Classify First, and Then Extract: Prompt Chaining Technique for Information Extraction

Alice Saebom Kwak[1], **Clayton T. Morrison**[2],
Derek E. Bambauer[3], **Mihai Surdeanu**[4]

[1]Department of Linguistics, The University of Arizona
[2]College of Information Science, The University of Arizona
[3]Levin College of Law, The University of Florida
[4]Department of Computer Science, The University of Arizona
{[1]alicekwak, [2]claytonm, [4]msurdeanu}@arizona.edu
[3]bambauer@law.ufl.edu

Abstract

This work presents a new task-aware prompt design and example retrieval approach for information extraction (IE) using a prompt chaining technique. Our approach divides IE tasks into two steps: (1) text classification to understand what information (e.g., entity or event types) is contained in the underlying text and (2) information extraction for the identified types. Initially, we use a large language model (LLM) in a few-shot setting to classify the contained information. The classification output is used to select the relevant prompt and retrieve the examples relevant to the input text. Finally, we ask a LLM to do the information extraction with the generated prompt. By evaluating our approach on legal IE tasks with two different LLMs, we demonstrate that the prompt chaining technique improves the LLM's overall performance in a few-shot setting when compared to the baseline in which examples from all possible classes are included in the prompt. Our approach can be used in a low-resource setting as it does not require a large amount of training data. Also, it can be easily adapted to many different IE tasks by simply adjusting the prompts. Lastly, it provides a cost benefit by reducing the number of tokens in the prompt.

1 Introduction

This work introduces a new prompt chaining technique for information extraction (IE) in the in-context learning (ICL) setting. Since the large language model (LLM)'s capability of handling various tasks in a few-shot setting has been demonstrated (Brown et al., 2020), many researchers have investigated using LLMs in the ICL setting.

A key challenge in this research area is example retrieval. Retrieving *good* examples for the prompt improves the performance of LLMs in the ICL setting (Gao et al., 2021; Liu et al., 2022). Different approaches have been made to retrieve good examples, but they rely mostly on semantic similarity with the underlying text. However, semantic similarity-based example retrieval does not guarantee good example quality. As Wan et al. (2023) indicates, there are cases where task-aware example retrieval works better. For example, when working on a IE task from a domain-specific document in which many sentences share high semantic similarity yet contain different types of information, retrieving examples based on the information *type* contained in each sentence is a better option than using a semantic similarity-based approach. There are a few works that present task-aware example retrieval techniques (Wan et al., 2023; Huang et al., 2023). However, the techniques are not easily adaptable because they require training or fine-tuning a model. They also focus on addressing specific tasks rather than general IE tasks.

Our approach with the prompt chaining technique provides an alternative to these methods as it does not involve any training nor fine-tuning. Also, it can be easily adapted to various IE tasks by simply adjusting prompts. The main idea of our approach is to split the IE tasks into two steps: (1) text classification and (2) information extraction. In the text classification step, an input text is classified based on the information contained in it. We prompt a LLM to do the text classification in a few-shot setting. The output from this step is used to retrieve examples of the relevant type(s) that are relevant to the input text. With the retrieved examples, the prompt for the information extraction is generated. Lastly, we ask a LLM to do the information extraction using the generated prompt.

The main contributions of this work are:

- This work introduces a new task-aware example retrieval technique using prompt chaining. This approach does not require any model training nor fine-tuning. It can be applied in the low-resource setting as it does not require training data. Also, this approach can be easily adapted to many IE tasks by simply adjusting the prompts.

- We demonstrate that the prompt chaining technique improves LLM's performance on the IE tasks in a few-shot setting when compared to the baseline model in which examples from all possible classes are included in the prompt. GPT-4's results show that in the in-domain dataset, the prompt chaining approach improves the F1 score by 3.41 percentage points for entity extraction (76.40% vs. 72.99%) and 3.68 percentage points for event extraction (67.02% vs. 63.34%) compared to the baseline. In the out-of-domain dataset, it also outperforms the baseline for entity extraction (59.76% vs. 55.93%) and event extraction (43.55% vs. 37.24%). GPT-4o mini shows a similar trend, with the prompt chaining boosting entity extraction by 1.52 percentage points (77.05% vs. 75.53%) and event extraction by 2.37 percentage points (70.67% vs. 68.30%) in-domain. Out-of-domain, it improves entity extraction by 0.71 percentage points (58.82% vs. 58.11%) and event extraction by 7.09 percentage points (42.13% vs. 35.04%).

- Employing the technique provides cost benefits by reducing the number of tokens contained in a prompt. In our evaluation, the prompt chaining approach is 6.99 times cheaper in input processing compared to the baseline model.

2 Related Work

2.1 Prompt Engineering Focusing on Example Retrieval

Recently there has been considerable research on prompt engineering techniques, focused particulary on example retrieval. Earlier works focus on retrieving examples that are semantically similar to the query. Gao et al. (2021) and Liu et al. (2022) use a k-nearest neighbors (NN) algorithm to retrieve examples that are semantically similar to the query.

More recent works train example retrievers to find examples with higher relevance to the input query. Rubin et al. (2022), Luo et al. (2023) and Li et al. (2023b) train dense retriever using the LLM's training signal. Wang et al. (2024) presents a framework which can be used to train dense retrievers iteratively by employing a reward model trained on the LLM's training signal.

Another approach emphasizes the inclusion of a wide range of examples, rather than just those that are semantically similar or relevant to the query. Ye et al. (2023) and Polat et al. use a Maximal Marginal Relevance-based approach to select examples that are not only relevant to the given query, but also complementary to each other. Mo et al. (2024) uses k-NN algorithm and a self-consistency retrieval strategy to include both correct/semantically similar examples and wrong/negative examples in the prompt. He et al. (2023) and Guo et al. (2024) focus on constructing diverse demonstrations to handle document information extraction and unified information extraction, respectively.

There are a few works that concentrate on task-aware retrieval. Wan et al. (2023) proposes two task-aware retrieval methods for relation extraction tasks: (1) entity-prompted sentence embedding and (2) fine-tuned relation representations. Huang et al. (2023) presents a API Entity-Relation Joint Extraction framework, which consists of a dynamic prompt generator and a joint entity-relation extractor. The work employs a BERT-based classifier to identify the top-3 candidate relations from an input text, generating a prompt that includes only examples relevant to these candidate relations.

Our work also concentrates on task-aware retrieval, but it differs from previous efforts in two aspects. First, our work does not involve any model training or fine-tuning. Both Wan et al. (2023) and Huang et al. (2023) require training or fine-tuning a model. Our approach can be applied in a low-resource setting as it does not require a large amount of training data. Second, our approach can be adapted to various types of IE tasks. If the prompt is adjusted properly, our approach can handle a variety of IE tasks ranging from entity extraction to complex event extraction. In contrast, Wan et al. (2023) and Huang et al. (2023) focus on addressing specific tasks (relation extraction and API entity and relation extraction).

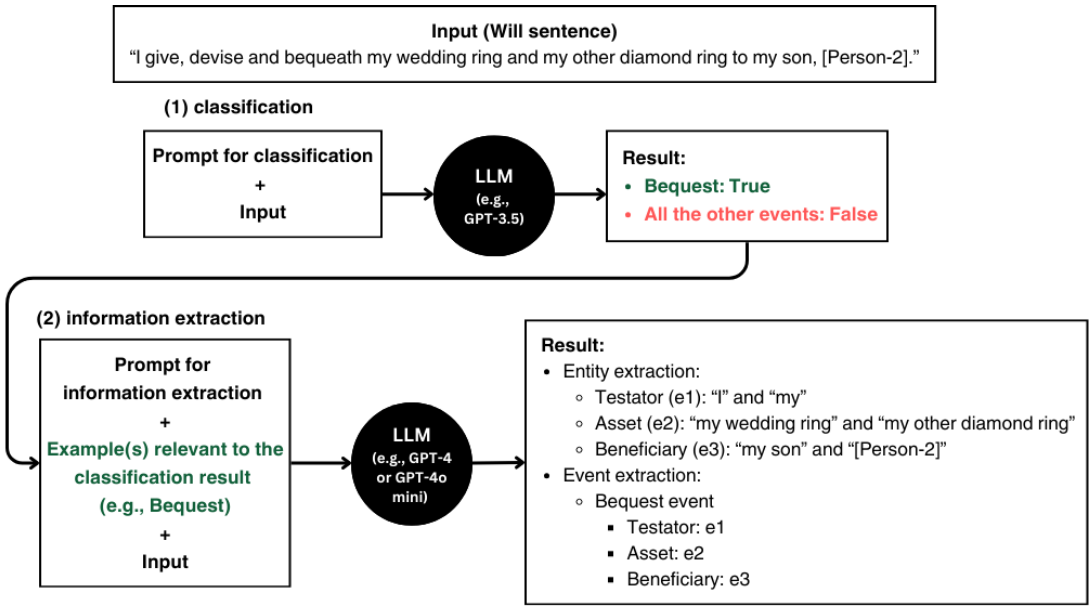

Figure 1: This figure shows an overview of our prompt chaining technique for information extraction. When an input text is given, it is first classified based on the information contained in it to understand which entities and events are likely to be present. We prompt an LLM (e.g., GPT-3.5 in this work) to do the text classification in a few-shot setting. Based on the text classification result, the examples for inclusion in the information extraction prompt are chosen. With the prompt completed with the chosen examples, we ask a second LLM (e.g., GPT-4 or GPT-4o mini in this work) to perform the information extraction task and produce the final result.

2.2 IE using LLMs with In-context Learning

With the rapid development of LLMs' capability in the ICL setting, many works investigate using LLMs in the ICL setting for various tasks, including information extraction. In particular, many works focus on named entity recognition (NER) and relation extraction (RE). For instance, Jimenez Gutierrez et al. (2022) evaluates GPT-3 on biomedical NER and RE in the ICL setting, while Kwak et al. (2023) examines GPT-4's performance in legal NER and RE. Additionally, Rajpoot and Parikh (2023) investigates LLMs for financial RE in the ICL setting. Wadhwa et al. (2023) evaluates GPT-3 and Flan-T5 on standard relation extraction tasks and reports that GPT-3 achieves near SOTA performance in the few-shot setting. Xu et al. (2023) experiments with GPT-3.5 to investigate if in-context learning and data generation enhance the model's performance on the few-shot RE. Li et al. (2023a) investigates the capabilities of LLMs on zero-shot RE. Wan et al. (2023) propose a new framework for RE using LLM in the ICL setting which utilizes task-aware example retrieval and incorporates gold label-induced reasoning logic into the demonstrations. Mo et al. (2024) presents a new example retrieval technique which utilizes both the correct/positive examples and the wrong/negative examples and evaluates it on NER and RE tasks.

There are a few works that address IE tasks other than NER or RE. He et al. (2023) proposes a new framework to perform IE from visually rich documents in the ICL setting. Peng et al. (2023) demonstrates how agricultural information, which includes entities, attributes, and descriptions, can be extracted from unstructured data using LLM in the zero-shot setting. Gao et al. (2023) assesses the LLM's generalizing capability to unseen information types and tasks in the ICL setting using the fine-grained IE benchmark dataset. Guo et al. (2024) proposes a framework for unified information extraction in the ICL setting utilizing diverse demonstrations.

Compared to NER and RE, event extraction in the ICL setting has been less investigated. Sun et al. (2024) evaluates the ChatGPT's capability of extracting pharmacovigilance events in the ICL setting and reports that it performs reasonably well when used with appropriate demonstration selection strategies. Further investigation is needed to confirm this finding, as the evaluation was conducted a specific task in a single domain (i.e., medical) using a single dataset. Our work addresses this gap by investigating event extraction in the ICL setting in a distinct domain (the legal one) using a

different dataset.

3 Method

We introduce a new information extraction approach using prompt chaining. Prompt chaining is a concept introduced by Wu et al. (2022). It is a method that divides a complex task into multiple smaller steps and prompts an LLM in each step; the output from an earlier prompt becomes an input for the following prompt.

Huang et al. (2023) suggests that using a dynamic prompt containing the reduced number of examples relevant to each input text improves extraction accuracy. Their method for generating dynamic prompts involves training a BERT classifier. Unlike them, we investigate using the prompt chaining technique to generate a dynamic prompt without training or fine-tuning a model.

We divide the IE task into two steps: (1) text classification, to understand which types are likely to be present in the underlying text, and (2) IE using prompts just for the likely types. Both steps are implemented using ICL and a vanilla LLM (i.e., not fine-tuned for the task). Figure 1 depicts our overall approach.

3.1 Few-shot Text Classification

We prompt a LLM to classify an input text based on the information that is contained in it, such as the types of entities or events that are mentioned. In this work, we perform text classification only based on the event types. This is because entities are included in the examples chosen based on the event types they participate in.[1]

Figure 2 shows the prompt for the text classification task (the first component of our method). The prompt consists of three parts: task instruction, format instruction, and example. The task instruction states the system's role and provides the full list of information types. The format instruction specifies the output format with a brief demonstration. The example demonstrates how the classification should be done using the chain-of-thought technique.

Text classification is implemented with GPT-3.5[2] in a three-shot setting. k value (for k-shot learning) was tuned on the development partition.

Instruction (task)

Your task is to classify a will text based on the events contained in it.

Here's the list of events: WillCreation, Direction, Bequest, Nomination, SignWill, Attestation, Authorization, Revocation, Excuse.

Read the given will text carefully and evaluate if each event type is contained in the text.

Instruction (format)

You should produce the output in a python list format containing 0 and 1. If a will text contains an event type, then it's 1 and if it doesn't, then it's 0.

The order of the list should be [WillCreation, Direction, Bequest, Nomination, SignWill, Attestation, Authorization, Revocation, Excuse]. For example, if a will text contains a WillCreation event and a Revocation event, the expected output is [1, 0, 0, 0, 0, 0, 0, 1, 0].

Example

Let's consider a few example cases step by step.

%%
(1) Will text: I, [Person-1], domiciled in Ripley, Lauderdale County, Tennessee, do hereby make, publish, and declare this to be my Last Will and Testament, hereby revoking all wills and codicils heretofore made by me. (WillCreation, Revocation)

Expected output: [1, 0, 0, 0, 0, 0, 0, 1, 0]

Explanation: There's a phrase "make, publish, and declare this to be my Last Will and Testament". This suggests that the will text contains a WillCreation event. There's another phrase "revoking all wills and codicils". This shows that the will text contains a Revocation event.
%%

Instruction (task)

Following the instructions above, classify the will text below and give me the result in a python list (only the list).

Figure 2: The prompt for classification of events contained in text.

The temperature is set to 0 and the maximum token size for the generation is set to 4096. The model's context size is 16,385 tokens.

3.2 Few-Shot Information Extraction

We create a prompt for the information extraction task based on the text classification result. Depending on the information type that the text contains, the examples to be included in the prompt are decided. Suppose our task is to extract A, B, and C events from a given text. If the text classification output suggests that only an A event is present in the text, then we select k (1 or 5 in this work) examples that are relevant to the A event from an example pool. If there are more than k relevant examples available, we randomly select k from the set of relevant ones. The selected examples are added to the prompt for the information extraction task. An example sample in the actual output format can be found in Appendix A.

[1] Texts containing the same event types typically feature similar types of entities.

[2] The model used in this work is `gpt-3.5-turbo-0125`. https://platform.openai.com/docs/models/gpt-3-5-turbo

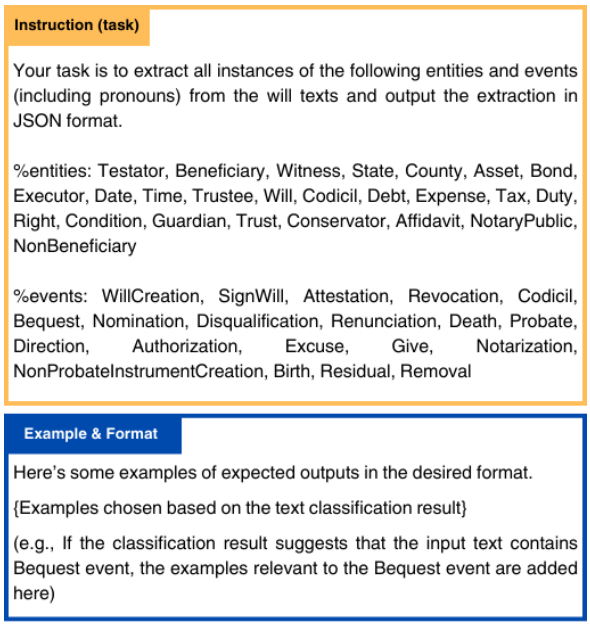

Figure 3: The prompt for information extraction, containing only the examples of entities/events identified during the text classification step.

Once the prompt is completed with these examples, we use it to ask a LLM to extract the information from the given text. Figure 3 shows the complete prompt. The prompt consists of instructions and examples. The instructions explain the task and provide the full list of information types to be extracted. The instructions are accompanied by examples chosen based on the classification output. The example demonstrates how the task should be completed while specifying the desired output format at the same time.

In this work, information extraction is done with GPT-4[3] and GPT-4o mini[4] in one-shot and five-shot settings. For both models, the temperature is set to 0, and the maximum token limit for generation is set to 4096. The context size for both models is 128,000 tokens.

4 Task and Evaluation

We evaluate our approach on the entity and event extraction tasks from the legal will dataset introduced by Kwak et al. (2023)[5]. We chose to do our evaluation on this legal task because wills contain highly diverse types of entities and events, and because the number of entity and event types is relatively large (see next subsection). Extracting many types of entities and events using the LLM in a standard few-shot setting can be challenging because the prompt should provide examples for all the entity and event types. These characteristics of the legal domain make it a good candidate for our approach, which focuses on (1) selecting examples with the same information type without depending on semantic similarity and (2) including only selected examples in the prompt.

4.1 Dataset and Task

We used the legal will dataset introduced by Kwak et al. (2023) for the evaluation. The dataset consists of wills from two US states: Tennessee and Idaho. The extractions from Tennessee wills are in-domain data while the ones from Idaho wills are out-of-domain (OOD) data. Tennessee and Idaho are considered different domains from the legal perspective since they have different probate codes (Kwak et al., 2023). The dataset contains 457 in-domain datapoints (will text segments, usually sentences) and 108 OOD datapoints. Among the 457 in-domain datapoints, 203 datapoints were used as an example pool and 145 datapoints were used as a development partition. The rest of the in-domain datapoints (109) and all of the OOD datapoints (108) were used as test partitions.

The dataset contains the annotations of 26 types of entities, 18 types of relations, and 20 types of events extracted from 45 wills. Our work focuses on extracting 25 types of entities (i.e., 26 entity types minus "Trigger") and 20 types of events.[6] In entity extraction, we identify key entities in wills, such as testator, beneficiary, executor, and asset. In event extraction, we capture key events in wills, which include will creation, signing will, bequest, nominations, and attestation. The full list of entities and events extracted in this work can be found in Appendix B. A detailed explanation of the entities and the events can be found in Kwak et al. (2023).

We prompt GPT-4 and GPT-4o mini to extract the entities and the events from given will text segments and output the result in JSON format. An example of entity and event extraction is shown as the Result in Figure 1. Although the prompt

[3]The model used in this work is `gpt-4-1106-preview`. https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4

[4]The model used in this work is `gpt-4o-mini-2024-07-18`. https://platform.openai.com/docs/models/gpt-4o-mini

[5]The dataset is licensed under CC BY-NC 4.0. Our use of the dataset is consistent with their intended use.

[6]Our primary interest lies in evaluating our approach on event extraction. Entity extraction is necessary for this purpose as entities function as arguments of events. We did not include relation extraction in our evaluation because it is less pertinent to our purpose.

Dataset	Setting	Model	Entity			Event		
			Precision	Recall	F1	Precision	Recall	F1
In domain	1-shot	Ceiling	*76.69 (0.98)*	*59.58 (1.85)*	*67.05 (1.36)*	*58.42 (0.14)*	*48.98 (0.72)*	*53.28 (0.36)*
		Prompt Chaining	76.85 (0.85)	60.08 (0.06)	67.44 (0.29)	55.06 (2.09)	45.80 (1.65)	50.00 (1.78)
		Full prompt	77.67 (0.39)	60.79 (0.92)	68.20 (0.67)	57.07 (2.34)	46.56 (2.55)	51.28 (2.48)
	5-shot	Ceiling	*84.38 (0.64)*	*70.92 (1.32)*	*77.06 (1.04)*	*73.05 (0.91)*	*65.14 (0.36)*	*68.87 (0.42)*
		Prompt Chaining	**83.49** (1.46)	**70.42** (1.71)	**76.40** (1.59)	**71.95** (1.51)	**62.72** (2.21)	**67.02** (1.88)
		Full prompt	82.09 (0.84)	65.71 (0.28)	72.99 (0.49)	70.51 (1.56)	57.51 (1.72)	63.34 (1.57)
OOD	1-shot	Ceiling	*65.43 (1.55)*	*47.12 (0.92)*	*54.78 (1.01)*	*45.40 (1.30)*	*32.72 (0.58)*	*38.02 (0.75)*
		Prompt Chaining	64.00 (0.58)	45.76 (0.66)	53.36 (0.64)	42.96 (0.88)	30.67 (0.88)	35.78 (0.88)
		Full prompt	67.58 (0.94)	49.35 (0.81)	57.04 (0.81)	45.42 (0.32)	31.08 (0.50)	36.90 (0.46)
	5-shot	Ceiling	*71.64 (0.08)*	*53.95 (0.56)*	*61.55 (0.39)*	*50.77 (2.13)*	*36.41 (0.77)*	*42.40 (1.17)*
		Prompt Chaining	**69.20** (1.28)	**52.59** (0.78)	**59.76** (0.93)	**50.53** (0.69)	**38.26** (0.95)	**43.55** (0.87)
		Full prompt	68.68 (1.02)	47.18 (0.98)	55.93 (0.86)	46.68 (0.77)	30.97 (0.63)	37.24 (0.69)

Table 1: GPT-4's results for the entity and event extraction tasks. The table shows the average scores from three-iteration experiments, with the standard deviation in parentheses. Overall, GPT-4 performs better in the 5-shot setting than a 1-shot setting. The model achieves the best F1 scores with the prompt chaining approach in both the in-domain dataset and the OOD dataset for both tasks. The results from the ceiling model (italicized) are given only to show the theoretical upper bound of our approach; they were not considered when determining the best scores because they were obtained from the ideal setting where the text classification is 100% correct.

chaining technique improves example selection for a given input, our approach is still affected by the randomness within each example pool. To mitigate this, we run our experiment three times under the same settings and report the average scores from the three iterations, with standard deviations shown in parentheses.

4.2 Evaluator

We use an automatic scoring script that compares the LLM's outputs with the gold data. The automatic evaluator compares each entity and event in the LLM's output with the one in the gold data and finds matching pairs. Any entities or events that match more than 70% with the gold data are considered to be correct in this work. As several previous works have pointed out (Wadhwa et al., 2023; Polat et al.), the open-ended nature of outputs from LLMs makes it hard to evaluate them with the predefined standards. One solution to this is to manually review the outputs, but its cost would be too high. As an alternative, we have tested our automatic evaluator with varying thresholds (60–100%) for matching. The threshold is heuristically set at 70% as it best aligned with the human reviewer's judgments during manual evaluation. A more detailed explanation of the evaluation can be found in Appendix C.

4.3 Benchmark Models

We compare our approach against two models: a *ceiling* model and a strong baseline called *full prompt* model. The ceiling model presents the results from the ideal setting where the text classification result is 100% correct. In this case, the examples in the IE prompt were chosen based on the type of events present in the gold data. This model suggests a theoretical upper bound for our approach. The full prompt model shows the results from the setting where the prompt contains the examples for all the major event types.

5 Results

5.1 Text Classification Task

In the three-shot setting, the accuracy scores for the text classification task, the first component of our method, are 96.74 for the in-domain data and 93.21 for the OOD dataset respectively. The result from the text classification task suggests that GPT-3.5 performs the text classification task well in a few-shot setting.

5.2 Information Extraction Task

5.2.1 GPT-4

Table 1 presents GPT-4's results for the entity and event extraction tasks. This is the second component of our method, which produces the final output. The table shows the average scores from three-iteration experiments, with the standard deviation in parentheses. The scores suggest that GPT-4 performs better in the 5-shot setting than in the 1-shot setting for both the in-domain test dataset and the OOD dataset. Overall, the model shows the best performance with the prompt chaining approach as suggested by the highest F1 scores

Dataset	Setting	Model	Entity			Event		
			Precision	Recall	F1	Precision	Recall	F1
In domain	1-shot	Ceiling	*80.71 (0.96)*	*60.48 (0.97)*	*69.14 (0.95)*	*65.30 (0.74)*	*52.16 (0.65)*	*57.99 (0.38)*
		Prompt Chaining	80.71 (1.34)	60.13 (1.32)	68.92 (1.32)	67.25 (1.96)	55.98 (2.52)	61.09 (2.28)
		Full prompt	80.62 (1.21)	62.56 (1.38)	70.45 (1.34)	71.49 (3.08)	53.82 (3.67)	61.40 (3.53)
	5-shot	Ceiling	*84.18 (0.57)*	*69.36 (0.48)*	*76.05 (0.52)*	*77.96 (3.02)*	*66.41 (2.04)*	*71.71 (2.29)*
		Prompt Chaining	**85.21** (0.41)	**70.31** (0.76)	**77.05** (0.57)	**77.42** (1.00)	**65.01** (1.30)	**70.67** (1.18)
		Full prompt	82.78 (0.52)	69.45 (1.01)	75.53 (0.73)	75.34 (1.90)	62.47 (0.95)	68.30 (1.17)
OOD	1-shot	Ceiling	*68.68 (2.06)*	*43.17 (1.28)*	*53.01 (1.57)*	*50.09 (0.12)*	*28.72 (0.52)*	*36.50 (0.45)*
		Prompt Chaining	68.37 (1.18)	40.74 (0.79)	51.06 (0.95)	48.17 (1.12)	27.38 (1.26)	34.91 (1.30)
		Full prompt	69.29 (1.82)	45.75 (1.01)	55.11 (1.31)	49.88 (2.08)	25.85 (1.40)	34.05 (1.70)
	5-shot	Ceiling	*74.28 (1.51)*	*50.46 (0.87)*	*60.08 (0.70)*	*58.23 (2.11)*	*33.44 (1.16)*	*42.48 (1.43)*
		Prompt Chaining	**72.69** (0.66)	49.40 (0.33)	**58.82** (0.16)	**57.23** (2.77)	**33.33** (1.63)	**42.13** (2.03)
		Full prompt	70.48 (1.73)	**49.43** (1.35)	58.11 (1.52)	49.70 (2.03)	27.08 (1.76)	35.04 (1.88)

Table 2: GPT-4o mini's results of the entity and event extraction tasks. The table shows the average scores from three-iteration experiments, with the standard deviation in parentheses. As with GPT-4, GPT-4o mini performs better in a 5-shot setting than the 1-shot setting. The model achieves the best F1 scores with the prompt chaining approach in both the in-domain dataset and the OOD dataset for both tasks. As mentioned earlier, results from the ceiling model (italicized) are given only to show the theoretical upper bound of our approach; they were not considered when determining the best scores because they were obtained from the ideal setting where the text classification is 100% correct.

within each category. In the in-domain dataset, the F1 score from the prompt chaining approach is 3.41 percentage points higher than the one from the full prompt approach for the entity extraction (76.40% vs. 72.99%). For event extraction, the score difference is 3.68 percentage points (67.02% vs. 63.34%). The score difference is even larger in the OOD dataset (3.83 percentage points for the entity extraction and 6.31 percentage points for the event extraction). In both cases, the F1 scores achieved with the prompt chaining approach are higher than the ones achieved with the full prompt approach (59.76% vs. 55.93% for entity extraction and 43.55% vs. 37.24% for event extraction).

5.2.2 GPT-4o mini

Table 2 presents GPT-4o mini's results of the entity and event extraction tasks. As explained earlier, it shows the average scores from three-iteration experiments, with the standard deviation in parentheses. The scores indicate that GPT-4o mini performs better in the 5-shot setting than in the 1-shot setting for both the in-domain test dataset and the OOD dataset, similar to the trends observed for GPT-4. GPT-4o mini performed the best with the prompt chaining approach as suggested by the highest F1 scores within each category. In the in-domain dataset, the F1 score for entity extraction with prompt chaining is 1.52 percentage points higher than with the full prompt approach (77.05% vs. 75.53%), which aligns with the performance improvements seen for GPT-4. Similarly, for event extraction, the prompt chaining approach outperforms the full prompt method by 2.37 percentage points (70.67% vs. 68.30%). The OOD dataset shows a smaller score difference for entity extraction (0.71 percentage points) but a larger one for event extraction (7.09 percentage points). Overall, the F1 scores with the prompt chaining approach exceed those of the full prompt method, consistent with the findings for GPT-4, with scores of 58.82% vs. 58.11% for entity extraction and 42.13% vs. 35.04% for event extraction.

6 Discussion

6.1 Prompt Chaining vs. Full Prompt

As suggested by the higher F1 scores, both GPT-4 and GPT-4o mini perform better with the prompt chaining approach than in the full prompt approach in the 5-shot setting. However, in the OOD dataset, GPT-4o mini achieves a marginally higher recall score with the full prompt approach than it does with the prompt chaining approach for the entity extraction task (49.43% vs. 49.40%). It is also noticeable that for GPT-4, the difference in scores between the prompt chaining approach and the full prompt approach is smaller in precision compared to recall. Specifically, for entity extraction, the precision difference is 1.4 percentage points in the in-domain dataset and 0.52 percentage points in the out-of-domain dataset, while the recall difference is 4.71 percentage points in the in-domain dataset and 5.41 percentage points in the out-of-domain dataset.

For event extraction, the precision difference is 1.44 percentage points in the in-domain dataset and 3.85 percentage points in the out-of-domain dataset, whereas the recall difference is 5.21 percentage points in the in-domain dataset and 7.28 percentage points in the out-of-domain dataset.

This tendency suggests that for GPT-4, the prompt chaining technique is more effective at reducing false negatives than reducing false positives. Providing examples specifically relevant to the input text helps the model focus on the targeted information, leading to fewer false negatives. In contrast, using a variety of example types (as in the full prompt approach) helps the model differentiate between relevant and irrelevant information, which reduces false positives. However, this pattern was not observed with GPT-4o mini, suggesting that this effect may be specific to the GPT-4 model.

Whether or not the tendency is present, the prompt chaining approach generally proves more effective than the full prompt approach when a sufficient number of examples is provided for each information type (e.g., in a 5-shot setting). This is supported by the fact that both GPT-4 and GPT-4o mini achieve higher F1 scores for entity and event extraction in both the in-domain and out-of-domain datasets when using the prompt chaining approach in the 5-shot setting.

6.2 1-Shot vs. 5-Shot

As previously noted, in the 1-shot setting, the models occasionally perform better with the full prompt approach compared to the prompt chaining approach. This is likely because the number of examples included in the prompt often becomes too small when using the prompt chaining approach in the 1-shot setting. On average, across the test and OOD datasets, the prompt chaining model includes 1.43 examples in the prompt, compared to 10 examples for the full prompt model. Considering the complexity of the task and the output format, 1.43 examples are not sufficient for the model to learn the details. The model occasionally makes formatting mistakes with the prompt-chaining approach in the 1-shot setting. This suggests that the model struggles to grasp the details of the output format from a few examples given.

In the 5-shot setting, at least 5 examples[7] are added to the prompt even in the prompt chaining scenario. As there are sufficient number of examples from which the model can learn the details of the task and the output format, the model does not show any formatting errors in the 5-shot setting.

Based on this observation, the prompt chaining approach should be used in the few-shot setting (e.g., 5-shot) rather than in the 1-shot setting to secure a sufficient number of examples, especially if the task and the output format are complex.

6.3 Ceiling vs. Prompt Chaining

The ceiling model offers the theoretical upper bound scores for the prompt chaining approach where the text classification is perfectly done. However, contrary to expectations, there are a few cases where the prompt chaining model *outperforms* the ceiling model. In the 5-shot setting, GPT-4 achieves a higher F1 score with the prompt chaining model compared to the ceiling model for event extraction in the OOD dataset (43.55% vs. 42.40%). Similarly in the 5-shot setting, GPT-4o mini obtains a higher F1 score with the prompt chaining model compared to the ceiling model for entity extraction in the test dataset (77.05% vs. 76.05%).

This can be explained by two factors: the high accuracy score for the text classification task in the test dataset (96.74%) and the randomness of the examples within the example pool for each information type. First, the accuracy of the text classification for the test dataset is very high: there are few cases where the classification results differ between the ceiling model and the prompt chaining model. This high accuracy likely stems from the formal language used in wills. This formality, aimed at ensuring clarity and legal precision, makes it easier for the model to classify these documents. Thus, the ceiling model offers little benefit over the prompt chaining model in the test dataset.

In addition, the randomness of examples within the example pool for each information type can contribute to the variability of the models' performance. The examples in the prompt are chosen based on the text classification result, but it does not guarantee consistent quality in the examples. To be precise, what is chosen is which information type's *example pool* is to be used, not the examples themselves. Once it is decided which information type's example pool is to be used, examples for

[7] When there is no major event included in the given input, it is classified as containing the 'Etc.' event and the examples for the 'Etc.' event type (which also does not contain any major event) are added. This is to prevent cases where no example is added to the prompt.

the prompt are randomly selected from within the pool. The quality of each example varies within the example pool. Some examples contain rich information while others do not. Therefore, it is possible that the examples' randomness can affect the models' performance. If the prompt chaining model is randomly given examples with rich information while the ceiling model is randomly given examples that contain less information, it is possible that the prompt chaining model could exceed the ceiling model.

6.4 Performance Decrease in Out-of-Domain

The performance of the LLMs (both GPT-4 and GPT-4o-mini) shows a marked decline on out-of-domain (OOD) data. For the entity extraction task, F1 scores in the in-domain setting range from 67.05 to 77.06, while in the OOD setting, they drop to a range of 51.06 to 61.55. Similarly, for the event extraction task, F1 scores range from 50.00 to 71.71 in-domain, but fall to 34.05 to 43.55 in the OOD scenario. This trend is consistent across all three models tested (ceiling, prompt-chaining, and full-prompt), with no approach showing a significantly larger drop in performance. This suggests that the performance degradation is more likely due to domain differences rather than any specific fault of the models themselves.

The error analysis of the OOD partition suggests that the performance decline is at least partially due to differences in formality between the two domains (i.e., Tennessee wills and Idaho wills). Idaho operates under a different probate code than Tennessee, and the template for drafting wills also varies. Idaho wills often include clauses that are uncommon in Tennessee wills. For example, declarations of marital status and/or children are frequently included at the beginning of Idaho wills, whereas such declarations rarely appear in Tennessee wills. Another example is the inclusion of no-contest clauses, which prevent beneficiaries from contesting the will. These clauses are common in Idaho wills but infrequent in Tennessee. This variation in formality leads to high error rates, as there are few relevant examples available for such cases.

6.5 Cost-Efficiency of Prompt Chaining

The prompt chaining approach not only improves the overall performance of the model, but also provides cost benefits. By using only examples that are relevant to information contained in the input text, it allows the prompt to have smaller tokens than with the full prompt approach has. As the API services for the LLM bill their clients based on token number, reducing the number of tokens in the prompt offers benefits in terms of lower cost.

For example, in our work, the average number of tokens per example is 468.89, and each input text contains an average of 1.43 information types. For the full prompt approach, we use the examples for the 10 major event types[8]. With a quick calculation, we conclude that the input for the prompt chaining approach contains 4018.39 ($468.89 \times 10 - 468.89 \times 1.43 = 4018.39$) fewer tokens compared to the full prompt approach, making the input processing cost 6.99 times cheaper (10/1.43 = 6.99) in our case. As demonstrated by this example, the prompt chaining approach offers cost benefits while also improving the model's overall performance on the task.

7 Conclusion

This work introduces a new prompt chaining technique for information extraction. The key idea of this approach is to split the information extraction into two steps: (1) text classification to understand which entity/event types are likely to be present, and (2) information extraction for the identified types. Both steps are implemented using an LLM with in-context learning. By classifying each input text based on the information type present in it first, we can complete the prompt for the information extraction task with the examples that are relevant to each input text. With the completed prompt, we ask a LLM to conduct the information extraction task. We evaluate this technique on entity and event extraction tasks in the legal domain. The evaluation results demonstrate that the prompt chaining technique improves the model's overall performance. The prompt chaining approach also provides cost benefits by reducing the number of tokens in the prompt. The code used in this work can be found at: https://github.com/ml4ai/pc4wills/

8 Limitations

The prompt chaining technique introduced in our work can be adapted to various IE tasks and used

[8]The 10 major event types include 9 event types listed in the text classification prompt plus the 'Etc.' event type. Any event type that either (1) does not occur independently of other event types (e.g., 'Death' event type does not occur on its own; it always accompanies other event type as it is used as a condition for another event.) or (2) has less than 50 occurrences across all the datasets falls under 'Etc.' category.

in different domains. However, we evaluated the technique with only a few models (i.e., GPT-4 and GPT-4o-mini) and a single dataset. Our findings need to be confirmed with further evaluation on different models and/or datasets. Even though the prompt chaining technique helps select better examples for the given input, our approach is still prone to the randomness of the examples within each example pool as discussed in the section 6.3. Using a semantic similarity-based technique for ICL example selection in conjunction with ours might mitigate this issue, as they are complementary to each other. Further investigation is needed to confirm this hypothesis.

Acknowledgments

We thank the reviewers for their thoughtful comments and suggestions. This work was partially supported by the National Science Foundation (NSF) under grant #2217215, and by University of Arizona's Provost Investment Fund. Mihai Surdeanu and Clayton Morrison declares a financial interest in lum.ai. This interest has been properly disclosed to the University of Arizona Institutional Review Committee and is managed in accordance with its conflict of interest policies.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jun Gao, Huan Zhao, Yice Zhang, Wei Wang, Changlong Yu, and Ruifeng Xu. 2023. Benchmarking large language models with augmented instructions for fine-grained information extraction. *Preprint*, arXiv:2310.05092.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Qian Guo, Yi Guo, and Jin Zhao. 2024. Diluie: Constructing diverse demonstrations of in-context learning with large language model for unified information extraction. *Neural Computing and Applications*.

Jiabang He, Lei Wang, Yingpeng Hu, Ning Liu, Hui juan Liu, Xingdong Xu, and Hengtao Shen. 2023. Icl-d3ie: In-context learning with diverse demonstrations updating for document information extraction. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19428–19437.

Qing Huang, Yanbang Sun, Zhenchang Xing, Min Yu, Xiwei Xu, and Qinghua Lu. 2023. Api entity and relation joint extraction from text via dynamic prompt-tuned language model. *ACM Trans. Softw. Eng. Methodol.*, 33(1).

Bernal Jimenez Gutierrez, Nikolas McNeal, Clayton Washington, You Chen, Lang Li, Huan Sun, and Yu Su. 2022. Thinking about GPT-3 in-context learning for biomedical IE? think again. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4497–4512, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alice Kwak, Cheonkam Jeong, Gaetano Forte, Derek Bambauer, Clayton Morrison, and Mihai Surdeanu. 2023. Information extraction from legal wills: How well does GPT-4 do? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4336–4353, Singapore. Association for Computational Linguistics.

Guozheng Li, Peng Wang, and Wenjun Ke. 2023a. Revisiting large language models as zero-shot relation extractors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6877–6892, Singapore. Association for Computational Linguistics.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023b. Unified demonstration retriever for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4644–4668, Toronto, Canada. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Man Luo, Xin Xu, Zhuyun Dai, Panupong Pasupat, Mehran Kazemi, Chitta Baral, Vaiva Imbrasaite, and Vincent Y Zhao. 2023. Dr.icl: Demonstration-retrieved in-context learning. *Preprint*, arXiv:2305.14128.

Ying Mo, Jian Yang, Jiahao Liu, Shun Zhang, Jingang Wang, and Zhoujun Li. 2024. C-icl: Contrastive in-context learning for information extraction. *Preprint*, arXiv:2402.11254.

Ruoling Peng, Kang Liu, Po Yang, Zhipeng Yuan, and Shunbao Li. 2023. Embedding-based retrieval with llm for effective agriculture information extracting from unstructured data. *Preprint*, arXiv:2308.03107.

Fina Polat, Ilaria Tiddi, and Paul Groth. Testing prompt engineering methods for knowledge extraction from text.

Pawan Rajpoot and Ankur Parikh. 2023. GPT-FinRE: In-context learning for financial relation extraction using large language models. In *Proceedings of the Sixth Workshop on Financial Technology and Natural Language Processing*, pages 42–45, Bali, Indonesia. Association for Computational Linguistics.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Zhaoyue Sun, Gabriele Pergola, Byron Wallace, and Yulan He. 2024. Leveraging ChatGPT in pharmacovigilance event extraction: An empirical study. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 344–357, St. Julian's, Malta. Association for Computational Linguistics.

Somin Wadhwa, Silvio Amir, and Byron Wallace. 2023. Revisiting relation extraction in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15566–15589, Toronto, Canada. Association for Computational Linguistics.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. GPT-RE: In-context learning for relation extraction using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3534–3547, Singapore. Association for Computational Linguistics.

Liang Wang, Nan Yang, and Furu Wei. 2024. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1752–1767, St. Julian's, Malta. Association for Computational Linguistics.

Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA. Association for Computing Machinery.

Xin Xu, Yuqi Zhu, Xiaohan Wang, and Ningyu Zhang. 2023. How to unleash the power of large language models for few-shot relation extraction? In *Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, pages 190–200, Toronto, Canada (Hybrid). Association for Computational Linguistics.

Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Veselin Stoyanov, Greg Durrett, and Ramakanth Pasunuru. 2023. Complementary explanations for effective in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4469–4484, Toronto, Canada. Association for Computational Linguistics.

A Example for the Information Extraction Task


```
{
    "text": "II (A) I give, devise and
        bequeath all my property, real,
        personal and mixed, of whatever
        kind and nature and wheresoever
        situated, to my wife, [Person
        -2], if she survives me.",
    "entities": [
        {
            "id": "e1",
            "texts": [
                "all my property, real,
                    personal and mixed,
                    of whatever kind and
                     nature and
                    wheresoever situated
                    "
            ],
            "type": "Asset"
        },
        {
            "id": "e2",
            "texts": [
                "if she survives me"
            ],
            "type": "Condition"
        },
        {
            "id": "e3",
            "texts": [
                "my",
                "I",
                "me"
            ],
            "type": "Testator"
        },
        {
            "id": "e4",
            "texts": [
                "my wife",
                "[Person-2]",
                "she"
            ],
            "type": "Beneficiary"
        }
    ],
    "events": [
        {
            "id": "v1",
            "type": "Bequest",
            "Asset": [
                "e1"
            ],
            "Condition": [
                "e2"
            ],
            "Testator": [
                "e3"
            ],
            "Beneficiary": [
                "e4"
            ]
        }
    ]
}
```


B Data Taxonomy

Below is a list of the entities and events extracted during our task. Each entity is accompanied by a description, while each event includes both a description and its associated arguments. The descriptions in this list were sourced from Kwak et al. (2023).

- Entities
  - *Testator*: a person who makes a will
  - *Beneficiary*: a person or an entity (e.g., organization) that receives something from a will
  - *Executor*: a person who executes a will (=personal representative)
  - *Witness*: a person witnessing a will
  - *Trustee*: a person who manages a trust
  - *Guardian*: a person who has a legal right and responsibility of taking care of someone who cannot takes care of themselves (usually a minor or an legally incompetent person)
  - *Conservator*: a person who handles the financial and personal affairs who cannot handles such affairs by themselves (usually a minor or an legally incompetent person)
  - *Notary Public*: a person who is authorized by state government to witness the signing of important documents and administer oaths
  - *Non-Beneficiary*: a person who is excluded from being beneficiary
  - *State*: any US state names
  - *County*: any US county names
  - *Date*: any dates
  - *Time*: any expression denoting a particular point in time
  - *Condition*: a condition under which an event (e.g., will execution, bequest, etc.) occurs
  - *Asset*: any money, personal property, or real estate owned by a testator
  - *Bond*: any bonds (usually probate bonds, which is a type of bond ordered and required by a court before they will appoint a person or entity as the personal representative of an estate)
  - *Debt*: any debts

- *Expense*: any expenses
- *Tax*: any taxes
- *Trust*: a fiduciary arrangement that allows a trustee to hold assets on behalf of a beneficiary
- *Duty*: any duty directed by a testator to fiduciaries (e.g., executors, trustees, guardians, or conservators)
- *Right*: any rights authorized by a testator to fiduciaries (e.g., executors, trustees, guardians, or conservators)
- *Will*: a legal document containing a person's wishes regarding the disposal of one's asset after death
- *Codicil*: a testamentary or supplementary document that modifies or revokes a will or part of a will
- *Affidavit*: a legal statement sworn and signed by a testator and witnesses to confirm the validity of a will (usually attached to a will)

• Events

- *Will Creation*: an event in which a testator creates a will
  - Testator
  - Will
  - Condition
- *Sign Will*: an event in which a testator or a witness signs a will
  - Testator
  - Will
  - Date
  - Condition
- *Attestation*: an event in which a witness attests the validity of a will
  - Witness
  - Attested events (e.g., Sign Will)
- *Revocation*: an event in which a testator revokes a will or a codicil
  - Testator
  - Will
  - Codicil
- *Codicil*: an event in which a codicil is made
  - Testator
  - Codicil
  - Time
- *Bequest*: an event in which a testator bequeath asset to a beneficiary
  - Testator
  - Asset
  - Beneficiary
  - Condition
- *Nomination*: an event in which a testator nominates a fiduciary
  - Testator
  - Executor
  - Trustee
  - Guardian
  - Conservator
  - Condition
- *Disqualification*: an event in which a beneficiary or a fiduciary is disqualified
  - Executor
  - Beneficiary
- *Renunciation*: an event in which a fiduciary renounces
  - Executor
- *Death*: an event in which any entity (e.g., testator, beneficiary, executor, etc.) dies
  - Testator
  - Beneficiary
  - Executor
- *Probate*: an event in which a will or any part of the will is probated
  - Will
  - Debt
  - Expense
  - Tax
  - Expense
  - Condition
  - Time
- *Direction*: an event in which a testator gives direction to someone (usually a fiduciary)
  - Testator
  - Executor
  - Duty
  - Directed events (e.g., Excuse)
- *Authorization*: an event in which a testator authorizes a fiduciary to a right
  - Testator
  - Executor
  - Right
  - Condition

- *Excuse*: an event in which a testator excuses a fiduciary from a duty
  - Testator
  - Executor
  - Duty
  - Bond
- *Give*: an event in which a testator gives a compensation to a fiduciary
  - Testator
  - Executor
  - Asset
  - Time
  - Condition
- *Notarization*: an event in which an affidavit is notarized by a notary public
  - Notary Public
  - Date
- *Non Probate Instrument Creation*: an event in which a non probate instrument (e.g., trust) is created
  - Testator
  - Asset
  - Trust
  - Condition
- *Birth*: an event in which a beneficiary is born
  - Beneficiary
  - Date
- *Residual*: an event in which asset becomes residuary estate
  - Asset
  - Condition
- *Removal*: an event in which a beneficiary is removed from a will
  - Beneficiary
  - Condition

C Description of Evaluation

The evaluator compares LLM's outputs against gold data, utilizing advanced similarity metrics for both entities and events. It comprises several key components:

1. Optimal matching: It is essential to match predicted entities and events with those in the gold data, as data contains multiple entities and events. The evaluator implements a greedy approach to identify optimal pairings between the predicted and the gold data. It operates at both the list level (for entity matching) and the dictionary level (for event matching).

2. Similarity Computation: The evaluator implements two distinct approaches: a) For entities: A weighted combination of type matching and text similarity. b) For events: A set-based comparison of key-value pairs, excluding the 'id' field. Similarity is calculated as the ratio of common values to total unique values across both dictionaries. True positives (TP), false positives (FP), and false negatives (FN) for both entities and events are calculated based on similarity thresholds.

3. Metrics Calculation: The evaluator computes precision, recall, and F1 score based on the TP, FP, and FN counts calculated earlier.

The evaluator employs a similarity threshold to determine whether the predicted output matches the gold data. The threshold in this work is heuristically set at 70% as it best aligned with the human reviewer's judgments. Below are the examples that received a similarity score of over 70%:

- Entity:
  - Gold data: my will (type: Will)
  - Predicted output: this my will (type: Will)
  - Similarity score: 73.68%
- Event:
  - Gold data:
    {'id': 'v1',
    'type': 'Authorization',
    'Condition': ['e1'],
    'Executor': ['e2'],
    'Testator': ['e3'],
    'Right': ['e4']}
  - Predicted output:
    {'id': 'v1',
    'type': 'Authorization',
    'Right': ['e1', 'e4'],
    'Executor': ['e2'],
    'Testator': ['e3']}
  - Similarity score: 75%

In both cases, the difference between the gold data and the predicted output is not significant. For the entity, the only variation is the addition of "this"

before "my will", which is not necessarily incorrect. In the case of the event, the predicted output categorized one entity (e1) differently, but this distinction does not significantly impact the overall results.

The examples below are the ones that received a similarity score of less than 70%:

- Entity:
  - Gold data: Idaho (type: County)
  - Predicted output: Buhl, Idaho (type: County)
  - Similarity score: 62.5%
- Event:
  - Gold data:
    {'id': 'v4',
    'type': 'Probate',
    'Tax': ['e2'],
    'Expense': ['e3', 'e14'],
    'Debt': ['e12'],
    'Condition': ['e10']}
  - Predicted output:
    {'id': 'v2',
    'type': 'Probate',
    'Expense': ['e3', 'e14'],
    'Debt': ['e12'],
    'Condition': ['e4']}
  - Similarity score: 66.67%

The difference between the gold data and the predicted output is more prominent in these cases. For example, it is evident that "Buhl, Idaho" is an incorrect extraction for county. It is also clear that the event from the predicted output misses a key argument ('Tax') and incorrectly identifies a condition ('e4' instead of 'e10').

The code and additional details can be found at: https://github.com/ml4ai/pc4wills/