% » is replaced by the rate: « 0.063% »;

b) The second paragraph of c is amended as follows:

-the rate: « 0.225% » is replaced by the rate: « 0.169% »;

-the rate: « 0.113% » is replaced by the rate: « 0.056% »;

this process, offering a rapid update of legal documents post-enactment and potentially pre-enactment, thereby enhancing the accessibility and reliability of legal information.

2 Related works

2.1 Information extraction approaches

Modification sections typically follow a consistent lexical structure. Arnold-Moore (1995, 1997) and Mazzei et al. (2009) exploit this formal consistency to extract amendments and construct a structured tree representation, applying information extraction techniques.

Subsequently, a clear trend is drawn in information extraction between tagging-based methods and generative methods. Tagging-based methods are designed to classify individual tokens (token-based methods) or clusters of tokens (span-based methods). In contrast, generative methods are oriented

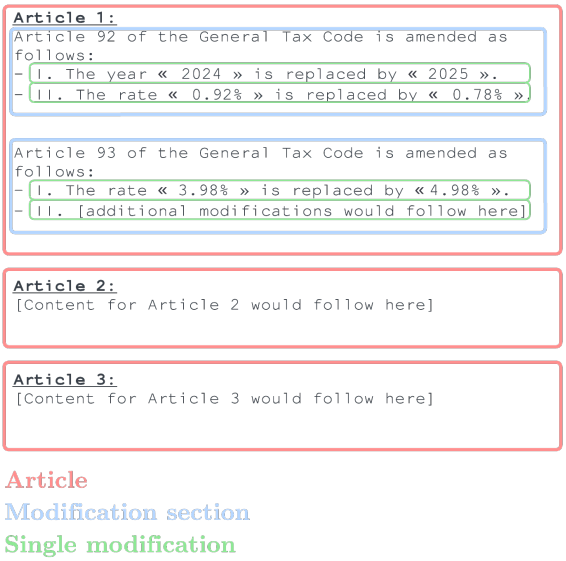

Figure 1: General structure of the PLF

towards producing textual content that is inherently construed as a relationship triplet. Hence, Shi and Lin (2019) undertake a notably question answering adaptation of the BERT model to facilitate generation across a diverse corpus, achieving a remarkably good baseline. In more recent times, generative models appear to exhibit superior performance. Josifoski et al. (2022) introduce the GenIE model, which succeeds in generating generation triplets through its utilization of the BART architecture.

In recent developments, models dedicated to text editing have garnered interest for their utility in tasks that necessitate the rearrangement of words and text spans, such as summarization. Malmi et al. (2019) introduced `LaserTagger`, an approach that assigns one of several tags to tokens, including `KEEP`, `DELETE`, `SWAP`, or `PRONOMINALIZE`, to facilitate text editing. Concurrently, Mallinson et al. (2020) developed a two-stage algorithm wherein the first model tags tokens, and the subsequent model is responsible for the rearrangement of these tagged tokens.

2.2 Generative approaches

Generative approaches rapidly took the lead to reinterpret any task of extraction, classification, or edition as generative problems under certain frameworks (Raffel et al., 2019). Building upon this foundational work, Chung et al. (2022) expanded the utility of these models through the fine-tuning process to accommodate a broad spectrum of human instructions, thereby enhancing their applicability. This advancement has catalyzed subsequent

research endeavors, focusing extensively on the exploration of instruction-based fine-tuning within the realm of generative models.

Instruction tuning It is crucial to recognize that fine-tuning the model for a specific task is pivotal (Brown et al., 2020; Wei et al., 2022). In specific-use Pretrained Large Language Models (PLLMs), such as for legislative text consolidation, we may use instruction tuning to ensure that our model consolidates the provided legal text in all cases.

Finetuning Existing parameter-efficient tuning methods still lag behind full fine-tuning on higher-resource and challenging tasks, but often succeed when dealing simple tasks, as consolidation would be (He et al., 2022). These approaches enable instruction tuning to be performed on cost-effective GPUs.

On one hand, prompt tuning methods involve concatenating the embeddings of input tokens with a trainable tensor. This tensor can be optimized through backpropagation to enhance the modeling performance for a specific task. Remarkably, prompt tuning achieves modeling performance comparable to fine-tuning all layers, yet only necessitates training 0.1% of the parameters (Li and Liang, 2021; Liu et al., 2021). On the other hand, adaptation methods involve the insertion of fully connected layers into the transformer blocks (Houlsby et al., 2019). These techniques achieve equivalent performance to prompt tuning, albeit slightly more parameter-intensive. He et al. (2022) finds an equivalence between prompt tuning and adapter methods: adapter tuning is prompt tuning in series.

Ultimately, the LoRA method has garnered significant popularity (Hu et al., 2021). This technique involves adding a low-rank matrix to certain matrices within the PLLM. The underlying notion is that low-rank matrices encapsulate all the required information for precise task fine-tuning, while PLLM matrices encompass the full spectrum of information from pretraining. Notably, this method is not restricted solely to instruction embedding; it is applicable to a broad array of fine-tuning tasks. Furthermore, when utilized in conjunction with model quantization methods, LoRA extends the capability of fine-tuning numerous PLLMs (Dettmers et al., 2023).

3 Dataset

Our first objective is to construct a dataset for automatic consolidation. Each sample in this dataset is a triplet of texts `(existing article, modification section, modified article)` in which the `modification section` specifies the changes to be made to the `existing article` to obtain the `modified article`. Our research shall primarily concentrate on the national consolidation. On a national scope, laws, decrees, and regulations revise existing legal regulations.

The publication of legal texts has seen significant growth over the past 20 years. Figure 2 shows the evolution of the number of modifying articles recorded in France. In 2022, 17487 texts were published from 2512 laws providing modifications on existing laws. We create a dataset of 5000 triplets `(existing article, modification section, modified article)`. We only keep `existing articles` that are modified only once. This condition helps avoid `existing articles` modified by two `modification sections` simultaneously.

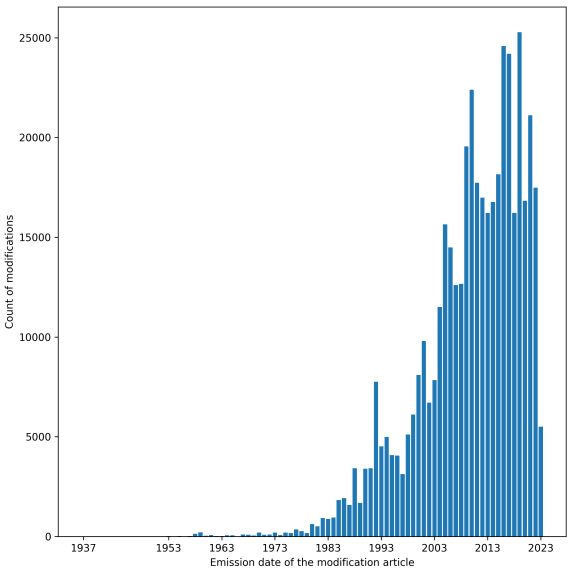

Figure 2: Number of modification sections published per year in France

The links between `(existing article, modification section, modified article)` are publicly available through the Légifrance platform. In the end, we accumulated a dataset comprising 3124 triplets. Example 3 shows a complete sample of the dataset, publicly

available on HuggingFace[5].

Example 3: Sample of the dataset

Existing article: *Article 10:*
Appointments are made each year in the last week of August.
The general meeting of the order meets at the courthouse.
Modification section: *Article 5:*
Article 10 is amended as follows:
1° The words « in the last week of August » are replaced by the words « during the month of December »;
2° The second sentence is deleted.
Modified article: *Article 10:*
Appointments are made each year during the month of December.

4 Approaches & developped methods

The objective is to modify an existing article by incorporating alterations delineated within a modification document. Initially, a foundational baseline employing a span extraction methodology was developed. Subsequently, this baseline is evaluated against our advanced methodology, which encompasses the fine-tuning of a pre-trained language model.

4.1 Baseline: span extraction through question answering

We aim to establish the baseline outlined in Section 2.1, which involves adapting a BERT model for the question answering task (Shi and Lin, 2019). Two distinct models are employed for this purpose. The first model is designed to extract spans that need to be added within the modification section, while the second model identifies spans for deletion within the existing article. Consequently, the span of words identified by the second model can be overwritten by the span generated by the first model. Both approaches utilize the same architecture, employing a CamemBERT Model with a span classification head [6]. This head consists of a linear layer on top of the hidden-state outputs to compute span start logits and span end logits. This model comprises 110M parameters. The batch size is 16 and the learning rate for the Adam optimizer is 2×10^{-5}. We train for 15 epochs.

Labeling and input format Example 4 illustrates the labeling for a sample of the dataset, where spans highlighted in red are predicted by an initial model and subsequently overwritten by spans highlighted in green, as predicted by a second model. This labeling schema facilitates the modeling of three modification types: additions are seamlessly integrated into existing text by substituting blank spaces. Moreover, we introduced the token [NL] (New Line) prior to the commencement of each paragraph and at the conclusion of each text, as it is denoted that the consolidation process often refers to paragraph. The models acquisition of this token contribute to improved performance in instances exhibiting such patterns.

Example 4: Labels in the span extraction dataset for a substitution

Existing article: [NL]the duties corresponding to the post of Chief State Public Works Engineer in the second group referred to in Article 8 of this Decree are, for the post reporting to the Minister for Foreign Affairs: [NL] Charged with the duties of Deputy Director of Real Estate Operations in the Real Estate Affairs Department within the General Administration Department. [NL]

Modification section: [NL] The second paragraph of Article 1 of the above-mentioned Order of 4 May 2007 is replaced by the following provisions: [NL] « Assistant to the Deputy Director of Real Estate Operations. » [NL]

Legend: Span to be predicted by the first model. Span to be predicted by the second model to be overwritten by the first span.

When inputting data into the model, the existing article and the modification section are concate-

[5] Link to dataset
[6] Camembert for question answering

nated with a [SEP] token in between. The modification section serves as the "Question" while the existing article acts as the "Paragraph"

Test set and metrics To assess the model's performance, we test the model on a dataset comprising 302 triplets. Once the spans are predicted, the consolidated text can be reconstructed accordingly. Therefore, it becomes pertinent to utilize a end-to-end oriented metric: word error. Commonly applied in speech-to-text algorithms, the word error measures the number of errors in the transcription of a speech. In our context, this metric assesses the error count within the predicted consolidated text relative to the expected version.

4.2 Text generation

Our aim is to leverage generative models to directly predict consolidated texts. Whereas the span extraction method can lead to linguistically nonsensical outcomes in case of prediction errors, generative models ensure the grammatical correctness of generated texts.

4.2.1 Fune-tuning & Instruction tuning

We opt to fine-tune a generative model using the LoRA approach (Dettmers et al., 2023). Given that we are solely focusing on a single task for fine-tuning, it did not seem particularly advantageous to employ a prompt tuning method, which is particularly suited for datasets containing diverse types of instructions. The LoRA technique was applied to the projection layers of the query, key and value components of the pretrained language model, targeting approximately 3% of the parameters from the original model.

The prompt format is straightforward and adheres to the conventions commonly employed in instruction tuning. Example 5 illustrates the input format during training. The `Instruction` corresponds to the modification to be performed, i.e., the modification section. The `Input` corresponds to the existing article on which the modification is to be applied. Lastly, the expected `Response` pertains to the modified article. During inference, the `Response` field is left empty, and the model is tasked with predicting it.

We are employing open-source models that are open for commercial use. Our baseline model is OpenLLama, which is a replication of LLaMa with less intrusive licenses. This model has undergone the same pretraining process as LLaMa and is available in various sizes, ranging from 3 to 13 billion parameters. For training these models, we will utilize Nvidia T4 GPUs with 16GB of memory or Nvidia A10G GPUs with 24GB of memory, depending on the model size.

Example 5: Example of prompt

```
### Instruction:
Article 10 is amended as
follows:
1° The words « in the last week
of August » are replaced by
the words « during the month of
December »;
2° The second sentence is
deleted.
### Input:
Appointments are made each year
in the last week of August.
The general meeting of the
order meets at the courthouse.
### Response:
Appointments are made each year
during the month of December.
```

Consistently across the conducted experiments, certain hyperparameters were kept uniform: the learning rate was set at 3×10^{-4}, and the LoRA dropout rate was sustained at 5%. A 4-bit quantization is employed. Only prompts containing fewer than 1024 tokens were selected for use. The micro batch size was determined to be 4, with gradient checkpointing applied after processing every 128 samples. The training duration was limited to 2 epochs.

4.2.2 Training on the modified article only

The first experiment involved comparing two models trained with the same prompt, which includes the `Instruction`, `Input`, and `Response` fields. However, the tasks differ: one model is trained to predict the entire prompt (i.e., all three fields: `Instruction`, `Input`, and `Response`), while the other model is trained solely to predict the `Response` field. In both cases, the full prompt is provided as input during training. Two opposing intuitions were considered. On one hand, training the model to predict the complete prompt could enhance its comprehension of legislative semantics. On the other hand, training the model to predict solely `Response` field removes certain constraints. For this experiment, we selected two Open-LLaMa models with

3 billion parameters each. The Table 1 below summarizes the results. Notably, training a model exclusively on the `Response` field yields superior performance, of +9.4%.

Model trained on	Average Word Error	Median Word Error
Whole prompt	18.6	10.5
Modified article	17.0	7.0

Table 1: Training on the whole prompt vs. training on the modified article only

4.2.3 Influence of cleaning the dataset

We also aimed to examine the influence of dataset quality on consolidation performance. To this end, we selected two OpenLLaMa models with 3 billion and trained them using two distinct consolidation datasets. The second dataset was a cleaned version of the open-sourced dataset, where all consolidation cases that did not involve any modification or involved tables were removed, comprising 1784 triplets.

The results of this comparison highlight the impact of dataset quality on consolidation performance, as shown in Table 2. By using a cleaner dataset that focuses exclusively on meaningful consolidation examples, the model tends to achieve better outcomes, even when compared to a larger dataset that includes less relevant instances. This underscores the significance of dataset quality in influencing model performance for the consolidation task.

Dataset	Average Word Error	Median Word Error
Full dataset	17.0	7.0
Curated dataset	12.0	4.0

Table 2: Influence of the quality of the training dataset

4.2.4 Influence of the size of the low rank matrix

The LoRA finetuning method encompasses two hyperparameters: the rank r of the added matrices and the multiplier α. The multiplier α operates as a learning rate for the added matrices and exhibits relatively modest effects once it reaches a sufficiently high value. It was set as twice the value of r. The matrix rank r significantly impacts the model's performance. A smaller r suggests limited fine-tuning, where the model requires minimal adaptation to accomplish the intended task. In contrast, a larger r implies extensive retraining, almost akin to starting from scratch. Table 3 showcases the performance of two models, each utilizing different r values. Notably, the model with the higher r value attains slightly better consolidation capabilities. It can be observed that the model with a higher rank value r trains faster but eventually converges to a similar value as the other model.

Rank r	Average Word Error	Median Word Error
16	12.0	4.0
64	11.7	4.0

Table 3: Influence of the rank of the added matrices

4.2.5 Influence of the size of the PLLM

We also examined the impact of the PLLM size on consolidation performance. To do so, we compared three OpenLLaMa models with 3 billion, 7 billion, and 13 billion parameters, respectively trained on a curated dataset with a large low-rank r. Despite being more challenging to fine-tune, larger models generally exhibit better performance due to their increased information retention capacity. We further compared these models with a 13-billion-parameter OpenLLaMa model that had already undergone an initial round of fine-tuning on an instruction dataset.

Table 4 outlines the results. It's observed that, on average, the number of errors is lower for the 3-billion-parameter model compared to the 7-billion-parameter model. However, the number of errors is higher in terms of median values. The 7-billion-parameter model generally predicts better modified articles, but in some cases, the modified article is significantly worse from the expected text. This can be attributed to the fact that the 7-billion-parameter model possesses a larger generative capacity. As a result, in complex consolidation examples, the 7-billion-parameter model might "hallucinate" and generate interpretations of the texts, whereas the 3-billion-parameter model tends to generate the unconsolidated existing article. This effect disappear in the 13-billion-parameter model which showcases a considerable performance gap. The 13-billion-parameter model, which was pre-fine-tuned on an instruction dataset, further enhances consolidation performance.

Model size	Average Word Error	Median Word Error
3b	10.0	2.0
7b	13.5	0.5
13b	6.07	0
13b pre-finetuned	5.09	0

Table 4: Influence of the size of the PLLM

4.3 Comparing all methods

We now proceed to compare the different approaches employed: span extraction and the generative method, on a more challenging dataset. While it would have been ideal to maintain the same test set as used in previous sections, it became evident that the models were insufficiently distinguishable based on that test set alone. Therefore, we opted to construct a second, more complex test set, utilizing the legal provisions from the previous year's PLF. Additionally, we compare our results with OpenAI's GPT-3.5 and GPT-4 models to further contextualize model performance.

Approach	Prompt size	Average Word Error (95% CI)
Span Extraction	512	36.2*(31.4)
Generative models		
Open-LLaMA 3b	1024	65.5 (29.2)
Open-LLaMA 13b	1024	**20.7** (7.79)
GPT3.5-turbo-0613	4k	44.8 (27.5)
GPT4-0613	8k	**9.41** (3.58)

* Computed only on single modifications

Table 5: Comparison of the proposed approach with the baseline

The results are denoted in Table 5. We first observe that the generative models yield the best performances. While these models generally produce highly accurate consolidations, in certain cases, the consolidation can result in aberrations, leading to hallucinations and the generation of lengthy texts, resulting in substantial consolidation errors. The distributions of word errors for each model and error type (addition, deletion, substitution) are depicted in Figure 3. Additionally, it is notable that GPT4 demonstrates superior performance, while our best model (Open-LLaMa 13b) falls between GPT3.5 and GPT4.

5 Application on a bill

Encouraged by our model's promising performance, we embarked on live automatic consolidation of the *Projet de Loi de Finance 2024* from September 2023 to December 2023. This bill was proposed on 26th of September, 2023 and contained 60 articles. After multiples debates at the parliements, the bill was promulgated on 29th of December, 2023 with 264 articles. This is a highly complex bill.

5.1 Pipeline

Our consolidation pipeline is depicted in Figure 4. This pipeline was up during the four months of life of the bill. The pre-processing consists of three primary steps.

5.1.1 Section splitter

A bill is structured into articles, each specifying modifications to current laws on particular topics. Consequently, a bill's article might introduce several changes to numerous laws. A section division using regular expressions is therefore created to break down the bill's article into these distinct sections. This splitter leverages the hierarchical structure of the bill's article to efficiently segment it into components.

5.1.2 Entity recognition

We employ an already fine-tuned entity recognition model system to identify the specific law articles targeted by each section. Upon identifying these articles, we retrieve their contents for further processing.

5.1.3 Our consolidation algorithm

We use our best model to generate the consolidated text.

5.2 Results

In this section, we delineate the consolidation process undertaken as of 16th of December, 2023. At this juncture, the legislative bill comprised 271 articles. Upon division, this legislative text was found to encompass 1399 simple modifications applicable to 606 articles of law.

Our pipeline incorporates two instances of human intervention, symbolized by hand icons, primarily focused on verification rather than labeling. The law article detection phase, leveraging an existing entity recognition component, achieved an 82.0% success rate. To quantify the success rate

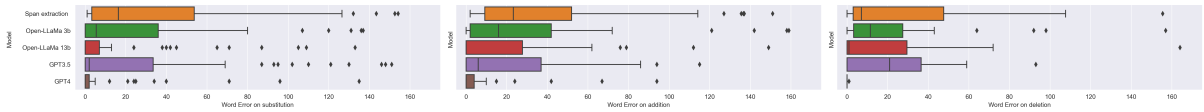

Figure 3: Word error distributions per model per modification type

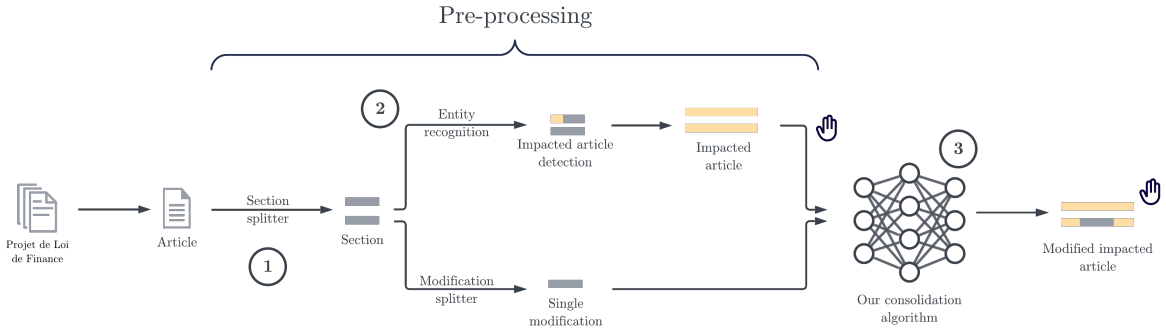

Figure 4: Full consolidation pipeline

of our algorithm, we executed the consolidation process on the legislative bill using both GPT-4 and our best model, OpenLLaMa-13, generating two sets of predictions. Subsequently, we scrutinized and amended the predictions made by GPT-4 to produce a third set, representing human annotations. For a prediction and an annotation, we removed special characters from each string, such as accents, commas, and line breaks, to facilitate the comparison of the raw texts. However, it exists two cases where the consolidation process can't be done: the presence of tables and lengthy prompts. Table 6 presents the rate of possible consolidations along the rate of correct consolidation for both algorithms.

Model	**Rate of possible consolidations**	**Correctness rate among possible consolidations**
Our model	49.8%	63.2%
GPT4-0613	91.3%	61.4%

Table 6: Correct consolidation rate

Our Open-LLaMa-13b model faces challenges due to its limited context size, allowing application in only 49.8% of consolidation cases. Conversely, GPT4-0613 encounters difficulties in consolidating only 8.7% of cases, all related to the inclusion of tables. In terms of correctness rates, both models achieve 63.2% and 61.4% respectively, considering their respective possible consolidations. While our algorithm appears to achieve a higher correctness rate, it's crucial to note that it consolidates far fewer samples with much smaller prompt sizes compared to GPT4, which consolidates most of them.

In Figure 5, we depict the correctness rate against the full prompt length, including the generated `Response`, for both models in cases possible for our Open-LLaMa-13b. Here, GPT4-0613 achieves a 73.6% correctness rate. Notably, the full prompt length for the GPT-4 model slightly differs due to the inclusion of few-shot examples. Both models exhibit differing behaviors in correctness rates against full prompt length. Open-LLaMa-13b peaks for full prompt lengths below 1000 tokens, with performance gradually decreasing for larger prompts, highlighting attention mechanism limitations. Conversely, GPT4-0613 demonstrates consistent performance across varying prompt lengths, showing no impact from larger prompts.

6 Conclusion

This research implements a generative method to automate legislative text consolidation, demonstrating a significant capability to process and automatically apply changes to legislative texts. We determined that the quality of the dataset and the size of the pre-trained model were two parameters that most significantly influenced consolidation performance. Despite exceptional performances of GPT4, in the end, we ideally prefer to use an open-source model for handling legal data due to its sensitivity. The consolidation, led on a real-time legislative bill, proved to be highly effective, although occasional issues in the generation process could result

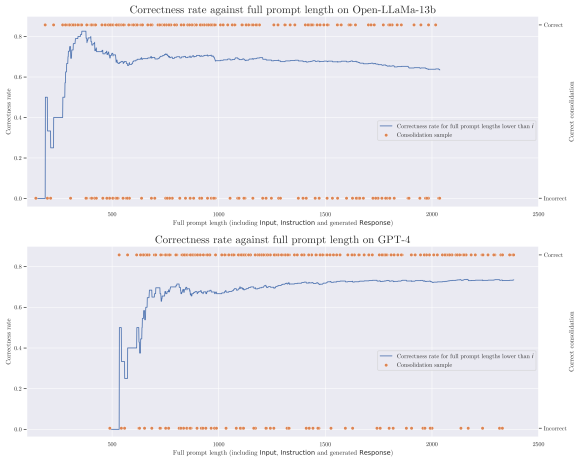

Figure 5: The correctness rates against prompt length are plotted for Open-LLaMa-13b and GPT-4 on the same consolidation samples (49.8% of the PLF). Each dot represents a sample of the PLF consolidation, indicating whether it is correct or not. The curve at prompt length i illustrates the rate of correct consolidation among samples with a prompt length less than i.

in nonsensical consolidations.

Moving forward, our objective is to delve into advanced fine-tuning strategies and broaden our methodology to encompass additional models. On one side, there exists a variety of models equipped with commercial licenses, such as LLaMA 3.1, that offer new possibilities for exploration. These models often feature larger context windows, enabling the consolidation of more samples. On the other side, innovative fine-tuning techniques are being developed, such as the Mixture of LoRA Experts approach. This technique is designed to fine-tune each expert within a Mixture of Experts.

This research opens promising avenues for integrating generative methods into legal processes, with the hope of radically transforming legal practice.

References

Timothy Arnold-Moore. 1995. Automatically processing amendments to legislation. In *Proceedings of the 5th International Conference on Artificial Intelligence and Law*, ICAIL '95, page 297–306, New York, NY, USA. Association for Computing Machinery.

Timothy Arnold-Moore. 1997. Automatic generation of amendment legislation.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. *Preprint*, arXiv:2110.04366.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *Preprint*, arXiv:1902.00751.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Martin Josifoski, Nicola De Cao, Maxime Peyrard, Fabio Petroni, and Robert West. 2022. Genie: Generative information extraction. *Preprint*, arXiv:2112.08340.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *Preprint*, arXiv:2101.00190.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *Preprint*, arXiv:2103.10385.

Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. 2020. Felix: Flexible text editing through tagging and insertion. *Preprint*, arXiv:2003.10687.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. *Preprint*, arXiv:1909.01187.

Alessandro Mazzei, Daniele P. Radicioni, and Raffaella Brighi. 2009. Nlp-based extraction of modificatory provisions semantics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *Preprint*, arXiv:1904.05255.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.