Generating Reasonable Legal Text through the Combination of Language Modeling and Question Answering

Weijing Huang[1] , **Xianfeng Liao**[2*] , **Zhiqiang Xie**[2*] , **Jiang Qian**[1] , **Bojin Zhuang**[1] , **Shaojun Wang**[1] and **Jing Xiao**[3]
[1]Ping An Technology
[2]University of Science and Technology of China
[3]Ping An Insurance (Group) Company of China
huangwaleking@gmail.com, sa1314hx@mail.ustc.edu.cn, earl.xiezhiqiang@foxmail.com,
{qianjiang456, zhuangbojin232, wangshaojun851, xiaojing661}@pingan.com.cn

Abstract

Due to the improvement of Language Modeling, the emerging NLP assistant tools aiming for text generation can greatly reduce the human workload on writing documents. However, the generation of legal text faces greater challenges than ordinary texts because of its high requirement for keeping logic reasonable, which can not be guaranteed by Language Modeling right now. To generate reasonable legal documents, we propose a novel method CoLMQA, which (1) combines Language Modeling and Question Answering, (2) generates texts with slots by Language Modeling, and (3) fills the slots by our proposed Question Answering method named Transformer-based Key-Value Memory Networks. In CoLMQA, the slots represent the part of the text that needs to be highly constrained by logic, such as the name of the law and the number of the law article. And the Question Answering fills the slots in context with the help of Legal Knowledge Base to keep logic reasonable. The experiment verifies the quality of legal documents generated by CoLMQA, surpassing the documents generated by pure Language Modeling.

1 Introduction

The improvement of Language Modeling [Mikolov *et al.*, 2010; Vaswani *et al.*, 2017; Radford *et al.*, 2018; Devlin *et al.*, 2019] has greatly changed the landscape of NLP, and begin to shed lights on automatic text generation, which can reduce the human workload on writing documents. The successful examples are the smart reply [Kannan *et al.*, 2016] and the smart compose in Gmail, helping people to type the fixed daily utterances in e-mail editor.

The similar efforts have being made in the legal domain [Alschner and Skougarevskiy, 2017] to automatically generate the official legal documents, even since 1990s [Branting, 1998]. However, the generating of legal text faces greater challenges than ordinary texts due to its own characteristics. There are at least two requirements for the generation of legal documents: (1) the syntax should be correct, and (2) the logic should be reasonable. The syntax correctness problem has been mainly relieved by the current pre-trained-then-fine-tuned language models such as GPT [Radford *et al.*, 2018]. But the current language models still lack the ability to keep logical rationality [Mao *et al.*, 2019; Guan *et al.*, 2020]. For example, the text may be generated by a language model such as "Someone misappropriated 100,000 yuan of public funds. In accordance with Article 100 of the Criminal Law of the People's Republic of China[1], the judgment is as follows ...", which is syntax-correct. But the language model here does not know what Article 100 is, even it has been trained on a lot of judicial documents. In fact, Article 100 in the Criminal Law of the People's Republic of China is about the crime reporting system, which logically conflicts with the fact of the misappropriation of public funds.

In another hand, the template-based generation seems a possible solution, but it needs large manual efforts for creating templates [Branting, 1998], and cannot be generalized to a more complex scenario. Meanwhile, a method for generating variational templates is proposed [Ye *et al.*, 2020] for further generating text from tables, but can not be easily extended to the legal domain. However, the template-based method meets another challenge that requires the knowledge of implementing the templates to avoid logic conflicts.

To solve this problem, we introduce the Knowledge Base enhanced Question Answering technique to the existing language modeling in order to generate the logical part. In our pilot study, when generating the "Article 100", the language model was very uncertain whether it was correct. In general, where the language model is uncertain, you can ask the system: what applicable law article should be applied to a certain person who misappropriates 100,000 yuan of public funds? The Question Answering component can look up the knowledge base and historical precedents, knowing that it is Article 185 and Article 272 in the Criminal Law of the People's Republic of China, which is further discussed in Subsection 4.3 and Figure 3. We use the slots to represent the logical part

*Contribution during internship at Ping An Technology.

[1]The English version of the Criminal Law of the People's Republic of China can be found on an official webpage https://www.fmprc.gov.cn/ce/cgvienna/eng/dbtyw/jdwt/crimelaw/t209043.htm.

need to be addressed in a QA system. And the slots can be generated as special tokens in a text sequence by a trained language model.

To implement this idea, we propose a novel method CoLMQA, which (1) combines Language Modeling and Question Answering, (2) generates texts with slots by Language Modeling, and (3) fills the slots by our proposed Question Answering method named Transformer-based Key-Value Memory Networks.

To sum up, our contribution is mainly in four aspects. (1) We convert the problem of keeping logic in the legal text generation into two sub-tasks of generating texts with slots and filling slots with the logical coherent values. (2) With the guiding of this divide-and-conquer idea, we propose a method CoLMQA, which benefits from the Language Modeling and Question Answering simultaneously. (3) We propose a Transformer-based Key-Value Memory Networks, which can encode a long query. (4) The experiment verifies that CoLMQA can generate fluent sentences with slots, and can fill in the correct values to keep the overall logical coherent.

2 Related Works

On methodology, there are two orthogonal lines of research related to our work: language modeling and question answering. And the related works also include the NLP techniques applied to legal documents.

Language modeling. The language models are used to predict the words when given context, describing the statistical pattern in a sequence, such as the n-gram in traditional statistical linguistics [Bellegarda, 2004]. With the development of deep learning, neural language models like LSTM [Hochreiter and Schmidhuber, 1997] and RNNLM [Mikolov *et al.*, 2010] enhance the prediction power. Furthermore, the Transformer-based neural language models [Vaswani *et al.*, 2017; Radford *et al.*, 2018; Devlin *et al.*, 2019] have greatly changed the landscape of NLP.

Both traditional statistical language models and neural language models are trained on text corpora to memorize patterns. Although Petroni [2019] and Bouraoui [2020] point out that the masked language model BERT [Devlin *et al.*, 2019] can learn certain types of factual knowledge from large text corpora, BERT cannot be used on the text generation task directly. On the other hand, the autoregressive language model GPT [Radford *et al.*, 2018], GPT-2 [Radford *et al.*, 2019] can generate syntax-fluent text, and see successful applications.

But GPT(-2) faces the problem of unabeling to generate factual aware text [Logan *et al.*, 2019; Mao *et al.*, 2019; Guan *et al.*, 2020]. To generate reasonable stories, Mao [2019] and Guan [2020] both independently conducts the GPT-2's multi-task fine-tuning on external common sense datasets (e.g., ConceptNet) to promote GPT-2's awareness of facts. Unlike generating reasonable stories, our task places additional emphasis on the preciseness of law article numbers to keep logic in the legal texts. Just like cardinal numbers in natural language need to be processed additionaly [Andor *et al.*, 2019], so are ordinal numbers in legal documents.

Question answering. Question Answering is a big family of NLP tasks, including QA on Knowledge Base [Berant *et al.*, 2013], machine reading comprehension [Rajpurkar *et al.*, 2016], cloze-style QA [Das *et al.*, 2017], etc. In a nutshell, it provides answers to natural language questions. Our task of filling correct values in generated legal text slots is equivalent to the cloze-style question answering.

Adopting an external memory module, the memory networks [Weston *et al.*, 2015; Sukhbaatar *et al.*, 2015; Bordes *et al.*, 2015] perform well on QA for mainly two reasons: (1) the addressing and reading on external memories help the multi-hop reasoning; (2) the memory module can visit an external knowledge base and scale up to complex reasoning. Key-value memory networks [Miller *et al.*, 2016] extends MemNNs by separating keys and values in memory module, making it suitable for reading the (key, value) style external knowledge [Das *et al.*, 2017; Xu *et al.*, 2019]. The aforementioned memory networks exploit bag of words or RNN to encode queries, keys, and values but cannot encode very long texts well in our slots filling scenario. The closest work to ours is Generative Transformer Memory Network [Dinan *et al.*, 2019], which employs Transformer as encoder and decoder in multi-turn dialogue. In our work, the answer should be selected rather than generated to ensure preciseness.

Legal documents. It's a big human workload to process massive legal texts. So employing NLP methods in legal domain has attracted a lot of attention recently, for instance, charge prediction [Luo *et al.*, 2017; Hu *et al.*, 2018; Chen *et al.*, 2019], question answering [Zhong *et al.*, 2020], and applicable law articles prediction [Zhong *et al.*, 2018]. And the generation of legal texts also draws interests in the NLP community. Alschner [2017] proposed a modified RNN and applied on bilateral investment treaties. Ye [2018] provided a seq2seq method to generate court views (written explanation from judges) from criminal facts.

The legal documents can be roughly categorized as follows: legislative, executive, judicial documents, and contracts [Gostojić and Marković, 2019]. The aforementioned methods mainly focus on only one type of documents. However, the contents of the law articles are in the legislative documents, while the numbers of the law articles are mentioned in the judicial documents. Therefore, combining them helps to generate reasonable legal documents.

3 Problem Definition

We consider the problem of generating a reasonable legal document d when given a prompt π and a legal knowledge base $\mathcal{K}$. More specifically, we narrow down the type of legal document d to judicial document, so the prompt π is a text sequence for describing the meta information of the judicial document, such as the criminal fact. A legal knowledge base is formalized as $\mathcal{K} = \{(k_i, v_i)\}_{i=1}^{|\mathcal{K}|}$, where k_i is the i-th law article and v_i is the corresponding content. Then the task of generating reasonable legal text is to find out the best document d under the constraint between π and $\mathcal{K}$, shown as Equation (1).

$$d = \arg\max_{d'} p(d'|\pi, \mathcal{K}). \tag{1}$$

It's infeasible to get the global optimal solution in Equation (1) because of the huge search space $O(|\mathcal{V}|^N)$, where $|\mathcal{V}|$ is the vocabulary size and N is the upper bound of document length. To get approximation solutions, we decompose the original problem with the chain rule as follows: $p(d'|\pi, \mathcal{K}) = \prod_{i=1}^{n} p(s_i|\pi, \mathcal{K}, s_{1:i-1})$, where $d' = s_{1:n}$ and s_i is the i-th sentence in document d'. We conduct the greedy search on the decomposed parts to get the final document, which means searching a feasible solution for sentence s_i when given the previous ones $s_{1:i-1}$ at each step i. With the approximation, the original problem is converted to the following one.

$$s_i^* = \arg\max_{s_i} p(s_i|\pi, \mathcal{K}, s_{1:i-1}) \tag{2}$$

Although the original optimization problem is relaxed, the new one is still challenging because the sentence s_i relates to the prompt π, previous sentences $s_{1:i-1}$ and a legal knowledge base $\mathcal{K}$ simultaneously. We will show how to tackle the problem proposed in Equation (2) through the combination of Language Modeling and Question Answering in Section 4.

4 Method

In this section we discuss the main components of our proposed method CoLMQA.

4.1 The Overall Architecture of CoLMQA

Before diving into the details of CoLMQA, recall the "Keep Calm and Carry On" meme, which's a typical phrase often being imitated, by keeping the slots but replacing values with others, such as "Keep Calm and Never Give Up". Our method CoLMQA follows a similar style: generate texts with slots and fill them on. In this example, "*Carry On*" and "*Never Give Up*" are two different values of the slots in the meme.

More specifically, we divide the challenging task defined in Section 3 into two sub-tasks: (a) text generation with slots and (b) automatic filling slots based on the text context and knowledge base. Therefore, the optimization target in Equation (2) can be further decomposed as Equation (3), where $s_i^{(-)}$ is the sentence by keeping slots with placeholder, and $v_i^{(+)}$ are the values of slots in sentence $s_i^{(-)}$ need to be filled in. Then the optimization of Equation (2) is divided into two sub-tasks of optimization.

$$\begin{aligned} p(s_i|\pi, \mathcal{K}, s_{1:i-1}) = p(s_i^{(-)}|\pi, \mathcal{K}, s_{1:i-1}) \\ \cdot p(v_i^{(+)}|\pi, \mathcal{K}, s_{1:i-1}, s_i^{(-)}) \end{aligned} \tag{3}$$

Illustrated in Figure 1, CoLMQA contains three parts: the controller, the language modeling, and the question answering. The controller runs the text generation with the following rules, and get the two sub-tasks in Equation (3) optimized separately.

Rule 1 (LM)**.** *If there's no slot in existing text, call LM to generate next sentence.*

Rule 2 (QA)**.** *If there are slots in existing text, call QA to fill slots.*

Rule 3 (End)**.** *If there's an END symbol in existing text, finish the generation.*

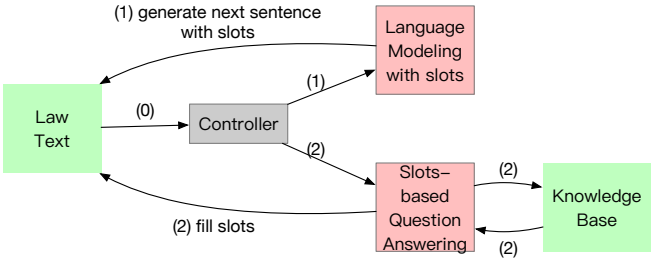

Figure 1: Overall Architecture of CoLMQA.

4.2 Generating Texts with Slots by Language Modeling

As shown in Figure 2, there are two phases in using a language model to generate texts with slots. The first phase is fine-tuning. In this phase, after replacing the corresponding values in the original text with slots, we put the legal documents into a pre-trained model for fine-tuning, where different slots occupy different positions in the vocabulary.

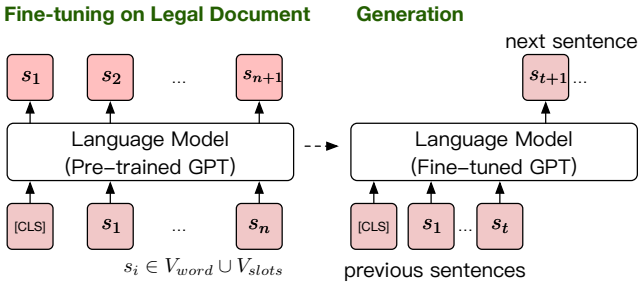

Figure 2: Generation of texts with slots.

The second phase is to use the fine-tuned language model to generate texts with slots. We choose GPT as the language model in our work. So V_{words} is the original vocabulary used by the pre-trained GPT, and V_{slots} includes the additional placeholders in the fine-tuned GPT.

4.3 Filling Slots by Transformer-based Key-value Memory Networks

Briefly, our proposed method is in a family of Key-Value Memory Networks [Miller *et al.*, 2016; Das *et al.*, 2017; Xu *et al.*, 2019], but enhanced by Transformer encoder. As Xu [2019] pointed out, the standard Key-Value Memory Networks contain the following components: key hashing, key addressing, value reading, query updating, and answer prediction, which also applies in our proposed method. And in the above components, the standard KV-MemNNs exploit the bag of words to encode queries, keys, values, and candidate answers. This straightforward approach is not appropriate to encode very long queries key-value pairs stored in the Legal Knowledge Base, as shown in a running example in Figure 4.

Encoders. Transformer has shown its ability to capture the long-range dependencies in natural language in GPT [Radford *et al.*, 2018] and BERT [Devlin *et al.*, 2019]. So rather

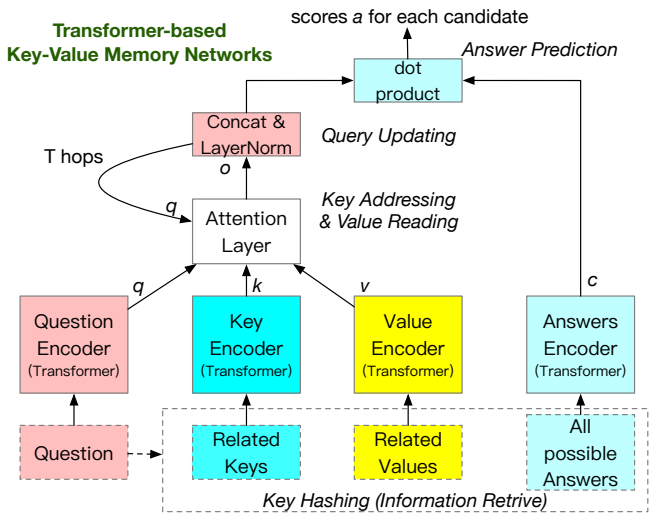

Figure 3: The illustration of the Transformer-based key-value memory networks, a KB enhanced QA method for filling slots.

Query: ... In a company, someone misappropriated 100,000 yuan of public funds, according to Article [ARTICLE_NUMBER] of the [LAW] ...	
Answer: [ARTICLE_NUMBER] = 272, [LAW] = Criminal Law of the People's Republic of China	
Related Keys & Values in Legal Knowledge Base	
Key(1): Criminal Law of the People's Republic of China, Article 185	Value(1): Personnel of banks or other monetary institutions who take advantage of their office to misappropriate funds of their respective institutions or customers are to be sentenced and punished in accordance with the stipulations of Article 272 of this law...
Key (2): Value (1)	Value (2): Key (1)
Key(3): Criminal Law of the People's Republic of China, Article 272	Value(3): When personnel of companies, enterprises, and other units, who take advantage of their offices to misappropriate their units' funds for their own use or for lending to others, and the amounts involved are relatively large and have not been returned for a period of over three months; ...
Key (4): Value (3)	Value (4): Key (3)
etc.	
Candidate Answers: Criminal Law of the People's Republic of China, Article 185; Criminal Law of the People's Republic of China, Article 272; etc.	

Figure 4: A running example of filling slots with KB enhanced QA. The slots of [ARTICLE_NUMBER], [LAW], and their context are treated as the query. The values of slots are the answer. In this example, the fact of misappropriating in a company leads to Article 272 as the answer instead of Article 185, because the latter is about the misappropriating in banks or other monetary institutions.

than encoding the input as a bag of words, we use Transformer. We append a special token, i.e., [CLS], at the tail of a sentence s, then input it into an encoder with n layer of Transformer encoder blocks. And then we use [CLS]'s corresponding output embedding $enc_{\text{[CLS]}}$ to represent the sentence s, and denote $Transformer(s) = enc_{\text{[CLS] in } s}$.

Since the Transformer encoder are pre-trained on all queries, keys, values, and candidate answers with an autoregressive task of predicting next words, the encoding of [CLS] can not be directly used in the Question Answering task. So we apply a linear projection $W_{\text{LHS}} \in \mathbb{R}^{H \times H}$ on $Transformer(s)$, where H is the hidden size of Transformer encoder.

Equation (4) defines the Question Encoder in Figure 3, given the original query q as the input, to get the representation $q^{(0)}$ as the output. And all encoders in Figure 3 use a same Transformer.

$$q^{(0)} = W_{\text{LHS}} \cdot \text{Transformer}(q) \tag{4}$$

Key hashing. This component searches on the whole Legal Knowledge Base and gets query-related key-value pairs by conventional IR methods, e.g., TF-IDF or SQL selection on specific fields. Because the IR method doesn't go deeper into the sentence-level semantics of the query, and key-value pairs, we need to store them in memory and do re-organization by the remaining components.

Key addressing and value reading. Key addressing reflects the relatedness between keys and query, as shown in Equation (5) and Equation (6). Similar to the scaled dot-product attention in Transformer, we rescaled the relatedness score $p^{(t)}$ with a factor of $1/\sqrt{H}$ when doing normalization. The superscript t indicates the t-th hop (iteration) in multi-hop reasoning, and no more than the number T of total hops.

$$p_i^{(t)} = (q^{(t)})^\top (W_{\text{LHS}} \cdot \text{Transformer}(k_i)) \tag{5}$$

$$\tilde{p}_i^{(t)} = \frac{\exp\left(p_i^{(t)}/\sqrt{H}\right)}{\sum_j \exp\left(p_j^{(t)}/\sqrt{H}\right)}, \text{ for all } i \tag{6}$$

Value reading is given by Equation (7), summing the weighted encodings of values.

$$o^{(t)} = \sum_i \tilde{p}_i^{(t)} \cdot W_{\text{LHS}} \cdot \text{Transformer}(v_i) \tag{7}$$

Essentially, key addressing and value reading implements an attention layer to select which values are used to answer the query $q^{(t)}$.

Query updating. In the case of multi-hop reasoning, the query needs to be updated after each hop of reading on KB, because the newly read-in values should also be treated as a part of the query. In the running example in Figure 4, the query is to ask which law and article number applies to the fact of misappropriating in a company. After one hop, the fourth value in Figure 4 could answer the question. But in a more complex situation, such as the misappropriating in a bank, one hop is not enough because Article 185 also refers to Article 272.

We concatenate the query representation $q^{(t)}$ and the representation of related values $o^{(t)}$, then map to the H dimension space by the matrix $M^{(t)} \in \mathbb{R}^{H \times 2H}$ to be a new query $q^{(t+1)}$.

$$q^{(t+1)} = \text{LayerNorm}(M^{(t)} \cdot (q^{(t)} \oplus o^{(t)})) \tag{8}$$

Answer prediction. Rather than picking up all keys in Legal Knowledge Base $\mathcal{K}$ as candidate answers like TextKBQA [Das *et al.*, 2017], we only conduct the similarity computation between $q^{(T)}$ and the retrieved candidates by Equations (9) and (10).

$$a_i^{(T)} = (q^{(T)})^\top \cdot W_{\text{RHS}} \cdot \text{Transformer}(c_i), \text{ for i-th candidate} \quad (9)$$

$$\hat{y}^{(T)} = \text{softmax}(a^{(T)}) \quad (10)$$

Considering the article number and law in $\mathcal{K}$ plays three roles in the network, such as the keys, the values, and the answers, we use a different matrix W_{RHS} to do the necessary decoupling in Equation (9). Finally, the loss of cross-entropy between the prediction score $\hat{y}^{(T)}$ and the ground truth y is taken for training the Transformer-based key-value memory networks.

$$Loss(\hat{y}^{(T)}, y) = -\sum_i y_i \log \hat{y}_i^{(T)} \quad (11)$$

5 Empirical Study

In this section, we demonstrate the effectiveness of our proposed method CoLMQA in the following aspects. (1) We evaluate the quality of the generated texts with slots to show the language model is capable of predicting the number of slots and the composition of slots and words, which is critical to our divide-and-conquer strategy. (2) Comparing to other QA methods, we test the accuracy of filling slots by our proposed Transformer-based key-value memory networks.

5.1 Legal Document Dataset and Legal Knowledge Base

Legal Document Dataset. All the experiments are conducted on the judicial documents from China Judgements Online[2], which are widely used in NLP works in legal domain [Luo *et al.*, 2017; Ye *et al.*, 2018]. We crawled 11,327,945 judicial documents from China Judgements Online in 2015. The legal documents cover diverse branches of law. For instance, we tokenized the document titles with a Chinese Tokenizer jieba[3], and do the quick overview of the wordcount, then find out that the word *"dispute"* appearing in 4,774,968 (42.1%) document titles, the word *"limited company"* 30.5%, the word *"criminal"* 17.2%, and the word *"divorce"* 8.5%, etc.

We randomly select 20,000 documents from the dataset, splitting to the training data, validation data, and testing data, with the portion 80%, 10%, and 10%. By regular expressions, the fact descriptions, and court views are extracted. We mainly consider the generation of court views, which is the explanation for the charge. And court views contain the applicable law article to make the charge interpretable.

Legal Knowledge Base. We collect the required legislative documents according to the law articles mentioned in the legal document dataset, e.g., the Criminal Law of the People's Republic of China. On the basis of the collected legislative documents, the regular expression and a finite-state machine are used to extract the (key, value) pairs, where the key represents law article and value represents the detailed content. In the end, the Legal Knowledge Base is built upon the 149 laws, civil codes, and interpretations, which appeared in the aforementioned Legal Document Dataset.

[2]http://wenshu.court.gov.cn/

[3]https://github.com/fxsjy/jieba

	without slots	with Type A slots	with Type B slots	with all slots
RNN	0.131	0.198	0.186	0.292
GPT	0.238	0.441	0.425	0.573
GPT (pre-trained)	0.421	0.642	0.573	0.682

Table 1: The F score of predicting law article slots (Type B slots) of different language models, on test dataset.

	without slots	with type A slots	with type B slots	with all slots
RNN	67.9	59.6	62.8	53.1
GPT	12.8	10.3	11.9	8.2
GPT (pre-trained)	11.3	8.8	10.5	7.8

Table 2: The perplexity on test dataset.

5.2 LM with Slots

Slots Based on the level of uncertainty, we modeled 7 slots: [NAME], [DATE], [MONEY], [ADDRESS], [NUMBER], [LAW], [ARTICLE_NUMBER]. We use the regular expression to replace the values in court views part in legal documents with the above slot placeholders. The first five slots reduce the uncertainty of language modeling. The last two slots correspond to two parts of a law article, which are the law's name and the number pointing to the law article. They have specific values in our built Legal Knowledge Base. These values can be obtained by reasoning based on the text content. Therefore, they are within the consideration of slot filling in our question-answering model. And we denote the first five slots as the Type A slots and the last two slots as the Type B slots. The Type A slots used here is to demonstrate that language models can benefit from reducing the uncertainties when generating text, especially when they don't have to predict the exact number of [MONEY] or the value of [DATE].

Language models with(out) slots. Language Models are categorized into different groups depending on whether trained with slots or without slots as shown in Table 1. We use two versions of GPT[4], pre-trained, and without pre-trained. The former is pre-trained on a news dataset[5] which contains 2.5 million news articles for one epoch on a 4× NVIDIA Tesla V100 machine, and with the stride being 768. The sequence length of the Transformer block used in both GPT versions is 1024. The text generation sub-task is given a prompt (the facts part in a judicial document), then to generate a sentence by using the Beam search.

[4]https://github.com/huggingface/transformers

[5]It's news2016zh dataset in https://github.com/brightmart/nlp_chinese_corpus.

Result. In the first sub-task, after removing some highly uncertain content and replacing it with slots, the language model can generate more deterministic text and make the sentence more fluent as shown in Table 1 and Table 2.

Given a prompt, we use a language model to generate the court view that contains slots and then count the generated Type B slots in it. The F score in Table 1 is computed between two Type B slot sequences by matching one by one. As a toy example, a ground truth in a court view containing slots like *"in accordance with Article b and Article b of a"*, where a=[LAW], b=[ARTICLE_NUMBER], and the slot sequences in a generated text is like *"in accordance with Article b of a, Article b of a, and Article b of a"*. Then the precision of predicting Type B slots ([LAW] and [ARTICLE_NUMBER]) is 1/6 because the first one slots are matched one by one, while the recall of predicting Type B slots is 2/3 when considering the total length of subsequences *"b*a"* in the ground truth is matched by the generated sequence. The F score in this toy example is 0.27. In this way, the F score is used to measure how good the text sequence with slots is matched to the ground truth. When computing the F score of the slots prediction by using the model without slots, we do a manual conversion to replace the values with slots in the result.

In Table 1, we compare different language models and different settings and find out that the pre-trained GPT can effectively generate a sentence with correct slots. In Table 2, the perplexity on the test dataset demonstrates that the generated sentence of the court view is fluent when considering slots.

5.3 Filling Slots by Transformer-based Key-Value Memory Networks

Construction of query-answer pairs. To derive the query-answer training pairs from the Legal Document Dataset in Subsection 5.1, we consider different situations as follows.

The first one is that a court view part in the legal judicial document only contains one applicable law article in the form of "According to Article 263 of the Criminal Law of ...". In this case, the query in a query-answer pair is the text span from the beginning to the position of current law article, while the value is the ground truth, e.g., Article 263 of the Criminal Law. The second one is that a court view contains more than one applicable law article with the same law, such as "According to Article 263 and Article 269 of the Criminal Law". In this one, we make multi tiny-modified copies, Query1: "According to [ARTICLE_NUMBER] of the [LAW]", Query2: "According to Article 263 and [ARTICLE_NUMBER] of the Criminal Law". The third one is the mixing case, that different laws and different law articles are cited in a court view, which can be processed like the second one. On average, there are 4.1 query-answer pairs produced per court view. And the average length of a query is 2783. For the query which is longer than the sequence length of Transformer Encoder in TKVMemNN, we keep the tail part of the query sequence, which is reasonable as the fact description appears more often in the tail part.

Training of our method. We use the first four layers of Transformers in our pre-trained GPT as the encoder for queries, keys, values, and answers in the experiment, then carry out the two-phases fine-tuning. The first-phase fine-tuning is conducted on the task of predicting the next words in the sequences of queries, keys, and values. The second-phase fine-tuning is training the proposed model TKVMemNN on our constructed query-answer pairs and the Legal Knowledge Base. To conduct the multi-hop reasoning, we set the number of hops T as 3.

Baselines. We use Whoosh (a Python IR open-source toolkit), Memory Networks, and their variants with or without the document title as baselines. As the title are important for providing additional information, such as the type of documents, they are useful for filling slots.

Because the computation of the dot product between the query and the encodings of all the candidates in the Legal Knowledge Base is very expensive, we limit the number of candidates in IR scope to improve the efficiency.

MemNN	Whoosh	Whoosh +title	TKVMemNN	TKVMemNN +title
0.23	0.13	0.17	0.39	0.41

Table 3: The accuracy of filling slots.

Result. In the second sub-task, the multi-hop reasoning on the knowledge base makes the generated text more logical as illustrated in Table 3.

6 Conclusion and Future Works

In order to solve the problem of conflicting logics appeared in the text generated by a language model, we propose a novel method CoLMQA which benefits from Language Modeling and Question Answering simultaneously. In the Language Modeling component, the sentences with slots are generated and provided a well-written skeleton of sentences. And in the Question Answering component, the slots are filled with accurate values with the help of Knowledge Base. In our experiment, we verified that our method can fill slots with logical coherent values. In the future, we'll expand CoLMQA to other scenarios of text generation which also require keeping logic, but using different kinds of slots.

Acknowledgments

We thank Xi Chen, Ni Li, Yan Cui, Mingkuo Ji, Jun Chen, Yuren Wu, and Hanzhang Yang for helpful discussions.

References

[Alschner and Skougarevskiy, 2017] Wolfgang Alschner and Dmitriy Skougarevskiy. Towards an automated production of legal texts using recurrent neural networks. In *ICAIL*, pages 229–232. ACM, 2017.

[Andor *et al.*, 2019] Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. Giving BERT a calculator: Finding operations and arguments with reading comprehension. In *EMNLP*, 2019.

[Bellegarda, 2004] Jerome R Bellegarda. Statistical language model adaptation: review and perspectives. *Speech communication*, 42(1):93–108, 2004.

[Berant *et al.*, 2013] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*, 2013.

[Bordes *et al.*, 2015] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[Bouraoui *et al.*, 2020] Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. Inducing relational knowledge from bert. *AAAI*, 2020.

[Branting, 1998] L Karl Branting. Techniques for automated drafting of judicial documents. *IJLIT*, 6(2):214–229, 1998.

[Chen *et al.*, 2019] Huajie Chen, Deng Cai, Wei Dai, Zehui Dai, and Yadong Ding. Charge-based prison term prediction with deep gating network. *EMNLP*, 2019.

[Das *et al.*, 2017] Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. Question answering on knowledge bases and text using universal schema and memory networks. In *ACL*, 2017.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, 2019.

[Dinan *et al.*, 2019] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. In *ICLR*, 2019.

[Gostojić and Marković, 2019] Stevan Gostojić and Marko Marković. Legal document management: An integrated approach. In *Sinteza 2019*, pages 374–380, 2019.

[Guan *et al.*, 2020] Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minglie Huang. A knowledge-enhanced pretraining model for commonsense story generation. TACL, 2020.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2018] Zikun Hu, Xiang Li, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. Few-shot charge prediction with discriminative legal attributes. In *COLING*, 2018.

[Kannan *et al.*, 2016] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, et al. Smart reply: Automated response suggestion for email. In *KDD*, pages 955–964, 2016.

[Logan *et al.*, 2019] Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*, 2019.

[Luo *et al.*, 2017] Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang, and Dongyan Zhao. Learning to predict charges for criminal cases with legal basis. *arXiv preprint arXiv:1707.09168*, 2017.

[Mao *et al.*, 2019] Huanru Henry Mao, Bodhisattwa Prasad Majumder, Julian McAuley, and Garrison W Cottrell. Improving neural story generation by targeted common sense grounding. *EMNLP*, 2019.

[Mikolov *et al.*, 2010] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

[Miller *et al.*, 2016] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *EMNLP*, 2016.

[Petroni *et al.*, 2019] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *EMNLP*, 2019.

[Radford *et al.*, 2018] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[Sukhbaatar *et al.*, 2015] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, pages 2440–2448, 2015.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[Weston *et al.*, 2015] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *ICLR*, 2015.

[Xu *et al.*, 2019] Kun Xu, Yuxuan Lai, Yansong Feng, and Zhiguo Wang. Enhancing key-value memory neural networks for knowledge based question answering. In *ACL*, pages 2937–2947, 2019.

[Ye *et al.*, 2018] Hai Ye, Xin Jiang, Zhunchen Luo, and Wenhan Chao. Interpretable charge predictions for criminal cases: Learning to generate court views from fact descriptions. In *NAACL*, pages 1854–1864, 2018.

[Ye *et al.*, 2020] Rong Ye, Wenxian Shi, Hao Zhou, and Lei Li. Variational template machine for data-to-text generation. In *ICLR*, 2020.

[Zhong *et al.*, 2018] Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. Legal judgment prediction via topological learning. In *EMNLP*, pages 3540–3549, 2018.

[Zhong *et al.*, 2020] Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. Jec-qa: A legal-domain question answering dataset. *AAAI*, 2020.