

Applying Large Language Models for Enhancing Contract Drafting

Kwok-Yan Lam¹, Victor C.W. Cheng² and Zee Kin Yeong³

¹ Nanyang Technological University, Singapore

² TAU Express Pte Ltd, Singapore

³ Singapore Academy of Law, Singapore

Abstract

This paper investigates the use of traditional AI and generative AI techniques in enhancing the work of legal professionals. We propose an approach that applies a combination of AI techniques, traditional AI augmented with generative models for automating some of the laborious tasks in contract drafting. With the launches of advanced AI models such as ChatGPT, legal professionals are anticipating how such technologies can streamline their works. We first introduce how these models generate text contents given a user prompt. Then we propose some practical approaches in “prompt writing” which enable better and more coherent contract clauses to be generated. As privacy is typically a great concern in using ChatGPT in professional domains, we also explore the feasibility and effectiveness of using on-premises Large Language Models (LLMs) such as “Vicuna” as practical alternatives that may address the privacy issues while producing acceptable performance in contract drafting. Since AI generated clauses may not match the strict legal requirements or even be incorrect, we propose an approach to evaluate the clauses with traditional AI by using sentence transformers to retrieve similar clauses from a trusted source and perform automatic content similarity analysis. Experimental results using the public dataset LEDGAR showed that LLMs are useful tools for contract clause drafting and the automated comparison results can work as hints or recommendations that users can consider to revise and enhance the generated clauses, hence simplifying the task of contract drafting by legal professionals in an augmented intelligence manner.

Keywords

LLMs, ChatGPT, clause recommender, contract drafting, hallucination, AI safety

1. Introduction

Contracts are legally binding between parties and sets out the agreement between them of their respective rights and obligations. They are usually written in text format and contain clauses setting out specific terms and conditions that capture their agreement. The textual nature of contracts presents a great potential for using natural language processing (NLP) to contribute and assist the drafting process. But unlike other NLP tasks, words and phrases in contracts must be carefully used to precisely, unambiguously, and

accurately capture the agreement between the parties and this pose a great challenge to researchers.

In general, contract drafting typically involves a two-step process [1], as follows:

- Selecting a contract template which contains clauses that match with the situation or context of the specific agreement being entered into; and
- Modifying the clauses to capture the specific context and details agreed between the parties.

These tasks require manual effort and domain knowledge, especially the second task which needs particularized information, accurate understanding and sophisticated thought-process. With the recent launches of ChatGPT [2] and some LLMs such as LLaMA [3], contracts can be

In: *Proceedings of the Third International Workshop on Artificial Intelligence and Intelligent Assistance for Legal Professionals in the Digital Workspace (LegalAIIA 2023)*, held in conjunction with ICAIL 2023, June 19, 2023, Braga, Portugal.

✉ kwokyan.lam@ntu.edu.sg (K.Y. Lam),

victor.cheng@tauexpress.com (V.C.W. Cheng).

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).



CEUR Workshop Proceedings (CEUR-WS.org)

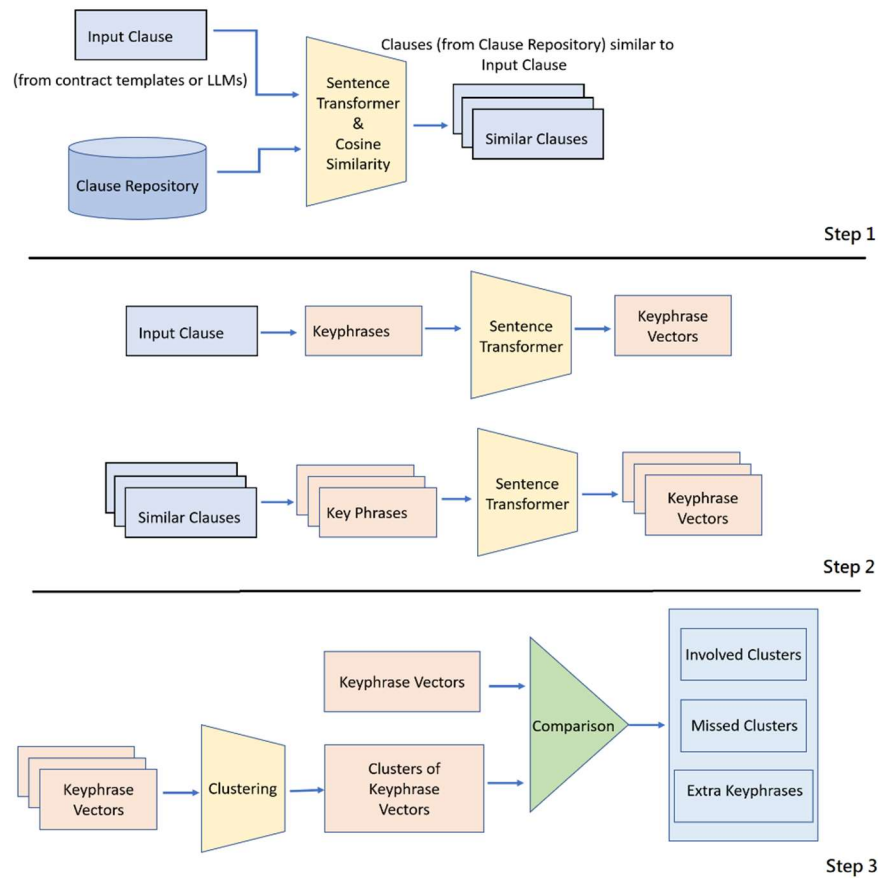


Figure 1: Analysis of an input clause by comparing with similar clauses from a trusted repository. Step 1: Use a sentence transformer to convert the input clauses to vector representation and extract similar clauses from the clause repository by means of cosine similarity. Step 2: Extract the keyphrases for both the input clause and the extracted clauses. Then convert them to vectors by using the sentence transformer. Step 3: Perform clustering on keyphrase vectors coming from the extracted clauses. Then compare the input clause vectors with the clusters and obtain “Involved Clusters” (clusters having the input clause keyphrases around), “Missed Clusters (No input clause keyphrase is found around), and ”Extra Keyphrases” input clause keyphrases that do not fall in any clusters).

generated automatically, given a user description (known as prompt) which consists of the contract context and the requirements. This paper focuses on: (a) generation of clauses with LLMs, and (b) identifying the aforementioned aspects in the clauses. Task (b) is very significant because the clauses obtained from (a), or contract templates, may not be usable, or cannot meet specific needs and circumstances, and require amendments. Moreover, LLMs are almost invariably viewed as “black box”, and the outputs are less explainable. For example, there is a phenomenon called hallucination that LLMs output results are not realistic, do not follow user given context or match any data patterns that it has been trained on. There is a need to identify any missed aspects or extraneous aspects in order to help users to improve the quality of the clauses and reduce risks.

To the best of our knowledge, this may be the first effort towards safe use of generative AI by performing post-validation of the machine-generated contract clauses using machine learning approaches.

Figure 1 describes the overall process of our proposed method for analyzing raw clauses. Detailed description is given in Section 5.

The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 introduces the working principles of LLMs and Section 4 illustrates practical prompts that can be used to generate clauses. Section 5 describes the approach for analyzing and comparing generated clauses against similar clauses retrieved from a trusted source. Section 6 introduces the used dataset and evaluation methods and presents the

experimental results. Section 7 presents the conclusions.

2. Related work

Contract clause generation has become one of the research focuses when transformer decoders [4] are introduced. Aggarwal et al. [5] proposed CLAUSEREC framework for clause type and clause content recommendation. It predicts the clause type and generate the corresponding content that users can consider to add to an incomplete contract. In their work, the recommended clauses are generated by a transformer decoder based on the vector representations of the clause type and the incomplete contract. Joshi et al. [6] further enhanced this work by identifying similar contracts and adding the similar contract representations to the clause generation process.

Regarding clause keyphrases (including keywords) identification, [7] investigated the performance of various traditional machine learning approaches including logistic regression, SVMs [8] and manually written rules. On the other hand, [9] introduced a BERT based model called ALeaseBERT fine-tuned on lease agreements for the extraction of specified types of keyphrases and red flags, if exist.

Unlike the aforementioned work, this paper explores the feasibility of using LLMs to generate contract clauses which have much larger model sizes, compared to BERT [10], and trained on huge amount of public domain data or internet data and fine-tuned on large amount of various tasks data, not solely fine-tuned on datasets containing contract precedents. Our clause analysis work is also different from the above work. Apart from extracting the keyphrases, from individual clauses, the extracted keyphrases are further clustered so that keyphrases with different wordings but having similar meanings can be grouped together. This process enables clause comparison to be carried out more effectively, not just depending on the spelling of words.

3. Large language models (LLMs)

The main task of a language model is to model the generation of languages. Most recent LLMs are based on the neural network architectures called Transformers [4]. They are trained on huge amounts of text data which enables them to capture the long range dependencies within text

and hence generate coherent content such that the generated texts are closely related to each other. This is usually done in the way that the model predicts the next word based on the previously generated words such that the generated content is grammatically correct and coherent. Figure 2 shows two generated sentences: “How are you” and “How do you”. The generation process is probabilistic. Even when the first word “How” is the same for the two sentences, the second word can be different. Depending on the text that is sampled, the second word can be some other word that is commonly found following the word “How”. In order to train the language models to be grammatically correct, a huge dataset of sentences is required.

Having just grammatically correct language models may not be useful. People prefer the generated content to be coherent and closely related to user given prompts. The desired generation of the word w_n can be described as

$$P(w_n | w_{n-1}, w_{n-2}, \dots, \text{prompt}), \quad (1)$$

where w_{n-1}, w_{n-2}, \dots are the previously generated words and *prompt* is the text string that contains:

1. User instruction,
2. Context, and
3. Examples.

Figure 3. illustrates an example that how a sentence “Planes have wings” to be generated, given the prompt “Write a sentence about a vehicle.”. This prompt only has the “user instruction” whereas context and examples are not given to the model. In order to train the language models to follow user instruction, a lot of labeled data (i.e. prompts with desired outputs) are required. Furthermore, for more sophisticated structural tasks such as the generation of a contract or a business plan, corresponding training data should be used for further training.

Apart from different types of training data, the language models must be sufficiently large to follow the instructions and understanding the context and examples. Other factors that have significant impacts on the language model performance are the language model structure, constituting components, training methods, and the loss functions. Different models’ training methods and data can derive different

implementations of the equation (1) and hence different text generation. Some efficient models such as the Vicuna [11] models, based on LLaMA, can be fine-tuned to have about 90% performance of the ChatGPT with just about 7% the number of parameters of ChatGPT.

4. Prompt writing for generating legal contracts and clauses

As most LLMs are capable of multiple tasks, such as sentiment classification, question and answering, information extraction, and text generation, users are required to provide the prompt to instruct models so that corresponding responses can be generated. There are few aspects that should be noted for writing a prompt:

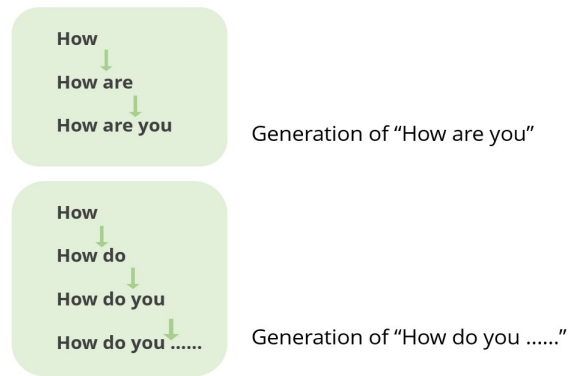


Figure 2: Generation of the sentences “How are you” and “How do you” by a language model. Each word is sampled from a vocabulary given the previously generated words.

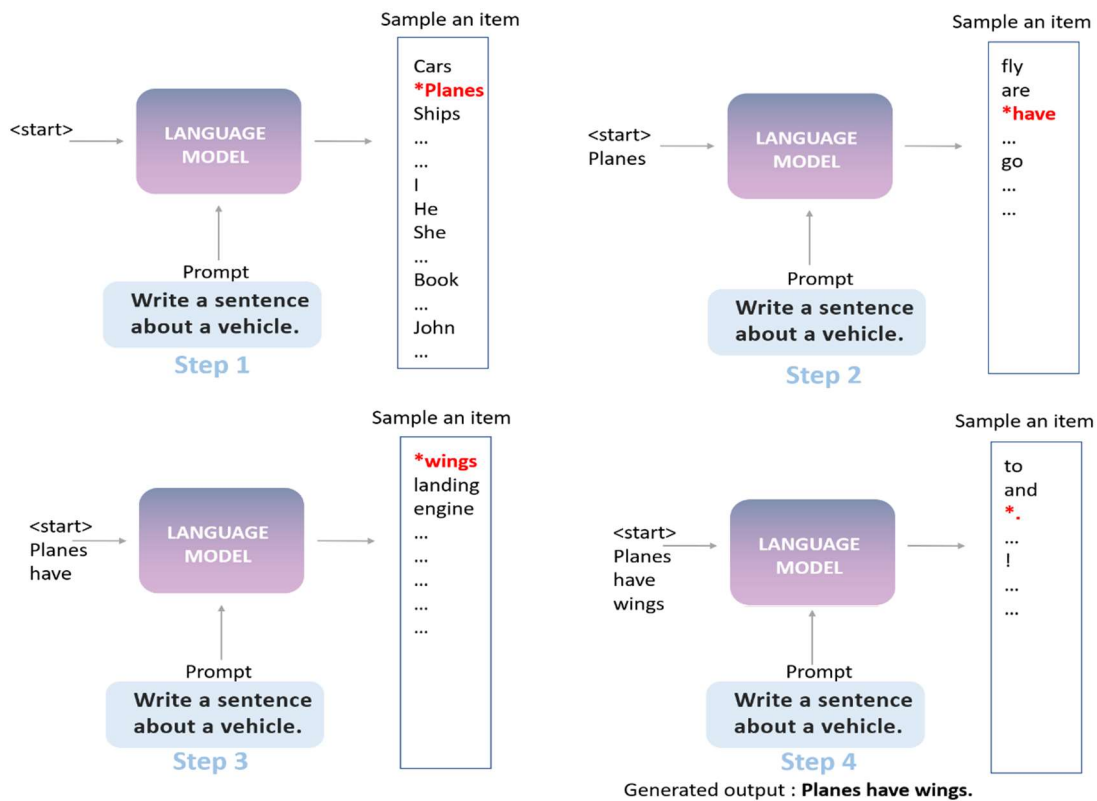


Figure 3: Example of using a prompt “write a sentence about a vehicle” to instruct a language model to generate a related sentence. “<start>”: is the sentence beginning tag. Step 1: Sample a word from a vocabulary, with words of vehicles having higher probabilities. The word “Planes” is sampled (marked with “*”). Step 2: Given the prompt and the first word “Planes”, words “fly”, “are”, “have” have higher probabilities to be the next word, and “have” is sampled. Step 3: Given the prompt and the sampled words “Planes” and “have”, the top probability next words are “wings”, “landing”, “engine” and the word “wings” is sampled. Step 4: Given the prompt and the sampled words “Planes”, “have” and “wings”, the words/punctuation with higher probabilities to be the next sampled entity are “to”, “and”, and “.”. The punctuation “.” is sampled in this case. Since a sentence is generated, the language model stops here.

- Clear task description,
- Style,
- Fairness, no bias, ethical, and not harmful.

For contract clause generation, prompts may be:

*Draft a legal clause for [task], or
Draft a contract clause between [person1]
and [person2] for [task]*

In the examples above, either using the word “draft” or “generate” is sufficient to indicate the type of the prompt. The words “legal” or “contract” should be included which enable the LLMs to use legal or contract styles. If users want to generate a clause that follow a list of specifics that form the context, the context should be included to the prompt as the following example.

{{{context text}}}
- *Based on the previous information, draft a legal clause to [task]*
- *The clause should include [entities]*

In the above clause, the text describing the context should be enclosed by “{{{” and “}}””. It is because they help the LLMs to differentiate the context from other text. Also, if there are additional conditions augmented to the clauses, users should specify them in a clear format.

5. Analysis and evaluation of the generated clauses

As clauses the legally binding and define the rights, obligations, and agreements between the parties in the contract, they should be clearly and precisely drafted. In contrast, the generation of clauses with LLMs is a black box process which currently lacks explainability. The hallucination phenomenon of LLMs even makes the situation worse because models can generate results that are not related to the prompt. Thus, there is a need to validate the AI generated clauses to match the user requirements. A common and secure method is the human-in-the-loop [12] in which a legal professional intervenes in the generation process by either (a) revising the generated clauses directly or (b) revising the model input prompt and direct the model to regenerate the clauses and hope better clauses can be regenerated. This presupposes that the person revising the generated clause or making the final selection is

knowledgeable. To assist the person revising or making the selection, we propose using a machine learning approach to identify various aspects of a clause and compare them with aspects obtained from clauses extracted from trusted sources. The benefit of this approach is that the differences, in aspects, between the generated clauses and selected clauses can be automatically identified. This can help to revise the generated clauses to ensure consistency and quality with reference to the trusted source.

Before introducing the proposed approach, we briefly describe 3 important tools that are used in the approach.

1. **Sentence transformers** [13]: for identifying clauses with similar contents. We use it because it is fine tuned to perform similarity measure between two sentences.
2. **PatternRank** [14]: extracting keyphrases from sentences.
3. **Uniform manifold approximation and projection (UMAP)** [15]: for manifold approximation and projection.

5.1. Sentence transformers

Sentence transformers are based on Siamese neural networks for obtaining embeddings (i.e. vector representations) of sentences or paragraphs. A sentence transformer uses a pre-trained language model such as BERT or RoBERTa as its core neural network component and fine tune it with a large labeled dataset which enhances its capability to differentiate which relationship a pair of sentences should be, one of the following:

- 1) Entailment, when the first sentence is true, the second sentence is also true;
- 2) Neutral, the truth of the sentences is independent of each other;
- 3) Contradiction, the sentences contradict with each other.

After fine tuning, the sentence transformer can transform sentences to vectors such that sentences with similar contents will have vectors

close to each other, whereas sentences with different contents will have vectors far apart from each other. Hence, given vector representations of a pair of sentences, the similarity can be evaluated by using cosine similarity.

5.2. PatternRank

PatternRank performs keyphrases extraction by finding the sub-phrases in a document that are the most similar to the input document itself.

The working principle is that it first uses a sentence transformer to get a document-level vector representation and vector embeddings of N-gram phrases appearing in the input document. Then cosine similarity is used to find the top phrases that are most similar to the input documents.

It basically is an enhancement of KeyBERT [16]. It allows user to define the patterns of phrases to be extracted instead of simple n-grams of pre-defined lengths.

5.3. UMAP + DBSCAN

Since the vectors produced from sentence transformers have relatively high dimension, 384 dimensional vectors in our experiments. Performing clustering directly on the vectors with common algorithms such as K-Means may give non-satisfactory results because of the curse of dimensionality. First the vectors may be distributed in a low dimensional manifold space embedded in a high dimensional space. Secondly, the distance difference between nearest data points and farthest data points decreases as the dimension increases.

UMAP is a tool for addressing high dimensional and manifold distributed data. The data are first non-linearly projected to a low dimensional space and then a density-based clustering (e.g., DBSCAN [17]) is performed.

5.4. Clause analysis and evaluation

Referring to Figure 1, an input clause is first obtained from a contract template or an LLM, and a set of similar clauses is obtained from a trusted sources (e.g., a repository of examined clauses) by using a sentence transformer and cosine similarity. Since the similar clauses are trusted and reliable

clauses, they contain significant and desirable aspects that the user should check against with the input clause. In order to explicitly extract these aspects from the similar clauses, we propose to perform the following procedure:

1. Extract keyphrases from the input clause and similar clauses separately by PatternRank.
2. Use sentence transformer to transform all of the keyphrases to vectors.
3. Perform a clustering solely on the vectors coming from similar clauses by UMAP and DBSCAN.
4. Compute the distance between the keyphrase vectors of the input clauses to the cluster centers.
5. Assign the keyphrase vectors of the input clause to *cluster x* if distance between the keyphrase vector and *cluster x* < *threshold value*. Those clusters containing the keyphrases from the input clause are labeled as **Involved Clusters (Involved Aspects)**.
6. Label the keyphrases as **Extra Keyphrases** if it cannot be assigned to any clusters.
7. Clusters without keyphrase vector from the input clause are labeled as **Missed Clusters (Missed Aspects)**, because the input clause has no keyphrase related to these clusters.

Here we have made an assumption that an aspect of a clause is closely related to the keyphrases appearing in that clause. Individual clusters or groups of these keyphrases represent the aspects of the extracted similar clauses. Finally, user may augment the input clause with the aspects represented by Missed Clusters and to decide whether to preserve the Extra Keyphrases or not as they are not found in similar clauses.

6. Experiments

We investigate the quality of generated contract clauses by comparing them with the clauses of the trusted dataset LEDGAR [18]. The quality is assessed by using ROUGE F1 [19], precision, recall and F1. ROUGE F1 is a widely used tool for text content comparison, whereas precision, recall and F1 are used for assessing the extent of aspects to be addressed by the model generated clauses. They are evaluated as follows:

1. ROUGE F1 score: For each machine generated clause, we compute the F1 scores of ROUGE-1, ROUGE-2, and ROUGE-L with respect to the 15 extracted similar clauses from LEDGAR, having cosine similarity > 0.5. The means of the scores are reported.
2. Precision, Recall, and F1: For each machine generated clause, we compute the precision, recall, and F1 with respect to the aspects derived from the 15 extracted similar clauses from LEDGAR. The precision and recall are defined as follows:

$$\text{Precision} = \frac{\# \text{Involved Aspects}}{\# \text{keyphrases from generated clause}} \quad (2)$$

$$\text{Recall} = \frac{\# \text{Involved Aspects}}{\text{Total No. of Aspects}} \quad (3)$$

In the experiments, we generate the clauses with ChatGPT and a local LLM called Vicuna which also has an interactive text input and output similar to ChatGPT. Since both models also generate the text of description of the tasks and legal disclaimers in addition to the generated clauses, we remove all these text before the evaluation.

6.1. Data

The public domain dataset LEDGAR was introduced by Tuggener et al. at 2020. It was crawled and scraped from the U.S. Securities and Exchange Commission filings (SEC filings) and contains contract clauses between years 2016-2019. These contracts are mainly material contracts (called Exhibit-10) including shareholder agreements, employment agreements, and non-disclosure agreements. After data cleanup and preprocessing, the dataset contains 60,540 contracts and a total of 846,274 clauses.

6.2. ChatGPT, Vicuna and other component settings

In the evaluation, the ChatGPT used is the March 23 version. For the Vicuna model, we use the “llama.cpp” to interact with the model because

“llama.cpp” is a pure C/C++ implementation with the support for 4-bit quantization of the model. This allows the quantized Vicuna model to be assessed with a general 12GB RAM notebook, even without a GPU. These features are very desirable because people with privacy concern can have a ChatGPT-like application to be executed locally. The model we used is the fine-tuned and 4-bit quantized 13B LLaMA model and the important program configuration parameters are given below:

- Temperature: 0.7,
- Top_k: 40,
- Top_p: 0.8,
- Repeat_penalty: 1.2,
- Prompt context length: 2048,

and use default values for other parameters. For the sentence transformer, the model used is all-MiniLM-L6-v2, which transforms input sentences to 384 dimensional vectors.

For PatternRank, top 10 ranked keyphrases are used. We also define a more complex POS patterns to extract various types of keyphrases in clauses:

```
(<J.*|V.*>+<CC.*>)*<J.*|V.*>*<N.*>+
```

For UMAP, number of neighbors is set to 5, minimum distance between two projected data points is 0.0 and the projected dimension is 5. For the DBSCAN, default parameters are used except the minimum samples to form a cluster is set to 4.

6.3. Clause generation

The test clauses are generated manually by using the ChatGPT’s and Vicuna’s interactive interfaces. We generated the clauses for a house rental agreement and a sales agreement with the prompts shown in Table 1. In the table, a simple context “For a house rental agreement:” is only given for the 1st prompt. The LLMs can refer back to the context for the rest prompts. We had tested the generation of a total of 9 clause types for the rental agreement. Similarly, for the sales agreement, the 1st prompt has a context “For a sales agreement:” and a total of 8 clause types are generated. For each clause type, we generate 5 sample clauses. Hence, there are in total 85 model generated clauses for testing. For ChatGPT, multiple samples are obtained by regeneration

whereas for Vicuna, the samples are obtained by changing the randomization seed.

Table 1: The prompts used to generate 9 types of contract clauses for a house rental agreement. E.g., For generating the 3rd type clauses, the prompt used is “Draft a legal clause for security deposit”.

1st prompt	For a house rental agreement: draft a legal clause for the term of agreement
Other prompts	Draft a legal clause for the [clause type]
clause types for other 8 prompts	<ul style="list-style-type: none"> • rent and payment method • security deposit • late payment of rent • utilities of the property • maintenance and damages of the property • occupancy and sublease of the property • miscellaneous provisions for the agreement • government law

6.4. Results

Table 3 summarizes the mean ROUGE F1 scores of the generated clauses for a house rental agreement and a sales agreement, using the prompts given in Table 1 and 2, respectively. The results shows that the scores are relatively low. The low values point out that the vocabulary and

Table 2: The prompts used to generate 8 types of contract clauses for a sales agreement. E.g., For generating the 4-th type clauses, the prompt used is “Draft a legal clause for inspection and acceptance”.

1st prompt	For a sales agreement: draft a legal clause for the term of agreement
Other prompts	Draft a legal clause for [clause type]
Clause types for other 7 prompts	<ul style="list-style-type: none"> • product(s) subject to sale • delivery of the product(s) • inspection and acceptance • price and payment • mechanical warranty • force majeure • termination

the styles used in the generated clauses are different from the extracted similar clauses. It is expected as the LLMs are pre-trained on tremendous amount of text, in addition to legal text. Moreover, the generated clauses are just based on the simple prompts and thus the clauses are generated for general situations. This means that the requirement of attorneys to review and tailor clauses to the specific circumstances of the parties involved is still necessary.

Table 4 shows the mean precision, mean recall and mean F1 score of the LLM generated clauses for the two agreements. The results show that considerable aspects of the related and trusted clauses are addressed by the generated clauses. This indicated LLM generated clauses can cater for many aspects of contract clauses and can be used as an initial point for drafting clauses, even there are still some enhancements required. When comparing the performance of the ChatGPT and Vicuna, it is quite clear that the precision of the ChatGPT on the tasks is significantly higher than Vicuna, given that the Recall values are similar. This is also expected as model size of ChatGPT is much larger than that of Vicuna.

Finally, Tables 5 and 6 show two samples (one for ChatGPT and one for Vicuna) to illustrate the generated clauses, derived aspects, Involved Aspects, Missed Aspects, and Extra Keyphrases. With this information, users can consider to revise their clauses, if necessary.

Table 3: Mean ROUGE F1 scores of the LLM generated clauses for a house rental agreement and a sales agreement.

House Rental Agreement			
Model	Mean ROUGE-1	Mean ROUGE-2	Mean ROUGE-L
ChatGPT	0.32	0.10	0.28
Vicuna	0.32	0.11	0.28
Sales Agreement			
Model	Mean ROUGE-1	Mean ROUGE-2	Mean ROUGE-L
ChatGPT	0.30	0.09	0.25
Vicuna	0.29	0.09	0.25

Table 4: Mean Precision, Recall, and F1 scores of the LLM generated clauses for a house rental agreement and a sales agreement.

House Rental Agreement			
Model	Mean Precision	Mean Recall	Mean F1
ChatGPT	0.77	0.61	0.68
Vicuna	0.60	0.62	0.61

Sales Agreement			
Model	Mean Precision	Mean Recall	Mean F1
ChatGPT	0.72	0.59	0.65
Vicuna	0.60	0.61	0.58

Table 5: The clause generated with the prompt “draft a legal clause for the late payment of the rent” using ChatGPT. The derived aspects (using similar clauses from LEDGAR dataset), Involved Aspects, Missed Aspects, and Extra Keyphrases are illustrated.

Model	Prompt: draft a legal clause for the late payment of the rent
ChatGPT	<p>Tenant acknowledges that the timely payment of rent is of the essence of this agreement. If Tenant fails to pay rent in full on or before the due date specified in this agreement, Tenant shall be considered in default under this agreement. If rent is not paid on or before the due date, Tenant shall be subject to a late fee of [insert late fee amount or percentage] per day until the rent is paid in full. The late fee shall be in addition to any other charges or damages owed by Tenant under this agreement. If Tenant fails to pay rent within [insert number of days] days after the due date, Landlord may, at Landlord's sole discretion, give Tenant a notice of termination of this agreement, and Tenant shall vacate the premises within [insert number of days] days from the date of the notice. In addition to any other remedies available under applicable law, the Landlord may also seek damages for any unpaid rent or other amounts owed by Tenant under this agreement. Tenant acknowledges that the late payment of rent may cause the Landlord to incur additional costs and expenses, including but not limited to, late fees, legal fees, and other costs associated with collecting rent. Tenant agrees to indemnify and hold harmless the Landlord from any and all such costs and expenses incurred by the Landlord as a result of Tenant's failure to pay rent in a timely manner.</p> <p>Involved Aspects: 0, 1, 8, 9, 10, 11, 13 Missed Aspects: 2, 3, 4, 5, 6, 7, 12 Extra Keyphrases: “unpaid rent”</p>
Aspect 0	'late payment', 'subsequent late payments', 'first late payment'
Aspect 1	'late charges', 'pay such late charge', 'late charge therefor'
Aspect 2	'pay landlord', 'compensates landlord', 'costs landlord'
Aspect 3	'incur costs', 'attorney fees', 'accounting charges'
Aspect 4	'additional minimum rent', 'additional rent', 'basic lease terms'
Aspect 5	'rent payment', 'rent adjustment deposits', 'real property taxes',
Aspect 6	'default rate', 'overdue amount', 'overdue payment'
Aspect 7	'installment', 'bear interest', 'pay such interest'
Aspect 8	'notice', 'written notice', 'foregoing notice'
Aspect 9	'grant tenant', 'tenant', 'gives tenant notice'
Aspect 10	'liquidated damages', 'damage', 'actual damages'
Aspect 11	'due date', 'applicable due date'
Aspect 12	'payment', 'creditor relationship', 'future payments'
Aspect 13	'grace period', 'cure period', 'month period'

Table 6: The clause generated with the prompt “draft a legal clause for force majeure” using Vicuna.

Model	Prompt: Draft a legal clause for force majeure
Vicuna	Neither party shall be liable to the other for any delay or failure to perform its obligations under this Sales Agreement if such delay or failure arises from circumstances beyond its control, including but not limited to acts of God, fire, flood, earthquake, riot, civil commotion, war, invasion, act of government, sabotage, labor shortages, or failures of the Internet. If either party is prevented from performing any obligation by such circumstances, it shall give prompt notice thereof to the other party and make a reasonable effort to minimize the impact of such events on its performance.
	Involved Aspects: 0, 1, 2, 3, 4, 8, 10 Missed Aspects: 5, 6, 7, 9 Extra Keyphrases: 'give prompt notice', 'God'
Aspect 0	'agreement', 'written notice'
Aspect 1	'delays', 'manufacturer delays', 'excusable delay'
Aspect 2	'failure', 'causing cessation', 'communication failures'
Aspect 3	'affected party', 'parties', 'other party'
Aspect 4	'obligations', 'respective obligations', 'payment obligations'
Aspect 5	'liability', 'damages', 'negligence'
Aspect 6	'circumstance', 'present or future law', 'event'
Aspect 7	'limitation', 'governmental action', 'governmental regulation'
Aspect 8	'labor troubles', 'work stoppages', 'labor difficulties'
Aspect 9	'terrorism', 'including disruption', 'cyber attack'
Aspect 10	'limitation earthquakes', 'earthquake', 'severe weather conditions'

7. Conclusions

The launch of ChatGPT and the emergence of LLMs have tremendous impacts, both as an opportunity and as a threat, on different aspects of digital transformation. In this paper, we introduce the underlying working principle and investigate their performance on contract clause drafting. We also propose an approach to enhance these clauses by using a trusted contract clause source. Experimental results show that the generated clauses are usable as they can address a significant number of related aspects that usually appear in clauses of practical contracts.

8. Acknowledgement

This research / project is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

9. References

- [1] D. Simonson, D. Broderick, J. Herr, The extent of repetition in contract language, in: Proceedings of the Natural Language Processing Workshop 2019, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 21–30.
- [2] ChatGPT, personal communication, March 23, 2023.
- [3] H. Touvron et al., LLaMA: Open and efficient foundation language models, arXiv:2302.13971, 2023. URL: <https://arxiv.org/abs/2302.13971>.
- [4] A. Vaswani et al., Attention is all you need, in: Proceedings of Neural Information Processing Systems, 2017, pp. 5998-6008.
- [5] V. Aggarwal et al., A Clause Recommendation Framework for AI-aided Contract Authoring, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 8770-8776.
- [6] S. Joshi et al., Investigating strategies for clause recommendation, Legal Knowledge and Information Systems, 2022. doi:10.3233/FAIA220450.
- [7] I. Chalkidis, A. Michos, Extracting contract elements, in: Proceedings of the 16th Edition of the International Conference on Artificial

- Intelligence and Law, 2017, pp. 19-28. doi: 10.1145/3086512.3086515.
- [8] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.
- [9] S. Leivaditi, J. Rossi, E. Kanoulas, A benchmark for lease contract review, arXiv:2010.10386, 2020. URL: <https://arxiv.org/abs/2010.10386>.
- [10] J. Devlin et al., BERT: Pre-training of deep bidirectional transformers for language understanding, arXiv:1810.04805v2, 2019. URL: <https://arxiv.org/abs/1810.04805v2>.
- [11] W. L. Chiang et al., Vicuna: An open-source chatbot impressing GPT-4 with 90% ChatGPT quality, 2023. URL: <https://vicuna.lmsys.org>.
- [12] Wikipedia contributors, Human-in-the-loop, Wikipedia, The Free Encyclopedia, 2023. URL: <https://en.wikipedia.org/w/index.php?title=Human-in-the-loop&oldid=1145872129>
- [13] N. Reimers, 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, Association for Computational Linguistics, 2018, pp. 3982-3992.
- [14] T. Schopf, S. Klimek, F. Matthes, PatternRank: Leveraging pretrained language models and part of speech for unsupervised keyphrase extraction, in: Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, 2022, pp. 243-248. doi: 10.5220/0011546600003335.
- [15] L. McInnes, J. Healy, J. Melville, UMAP: Uniform Manifold Approximation and Projection for Dimension and Reduction, arXiv:1802.03426, 2018. URL: <https://arxiv.org/abs/1802.03426>.
- [16] M. Grootendorst, KeyBERT: Minimal keyword extraction with BERT, Zenodo, 2020. doi: 10.5281/zenodo.4461265.
- [17] M. Ester et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226-231.
- [18] D. Tuggener et al., LEDGAR: A large-scale multi-label corpus for text classification of legal provisions in contracts, in: Proceedings of the 12th Language Resources and Evaluation Conference, 2020, pp. 1235-1241.
- [19] C. Y. Lin, ROUGE: A package for automatic evaluation of summaries, In: Text Summarization Branches Out, Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 74-81.