Graph-based Keyword Planning for Legal Clause Generation from Topics

Sagar Joshi[*], **Sumanth Balaji**[*]
IIIT Hyderabad
{sagar.joshi,sumanth.balaji}@research.iiit.ac.in

Aparna Garimella
Adobe Research
garimell@adobe.com

Vasudeva Varma
IIIT Hyderabad
vv@iiit.ac.in

Abstract

Generating domain-specific content such as legal clauses based on minimal user-provided information can be of significant benefit in automating legal contract generation. In this paper, we propose a controllable graph-based mechanism that can generate legal clauses using only the topic or type of the legal clauses. Our pipeline consists of two stages involving a graph-based planner followed by a clause generator. The planner outlines the content of a legal clause as a sequence of keywords in the order of generic to more specific clause information based on the input topic using a controllable graph-based mechanism. The generation stage takes in a given plan and generates a clause. The pipeline consists of a graph-based planner followed by text generation. We illustrate the effectiveness of our proposed two-stage approach on a broad set of clause topics in contracts.

1 Introduction

Contracts are essential discourse units that frequent in several day-to-day business workflows, especially between companies and governmental organizations. The fundamental units of discourse in contracts consist of "clauses" that are paragraphs of text that outline the terms and conditions of various types or *topics* (e.g., *severability*, *benefits*) (Table 1). Legal clauses can be characterized by their high inter-sentence similarity, and topic-specific content (Simonson et al., 2019). For example, Zhong et al. (2020) showed that the sentences in legal corpora are almost 20% similar to each other. Drafting contracts by legal counsel is a manual process of taking a skeletal set of clauses and adding or modifying them for the contract goal. Given their highly domain-specific content and unique linguistic structure, contract drafters in legal counsel can significantly benefit from applying Natural Lan-

---

In case any provision herein or obligation hereunder or any Note or other Credit Document shall be invalid, illegal, or unenforceable in any jurisdiction, the validity, legality and enforceability of the remaining provisions or obligations, or of such provision or obligation in any other jurisdiction, shall not in any way be affected or impaired thereby.

Table 1: An example *severability* clause from a legal contract.

guage Processing (NLP) techniques to aid contract creation (Zhong et al., 2020).

There have been recent advances in Transformer-based (Vaswani et al., 2017) methods for text generation in varied flavors, such as prompt-based causal generation (Radford et al., 2019), conditional generation based on control codes (Keskar et al., 2019), and retrieval-augmented generation based on queries (Lewis et al., 2020b). However, these methods are primarily studied in generic NLP domains, and legal text generation remains largely unexplored. The only previous work that addressed the task of legal text generation is CLAUSEREC (Aggarwal et al., 2021), in which a Transformer-based decoder is trained to generate missing legal clauses in a given contract document, conditioned on the clause topic and the content in the contract. However, Aggarwal et al. (2021) noted that the clauses generated by CLAUSEREC suffer from low linguistic variations within topics, thus resulting in content that is thematically relevant but missing a few nuances. In general, we believe conditioning text generation on only the high-level clause topics or the contract content may not capture the subtleties present in legal clauses, hence call for an iterative approach to learn the clause-specific content in a top-down manner.

We find inspiration in the content planning paradigm for story generation (McIntyre and Lapata, 2010; Yao et al., 2018; Chen et al., 2021) in which an intermediate plan (e.g., a set of keywords) is used to generate final stories. In this paper, we study legal text generation, and propose a two-staged pipeline (Figure 1) to generate le-

---

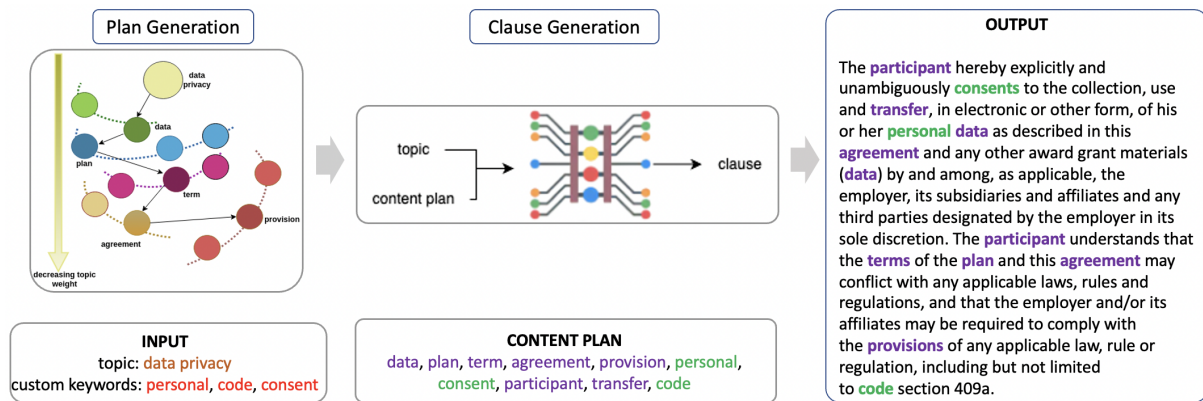[*]Both the authors have equal contribution to this work.

Figure 1: The proposed 2-stage clause generation pipeline: User specifies the input topic (*data privacy*) along with a set of keywords for customization (*personal, code, consent*). The first stage involves generating a customized content plan, including the custom keywords (green) along with other plan keywords derived purely from a graph walk (purple). In the second stage, a clause generator interpolates through the keywords in the plan to produce a coherent, meaningful legal clause for the given topic.

gal clauses from topics iteratively using keywords. Specifically, the first module of our pipeline comprises of a graph-based planner that takes in a topic (and an optional set of keywords) and produces an ordered content plan consisting of keywords, with those more generic to the topic ranked higher, and those more specific to a clause ranked lower. Note that unlike (Aggarwal et al., 2021), which aimed to generate missing clauses given a contract, we focus on generating legal clauses only based on a given topic and an optional set of keywords. We approximate the generated content plan to the user intent, which will be translated into a legal clause. In the second stage, the plan is used as a control mechanism, and a Transformer-based language model is trained to generate legal clauses conditioned on the topic and the plan.[1]

Following are the main contributions of this paper: **(1)** We propose a novel two-staged pipeline for legal clause generation, comprising a content planner that generates a keyword-based plan, and a content generator that generates legal clauses conditioned on the clause topics and keyword plans. Our proposed content planner consists of a simple, lightweight graph-based mechanism that performs a graph walk using the input topic to generate a plan consisting of generic to specific keywords. The plan can be customized by specifying a few control keywords, bringing controllability to the generation. **(2)** We compare our approach with several conditional and causal text generation baselines, and illustrate strong empirical results for le-

gal clause generation. **(3)** We also show that our approach can be generalized well to a diverse range of clause topics, thus indicating the extensibility of our approach for legal text generation. We believe our work takes one step further in the area of automatic clause generation for AI-aided drafting of legal documents.

2 Graph-based Planning for Clause Generation

Our proposed approach aims to generate legal clauses to aid legal counsel in contract drafting. To do so, it takes as input a clause topic (e.g., *data privacy*) along with a few keywords for customization (*personal*, *code*, *consent*), and generates more keywords in order to obtain a customized content plan as illustrated in Figure 1. This keyword-based plan, which we consider an approximation to user-specified keywords per their preferences, is then used to generate a meaningful legal clause.

2.1 Dataset creation

Ranked keyword extraction per topic. For every clause topic t from the set of all clause topics T, we extract an ordered set of keywords $K^t = \{k_1^t, k_2^t, \ldots, k_{m_1}^t\}$ representing the topic using an off-the-shelf keyword extractor. Each extracted keyword is a single, comprehensible word unit occurring in a clause. The ordered set of keywords under a topic represents the salient words under that topic, approximately ranked based on their prominence. In the ranked order, the words more generic to the topic (perceived to carry more

---

[1]The code for this work can be found here: https://github.com/sagarsj42/legal_clause_gen_from_topic_keywords
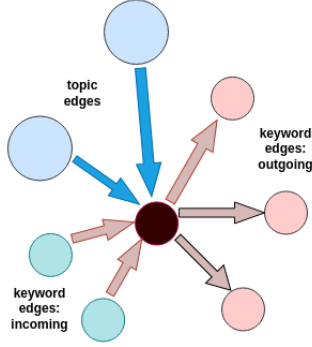
Figure 2: View of the types of node connections in the directed graph G.

information about the topic's generic form[2]) are ranked higher. The ones less generic to the topic but more specific to individual clauses (perceived to be more characteristic of an individual clause) are ranked lower. The keywords are lemmatized using a WordNet-based lemmatizer.

Reference keyword plans for clauses. Each clause is represented by a reference keyword plan which consists of a ranked list of keywords corresponding to the topic of that clause. For every clause in a topic, we check for the existence of each ranked topic keyword and sequentially add them as plan keywords, thus preparing a ranked plan of keywords appearing in the clause. Applying for all the topics, we have a dataset of clause-keyword plans $D^t = (C^t, R^t)$ for each topic t in the dataset. C^t represents the set of clauses $\{c_1^t, c_2^t, ..., c_{m_2}^t\}$ under the topic, and R^t represents the set of corresponding reference keyword plans for the clauses, with the plan for a clause c being a ranked list of keywords, $r^c = [r_i^c]_{i=1}^{i=n}$ (where r_i^c represents keyword selected for each stage i).

2.2 Graph construction

A single, directed graph G is constructed to capture the keyword plan information from all the topics in a unified representation as illustrated in Algorithm 1. The graph G is initialized with the set of nodes $N = T \cup K$ consisting of all the topics T and an accumulated set of keywords from the topics, $K = \bigcup_{t \in T} K^t$. Each node in the graph has incoming connections from relevant topic nodes along with incoming and outgoing connections to keyword nodes, as shown in Figure 2.

Edges weights between every pair of topic-keyword and keyword-keyword nodes are calcu-

---

2By topic's generic form, we refer to the clause content that most commonly occurs across clauses under that topic, being characteristic to that topic.

Algorithm 1 Graph construction

Require: Topics T, keywords K, reference plans $R = \bigcup_{t \in T} R^t$
1: Initialize graph G with nodes $N \leftarrow T \cup K$
2: Edges $e(n_1, n_2) \leftarrow 0 \;\; \forall (n_1, n_2) \in N$
3: **for** topic $t \in T$ **do**
4:     Topic frequency, $f \leftarrow len(R^t)$
5:     **for** reference plan $r^t \in R^t$ **do**
6:         **for** step $s \leftarrow 1 \; to \; n$ **do**
7:             Step value, $v = 1/(s \cdot f)$
8:             $e(t, r_s^t) \leftarrow e(t, r_s^t) + v$
9:             $e(r_{s-1}^t, r_s^t) \leftarrow e(r_{s-1}^t, r_s^t) + v$
10:         **end for**
11:     **end for**
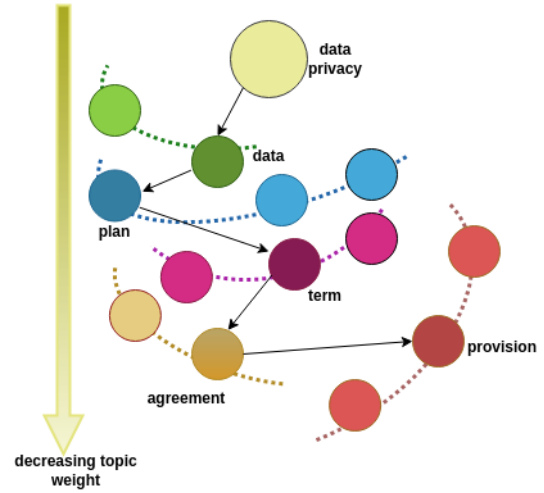12: **end for**
13: **return** G

Figure 3: Illustration of the first 5 stages of freeform plan generation, i.e., without using custom keywords for control. Given input topic: *data privacy*.

lated based on their occurrence in the train set as demonstrated in the algorithm. In this process, we walk through the reference plan for each clause under all the topics in the train set. A topic-keyword edge (t, r_s^t) is added for the occurrence of a keyword in a reference plan as a stage s of the plan. Similarly, a keyword-keyword edge $e(r_{s-1}^t, r_s^t)$ is added for the occurrence of consecutively occurring keywords r_{s-1} and r_s^t in the plan. Every occurrence of such topic-keyword or keyword-keyword pair adds an incremental weight to the corresponding edge. The weight given to each occurrence v is normalized by (1) no. of clauses f under that topic and (2) the stage value s at which the occurrence occurs. (1) accounts for the substantial imbalance in the clause type frequencies for the topics in the dataset, while (2) gives lesser importance to the keywords present at lower stages of the plan, thus statistically recording the generic to specific order within the edges of the graph.

278

Algorithm 2 Plan generation

Require: Graph G, topic t, stepwise thresholds TH, optional custom keywords q_c
1: **Initialize:** Plan $q \leftarrow []$; Current node, $cn \leftarrow t$
2: **for** step $s \leftarrow 1\ to\ n$ **do**
3:     Neighbors of current node, $N \leftarrow Neighbors(G, cn)$
4:     # candidates to score, $l \leftarrow len(N)$
5:     Neighbor scores, $S \leftarrow Zeros(l)$
6:     **for** $i \leftarrow 1\ to\ l$ **do**
7:         $cand \leftarrow Neighbors[i]$
8:         $S[i] \leftarrow G.edge\_score(t, cand) + G.edge\_score(cn, cand)$
9:     **end for**
10:     $SORT(N)$ by S, in descending order
11:     Top candidates, $TC \leftarrow N[: TH[i]]$
12:     **if** $\exists k \in q_c$ s.t. $k \in TC$ **then**
13:         $cn \leftarrow k$
14:         $q_c \leftarrow q_c - k$
15:     **else**
16:         $cn \leftarrow GET\_RANDOM(TC)$
17:     **end if**
18:     $APPEND(q, cn)$
19: **end for**
20: **return** q

2.3 Plan generation

Plan generation. Once we have the graph G, a plan q is generated at inference time by walking on the graph using the given input topic t as the starting point as shown in Algorithm 2. The provided input can also contain additional keywords q_c to be included in the plan, based on which the generated plan q can be customized.

A walk down the graph starts from the topic node t while selecting a keyword k from the best neighbors of each node. The selected neighbor then acts as the node for the next stage from which subsequent selection is to be made. This proceeds till we complete n stages of plan generation. For an appropriate selection, all neighbors of the current node cn are scored and ranked before making a selection. The window size for selecting a neighbor from the top-ranked ones at each stage is specified by the thresholds TH. For ranking the neighbors, each candidate $cand$ is scored based on the sum of their edge scores to the topic $G.edge\_score(t, cand)$ and the current node $G.edge\_score(cn, cand)$, thus facilitating the selection of keywords relevant to the topic and the current node context. At every stage, if the list of top-ranked candidates contains one of the custom keywords q_c, we directly select the custom keyword and remove it from q_c to avoid further repetition of that word in the plan. The generated plan can thus be given as:

$$q = [q_s]_{s=1}^{s=n} \quad (1)$$

2.4 Clause generation

Model training. We train a language model $LM(\theta)$ for generating a clause c by conditioning on a reference plan r^c and topic t, where θ are the parameters of LM. In this, we minimize the negative log-likelihood of the probability of c as given by the model:

$$L_{gen} = \sum_{t \in T} \sum_{c \in C^t} -log[\,p(c \mid r^c, t, \theta)\,] \quad (2)$$

This trained model can be used for generating a clause from a custom-generated plan q. Since the model has seen a large number of reference plans and their corresponding clauses, the model is expected to generate the right c for a q given by the planner.

Inferencing clauses from custom plans. At inference, we use the constructed graph G and the language model $LM(\theta)$. We expect a minimal input t indicating the topic of the clause to be generated along with an optional set of keywords q_{k_c} for customization. We run the plan generation algorithm based on this information to obtain a custom plan q as demonstrated in Algorithm 2. The customizability in plan generation can be exploited through an iterative plan-and-generate process involving iterative modifications to the plan before achieving a user-desired state of the clause. Appendix B illustrates such an example flow of plan modification followed by subsequent clause generation to enable an end user to achieve the clause in desired state.

3 Experiments

3.1 Dataset

We use the LEDGAR (Tuggener et al., 2020) dataset for our experiments. The cleaned version of this dataset consists of 60,540 contracts extracted from the EDGAR (Loukas et al., 2021) database containing 846,274 clauses (or "provisions") from 12,608 topics (or "labels"). We first create splits of the dataset at the contract level to ensure no data leakage in evaluation by making train, dev, and test sets made in a proportion of 85:5:10. We discard those clauses which belong to more than one topic in the subsequent experiments. From the train set contracts, we select those clause topics with a minimum clause frequency of 100, resulting in 387,210 clauses from 939 topics for training. We use these selected topics for identifying applicable clauses in the dev and test splits.

3.2 Keyword extraction and graph construction

We use the YAKE (Campos et al., 2020) keyword extractor for extracting keywords. Using YAKE allows us to extract a ranked order of keywords based on their prominence. The quality of ranked keywords given by the simpler statistical algorithm in YAKE was found to align with the notion of generic to specific information flow. To approximate the generic to specific order of keywords, we concatenate all the clauses under a topic and extract up to 200 (m_1) keywords per topic (K^t). These extracted keywords in ranked order represent each clause (c_i^t) as a reference plan of keywords (r^{c_i}) in which we limit the number of keywords per clause to 10 (n). The dataset of clause-plans ($\bigcup_{t \in T} D^t$) thus obtained is used to construct the graph (G) which consists of 267,893 edges ($e(.,.)$) and 46,953 nodes (N) - with a sparsity of 99.99% enabling it to be used as a lightweight mechanism for control.

3.3 Experimental settings

We experiment with pretrained GPT-2 (Radford et al., 2019) and BART (Lewis et al., 2020a) models for clause generation. Both models are trained on a batch size of 32 for 15 epochs. The learning rate schedule follows an initial warm-up till $1e-05$ for $1/4th$ of the total training steps, followed by linear decay. AdamW (Loshchilov and Hutter, 2019) optimization with a weight decay of 0.01 is used. The maximum generation length of the clause is kept to 700 tokens.

GPT-2 is trained in the usual causal generation paradigm in which we supply the topic concatenated with the plan as the prompt based on which the model generates a clause. For BART, conditional generation is employed in which a topic-plan concatenated input is supplied to the encoder, conditioning on which the model is trained to output a relevant clause.

3.4 Baselines

We consider the following baselines to evaluate the effectiveness of our plan-based approach for clause generation.

- **Prompt2Clause:** We consider the first 10 words of a clause as the prompt (plan) and fine-tune a GPT-2 model for clause generation following the usual causal generation paradigm.

- **Top2Clause:** We train a BART model to generate a clause solely conditioned on the topic of the clause.
- **RandKwd2Clause:** Keyword order is randomized in the plan, and supplied with the topic for BART-based conditional generation.
- **Plan2Clause-Retrieval:** We use the reference plans to retrieve from a TF-IDF-based index of clauses in the train dataset.

4 Results

4.1 Plan generation

	mean	median
rank	26.70	9.5
# neighbors	385.62	327

Table 2: Ranking generated plans based on references for estimation of plan quality generation.

To estimate the generated plans' quality, we walk the graph as shown in Algorithm 2. However, we use a reference plan to determine the rank given to the expected keyword at every stage of the plan. We take aggregated mean and median values of the ranks given at each stage across all the plans and compare them against the corresponding number of neighbor nodes encountered for ranking. As seen in Table 2, the graph walk gave a median rank of 9.5 against 327 neighbors. This shows the effectiveness of a simple, lightweight graph-based modeling for generating clause plans.

4.2 Clause generation

Experiment	BLEU	R-1	R-2	R-L
Prompt2Clause	20.2	19.68	12.56	16.1
Top2Clause	33.38	43.32	24.14	33.74
RandKwds2Clause	28.4	51.18	32.11	40.74
Plan2Clause-Retrieval	40.74	48.34	29.57	38.73
Plan2Clause-GPT2	39.18	48.24	29.73	39.25
Plan2Clause-BART	**48.98**	**58.99**	**37.95**	**46.11**

Table 3: Results of clause generation from plans.

We compare the baselines outlined in Section 3.4 against results based on our finetuned GPT-2 (Plan2Clause-GPT2) and BART (Plan2Clause-BART) models in Table 3. We use BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) metrics to evaluate the quality of clause generation based on the reference plans by comparing them against the reference (expected) clauses.

The BART-based generative model outperforms all the baselines consistently. The significantly lesser performance of the GPT-2-based approach

showcases the merit in conditional over causal generation for this problem. The difference in performance can also be observed in the Prompt2Clause (causal) and the Top2Clause (conditional) baselines. Plan2Clause-Retrieval turns out to be reasonably competitive, indicating the effectiveness of the proposed method for keyword planning and the potential advantage due to the similar nature of clauses. The importance of an ordered plan can be gauged from the poorer performance of the Rand-Kwds2Clause baseline.

Although CLAUSEREC is a pre-existing literature in the generation of legal clauses, it works on a contract-level problem of recommending a new clause to an incomplete contract in contrast to our clause-level problem. The significant difference in the input and nature of the problem hinders a direct comparison between the two approaches. However, our work merits in demonstrating extensibility (refer Section 5.1) over clause topics compared to the previous work which was limited to a selected set of 5 high-frequency topics.
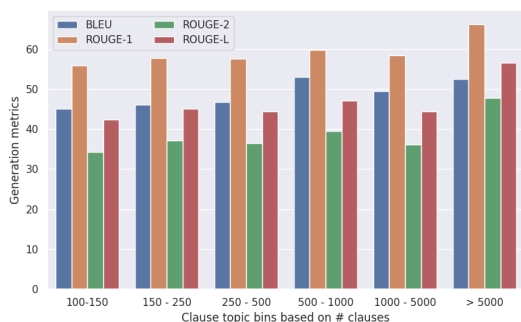
5 Analysis

Figure 4: Illustration of the robustness of pipeline to clause topics of various frequencies.

5.1 Robustness across clause topic frequencies

Table 5 shows statistics of the distribution of clause topics in the dataset for different bins based on their frequencies.

As can be seen, a significant variation exists in the nature of clause topics: a large number of clause topics with fewer clauses per topic coexist with a small number of clause topics with a very high number of clauses. We inspected the generation performance across multiple bins of clause frequencies to analyze the difference in generation quality. As seen in Figure 4, there is only a small difference in the performance of the lowest frequency bin compared to the highest. This is despite

the number of clauses under the lowest frequency topics being almost 2 orders of magnitude lesser than the highest frequency ones. This shows no significant bias towards the clause topics with a very high number of clauses, and the model can handle a diverse range of topics well. These observations demonstrate the extensibility of our approach to handle newer topics with fewer clauses per topic.

5.2 Comparison with sequential keyword order

We also ablate our proposed generic to specific order of keywords against a natural, sequential order for the planning and generation stages. Much of the existing literature uses a sequential content plan as the conditional prior for generation. This is natural, since the models typically work by generating the content sequentially based on the keyword information, which intends to plan a 'story' in that order.

5.2.1 Planning
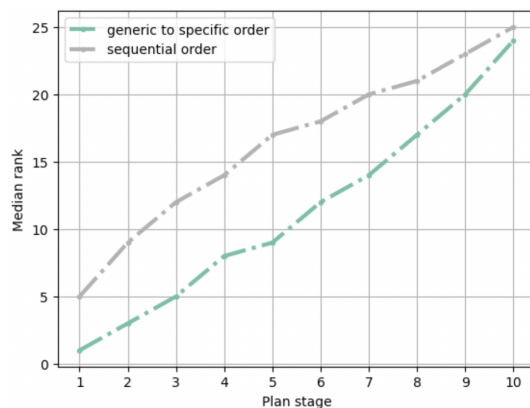
Figure 5: Comparison of stage-wise median ranks given to plans generated by generic to specific ordered keywords versus sequential keywords (the lower the better).

Figure 5 shows the median ranks given to the plans generated by our graph-based approach to each stage of the 10-stage plan. For generating plans with the sequential keywords approach, the first top 10 keywords from every clause were extracted to prepare the dataset $D^t \forall t \in T$, following which the same procedure was followed for graph generation and planning.

As can be observed from the figure, the proposed approach guided by topic-level information for keyword extraction performs better than the approach based on keyword extraction at a clause level for the initial stages of the plan. The lower ranks given

in the initial stages of planning highlight the predictive components in both the approaches. There are only a few possibilities for topic-generic content keywords for the proposed generic to specific approach. In the sequential approach, the initial lower ranks bring out the predictable nature of initial phrases in a legal clause. As we move to later stages, the gap between the two approaches decreases as the ranks increase. This demonstrates the loss of predictability as we move on to increase the no. of plan keywords.

5.2.2 Generation

Figure 6: Ablation analysis based on the number of keywords for clause generation & comparison of the proposed content plan order against the traditional sequential keyword order.
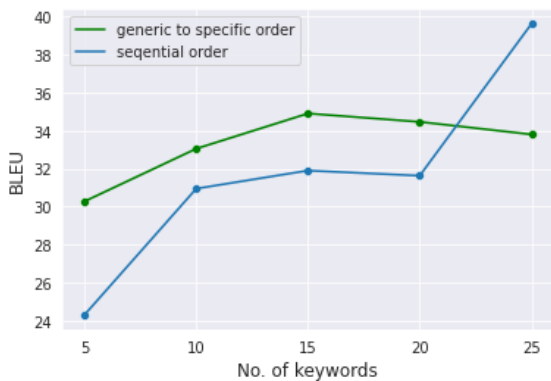
In order to study the impact of the no. of keywords provided for generation, we conducted an ablation analysis by repeating our experiments for clause generation on a BART-base model for 10 epochs each, keeping the rest of the hyperparameters the same. The no. of keywords was changed from 5 to 25 in step of 5 for conducting these experiments.

To contrast our method with the sequential keyword order, we repeated the ablation experiments for the same number of keywords while considering the content plan as a sequential order of keywords in the clause. We measured the performance across these studies using BLEU and have illustrated the results in Figure 6.

For our proposed keyword order, we found the generation performance to initially increase with an increase in the number of keywords but later follow a decreasing trend. As we increase the number of keywords for generation, the number of keywords specific to a clause increases. This could help initially since the generative model gets more context for generation. However, adding too many clause-specific keywords away from the topic confounds the model by supplying too much information. Another factor that could play here is the increased noisiness in keywords as we move down the ranked hierarchy, where the keywords keep getting less indicative of the content.

However, when providing the keywords sequentially, the performance followed a continuously increasing trend. The increase seems natural since the generative model has to only fill in a lesser amount of content in the already provided natural order of keywords as we keep providing more keywords. So the task of generation for the model keeps getting easier. However, the sequential order performs markedly poorly against our approach for the initial stages before crossing over to higher scores, as shown in Figure 6. This shows the merit of our approach for clause generation based on minimal keyword information, helping the system to reach the desired clause quickly - without asking for a more significant number of keywords. It is important to appreciate the practicality aspect here since we may not expect an end user to keep providing input to the extent that the basis of the motive lying in automated clause generation gets moot.

5.3 Generation quality

We show a few examples of clauses generated from reference plans in Table 4. The generation quality observed, backed by the aggregated quantitative results showcases the clause generation model's efficacy in generating appropriate clauses from their corresponding plans.

The generated clauses succeed in conveying the same intent as the actual clause while also being strikingly similar in their lexical content. The clauses seem to capture the nuances in legal writing very well, and also change based on the topic and context of the content being generated. The generated clauses naturally fail to add any entity-specific information (such as the clause shown under the topic *brokers*) since the approach does not account for taking in these inputs from the user. Future work can explore incorporating such information from the user to generate entity-specific clauses. Considering contract-specific reference information (such as "section 2.2(b)" in the generated clause under the topic *advances* as opposed to "section 2.3(b)") can be yet another scope for future work. Many of the differences between the actual and generated clause content involved phrases implying similar intent, such as "explicitly and

Topic	advances
Plan	advance, amount, section, revolving, rate, accordance, outstanding, interest, principal, subject
Generated	Subject to section 2.2(b), the principal amount outstanding under the revolving line shall accrue interest at a floating per annum rate equal to the prime rate plus the applicable margin, which interest shall be payable monthly in accordance with the terms of this agreement.
Actual	Subject to section 2.3(b), the principal amount of advances outstanding under the revolving line shall accrue interest at a floating per annum rate equal to the prime rate, which interest shall be payable monthly in accordance with section 2.3(d) below.

Topic	brokers
Plan	broker, connection, commission, fee, agreement, finder, transaction, contemplated, brokerage, entitled
Generated	No broker, finder or investment banker is entitled to any brokerage, finders or other fee or commission in connection with the transactions contemplated by this agreement based upon arrangements made by or on behalf of buyer.
Actual	Except for Raymond James Financial, Inc. and Oppenheimer & Co. Inc., no broker, finder or investment banker is entitled to any brokerage, finder's or other fee or commission in connection with the transactions contemplated by this agreement or any other transaction agreement based upon arrangements made by or on behalf of strand.

Topic	data privacy
Plan	data, company, plan, personal, transfer, participation, administering, consent, purpose, managing
Generated	The grantee hereby explicitly and unambiguously consents to the collection, use and transfer, in electronic or other form, of his or her personal data as described in this agreement and any other award grant materials (data) by and among, as applicable, the employer, the company and any parent or subsidiary for the exclusive purpose of implementing, administering and managing the grantees participation in the plan.
Actual	The grantee hereby voluntarily consents to the collection, use and transfer, in electronic or other form, of the grantees personal data as described in this agreement and any other award grant materials by and among, as applicable, the company and any subsidiary or affiliate for the exclusive purpose of implementing, administering and managing the grantees participation in the plan.

Topic	withholdings
Plan	withholding, tax, applicable, payment, agreement, pursuant, subject, made, income, employment
Generated	All payments made pursuant to this agreement shall be subject to withholding of applicable income and employment taxes.
Actual	All payments made pursuant to this agreement will be subject to withholding of applicable income, employment and excise taxes.

Topic	limitation of liability
Plan	damage, party, consequential, indirect, punitive, incidental, notwithstanding, foregoing, entitled
Generated	The collateral trustee shall not be liable for any action taken or omitted to be taken by it hereunder or under any other secured debt document, except for its own gross negligence or willful misconduct.
Actual	The collateral trustee will not be responsible or liable for any action taken or omitted to be taken by it hereunder or under any other security document, except for its own gross negligence or willful misconduct as determined by a final order of a court of competent jurisdiction.

Table 4: Example clauses generated by the best performing BART model given a topic with the corresponding plans and actual (reference) clauses.

unambiguously" versus "voluntarily" for the example under *data privacy*. Most of the shorter length clauses showed good lexical overlap with the actual (e.g. *withholdings*) with hallucination observed in some content (e.g. *limitation of liability*).

Besides entity and contract-specific information, future work can also handle allowing phrase-level control in clauses, where the user can ask for explicit phrases to be included, or not to be included in the clause along with specification of a few custom keywords for reference. The challenge here would lie in detecting the appropriateness of position for placement of that phrase within the clause.

5.4 Controllable, iterative plan-to-clause workflow

We demonstrate the controllability in clause generation with an example flow of iterative clause cus-

tomization in Appendix B. We found the generated clauses to suitably vary content based on simple addition and removal of necessary keywords which can encourage approaches for developing efficient tools in legal clause drafting.

Consider an end user to be acquainted with drafting legal contracts - for instance lawyers. An iterative flow involving content planning followed by clause generation allows the user to keep deleting and adding keywords to the plan for driving down towards a desired state of the clause. The idea is to allow an end-user to generate a legal clause by the specification of minimal information such that the final generated state of the clause can be used with minimal edits necessary. A keyword-based information control facilitates this simplicity compared to control based on latent space representation.

An interesting problem we believe future work

could look at is ensuring only necessary phrasal changes are made between two successive stages of clause generation in the iterative pipeline shown in Figure 7, thus making the control more precise. For instance, the addition of the keyword "law" to the plan in the third stage of generation makes changes to the clause like changing "governmental authority" to "arbitrator", increasing the verbosity of the clause and a slight change of meaning w.r.t. company's shares of common stock. Explainability and more nuanced control in this process would make clause generation more precise.

6 Conclusion

We propose a plan-based approach for generating legal clauses inspired by content planning techniques in story generation. The pipeline involves customizable content plan generation based on the clause topic and optional control keywords using a simple, lightweight graph followed by clause generation. The content plan represents its corresponding clause as an ordered list of generic to specific keywords. Our approach achieves promising results for clause generation across the broad range of clause topics in the dataset, indicating the extensibility of our approach. We also show the merit of our proposed order in generating clauses with lesser keyword information. While we discuss a use case for controllability in clause generation possible through our pipeline, the generation of clause content shows substantial changes for minor changes in the plan. Future work can look at increasing the preciseness of control involved by changing only the content of a clause as necessitated by a change in the input plan. The customization of clause content can be further drilled down to inclusion of entity-specific and contract-specific information.

7 Limitations

While we evaluate the generation of clauses by using regular generation-based metrics (BLEU & ROUGE), establishing results based on human evaluations would have provided substantial qualitative backing for the empirically strong results. However, the understanding and evaluation of clauses would require strong domain knowledge in legal clauses, and any evaluation from a layperson would not help in gauging the quality. Due to the practical difficulties in involving domain-specific experts to evaluate a substantial number of clauses to make

a judgment, we relied on the quantitative metrics and some qualitative analysis performed randomly on a select set of clauses.

Controllable content generation has been popularly demonstrated by the CTRL (Keskar et al., 2019) architecture that shows fine capabilities in controlling the content based on the specification of control keywords appended before the prompt for a generation. Although it would have been interesting to study the performance of this model fine-tuned for clause generation, we were limited by sufficient computational resources to carry out the experiment on this model, and on the larger variants of models (BART, GPT-2) we currently have used.

References

Vinay Aggarwal, Aparna Garimella, Balaji Vasan Srinivasan, Anandhavelu N, and Rajiv Jain. 2021. ClauseRec: A clause recommendation framework for AI-aided contract authoring. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8770–8776, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Inf. Sci.*, 509(C):257–289.

Hong Chen, Raphael Shu, Hiroya Takamura, and Hideki Nakayama. 2021. Graphplan: Story generation by planning with event graph.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. 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*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Lefteris Loukas, Manos Fergadiotis, Ion Androutsopoulos, and Prodromos Malakasiotis. 2021. EDGAR-CORPUS: Billions of tokens make the world go round. In *Proceedings of the Third Workshop on Economics and Natural Language Processing*, pages 13–18, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572, Uppsala, Sweden. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

D. Simonson, Daniel P. Broderick, and Jonathan Herr. 2019. The extent of repetition in contract language. *Proceedings of the Natural Legal Language Processing Workshop 2019*.

Don Tuggener, Pius von Däniken, Thomas Peetz, and Mark Cieliebak. 2020. LEDGAR: A large-scale multi-label corpus for text classification of legal provisions in contracts. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1235–1241, Marseille, France. European Language Resources Association.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2018. Plan-and-write: Towards better automatic storytelling.

Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. 2020. How does nlp benefit legal system: A summary of legal artificial intelligence.

A Frequency distribution of clause topics in the dataset

Aggregated statistics of bins of clause topics based on their frequencies are given in Table 5. By the frequency of a clause topic, we mean the number of clauses under that topic.

range	# topics	mean	std.	median
100-150	309	120.5	14.2	118.0
150-250	233	192.1	29.2	186.0
250-500	230	344.3	65.5	332.0
500-1k	102	660.7	119.2	643.5
1k-5k	62	1901.9	912.2	1483.5
>5k	5	8205.2	1304.7	7853.0
Overall	939	412.4	764.5	210.0

Table 5: Distribution of clause topics in the dataset w.r.t. the number of clauses under each topic. The values of mean, median, and std indicate the corresponding values of the number of clauses under a topic in that bin range. The topic *governing laws* had the highest number of clauses at 10,636.

B Iterative planning and generation: Example use case

Figure 7 shows an example use case of the customized plan to clause generation.

Taking in user input: topic & (optional) custom keywords	
topic	severability
custom keywords	court, invalid, modified

Generation of content plan and clause based on the user input

plan	provision, invalid, remaining, invalidity, court, agreement, section, company, share, fractional, modified
clause	If any provision of this agreement shall be declared by any court or governmental authority to be invalid, such invalidity shall not invalidate the remaining provisions of the agreement, and the company shall use commercially reasonable efforts to obtain from each holder of registrable securities a certificate or certificates representing such fractional shares, if any, as may be necessary to effectuate the original intent of the company and the holder. The provisions of this section shall not be modified in a manner that would adversely affect the rights of the holders of the securities covered by this agreement.

User doesn't need the part on fractional shares, hence removes 'fractional' from the plan

plan	provision, invalid, remaining, invalidity, court, agreement, section, company, share, ~~fractional~~, modified
clause	If any provision of this agreement shall be declared by any court or governmental authority to be invalid, such invalidity shall not impair the remaining provisions of the agreement, and the agreement shall otherwise remain in full force and effect. The company shall not be required to issue any shares of common stock or other securities pursuant to this agreement unless and until such provision shall be deemed to have been modified in accordance with the provisions of this section.

User needs to explicitly mention law applicability, hence adds the 'law' keyword to the plan

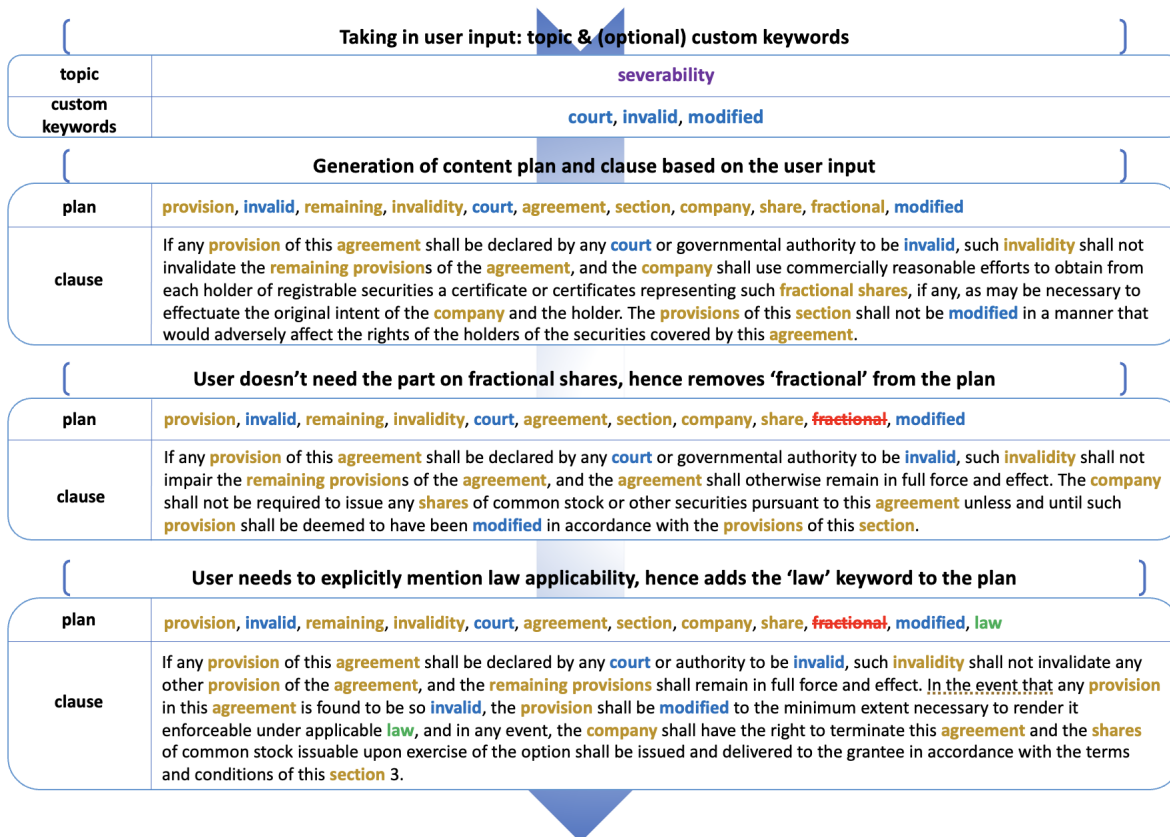|	
plan	provision, invalid, remaining, invalidity, court, agreement, section, company, share, ~~fractional~~, modified, law
clause	If any provision of this agreement shall be declared by any court or authority to be invalid, such invalidity shall not invalidate any other provision of the agreement, and the remaining provisions shall remain in full force and effect. In the event that any provision in this agreement is found to be so invalid, the provision shall be modified to the minimum extent necessary to render it enforceable under applicable law, and in any event, the company shall have the right to terminate this agreement and the shares of common stock issuable upon exercise of the option shall be issued and delivered to the grantee in accordance with the terms and conditions of this section 3.

Figure 7: Example flow where an end user generates a clause for the clause topic *severability* by specifying a few additional keywords at the start. In subsequent stages, the user removes and adds keywords from the generated plan to directly control the clause content.