Prompt Optimization Literature Survey

Name: Michael Bressler, 03733031

Summaries of Covered Papers

CLAUDETTE: an Automated Detector of Potentially Unfair Clauses in Online Terms of Service

DATASET The Authors produced a dataset of 12.000 sentences, individually sourced from consumer contracts of 50 large online companies (e.g. Terms of Service) meant for contractual relationships with customers in the European Union. The dataset is the result of an XML-based annotation process ending in a segmentation into sentences. During the annotation process, XML-Tags were used to label paragraphs into 8 different commonly seen types of clauses (Arbitration, Unilateral change, Content removal, Jurisdiction, Choice of law, Limitation of liability, Unilateral termination, Contract by using). Additionally a numeric value of either 1 (clearly fair), 2 (potentially unfair), 3 (clearly unfair) was appended to the XML-tags to label the fairness of a particular clause type. In case a clause covered more than one paragraph, each paragraph was labeled individually. In case a paragraph contained more than one clause type, nested tags were used. The research goal is to use the annotated dataset (delivered in the form of sentences) to develop methods to both classify if a given sentence contains a potentially unfair clause and categorize the unfair clause type.
METHOD For the first task of detecting potentially unfair clauses (binary classification), various machine learning methods were compared and combined with traditional NLP methodology: This resulted in 8 different classifiers using a mix of methodologies of Support Vector Machines, Bag-of-Words feature representation, Part-of-Speech tag features, Tree Kernel-based classifiers using constituency parse trees, Structured Support Vector Machines with Hidden Markov Model sequence modeling, Convolutional Neural Networks, Long Short-Term Memory neural networks, and word embedding representations. For the second task of type of unfair clause categorization 8 Support-Vector-Machine-Classifiers were employed, one each for the 8 different unfair clause types.
METRICS For the detection task the best classifier (C8: Ensemble, combining C1, C2, C3, C6, C7) achieved a Precision of 0.828, Recall of 0.798 and and F1-Score of 0.806.
For the type of unfair clause categorization task the F1-Scores for each of the topical classifiers are as follows: Arbitration 0.823, Unilateral change, 0.823, Content removal 0.745, Jurisdiction 0.97, Choice of law 0.932, Limitation of Liability 0.932, Unilateral termination 0.853, Contract by using 0.953.
RESULTS & ANALYSIS The results clearly show that the type of unfair clause categorization task is simpler (due to higher F1-Scores) than the detection task. Nevertheless the ensemble method that performed best in the detection task still shows a promising result with an F1-Score of 0.806, correctly classifying about 81% of sentences, showing great potential in further research into and applications in automatically identifying unfair contractual practices.

Large Language Models as Optimizers

DATASET The paper uses datasets from three kinds of tasks: solving math equations using linear regression, finding the shortest path between cities using the Traveling Salesman Problem, and improving written prompts for language tasks using datasets like GSM8K and Big-Bench Hard. These datasets are used to test whether large language models can suggest better solutions or prompts through repeated improvements.
METHOD The main method is called Optimization by Prompting, where large language models are asked to improve a solution step-by-step using natural language. The models tested as optimizers include Instruction-tuned PaLM 2-L, Pre-trained PaLM 2-L, Text-Bison, GPT-3.5-Turbo, and GPT-4.
METRICS For the prompt optimization task on GSM8K, using Instruction-tuned PaLM 2-L as optimizer and Pre-trained PaLM 2-L as scorer gives the best test accuracy of 80.2%. For the

Traveling Salesman Problem with 10 nodes, GPT-4 gets an optimality gap of 0.0% in just 9.6 steps, meaning it finds the perfect solution every time and does so very efficiently.
RESULTS & ANALYSIS The results show that large language models can act like optimizers and improve their solutions over time through simple language-based instructions. GPT-4 works best for solving math-based problems, while prompt optimization with other models gives better performance than human-written prompts. However, these methods struggle with very large problems because the models can only remember so much at once and may get stuck.

Automatic Prompt Optimization with "Gradient Descent" and Beam Search

DATASET The dataset consists of four text classification tasks: Jailbreak detection, Ethos hate speech detection, Liar fake news detection, and Sarcasm detection. Each dataset contains a few hundred to several thousand text samples with binary labels, and they are used to evaluate whether automatically improved prompts can help large language models make better predictions.
METHOD The paper introduces a method called Prompt Optimization with Textual Gradients (ProTeGi), which mimics gradient descent using natural language feedback. It compares ProTeGi with other nonparametric methods like Monte Carlo sampling, Reinforcement Learning-based prompt editing, and AutoGPT feedback loops.
METRICS The best-performing method, ProTeGi with beam search, achieved F1 scores of 0.88 on Sarcasm detection, 0.85 on Jailbreak detection, and 0.67 on the Liar dataset, outperforming all other baselines in every task.
RESULTS & ANALYSIS The results show that ProTeGi reliably improves prompt quality across all tasks and often outperforms other methods by a significant margin. The method works well because it combines model feedback with structured prompt editing, though its performance can vary based on the dataset and is limited by API call costs and model context size.

GrIPS: Gradient-free, Edit-based Instruction Search for Prompting Large Language Models

DATASET The dataset comes from NATURAL-INSTRUCTIONS and includes eight binary text classification tasks, each with task descriptions and labeled examples. These tasks are used to test whether editing written instructions can help large language models follow directions better and make more accurate predictions.
METHOD The main method is called Gradient-free Instructional Prompt Search, which edits parts of instructions (like deleting, swapping, or paraphrasing phrases) to find better versions that improve model performance. It is tested with several models including GPT-2 XL, InstructGPT babbage, and InstructGPT curie, and is compared against manual rewriting, example search, and gradient-based tuning methods.
METRICS The best-performing method, GRIPS with beam search using GPT-2 XL, achieves an accuracy of 56.5%, compared to 48.4% with no search. GRIPS consistently improves performance across tasks by up to 9.4 percentage points over the original instructions.
RESULTS & ANALYSIS The results show that GRIPS improves accuracy more than manual prompt rewriting and even matches or beats gradient-based methods, despite not needing access to model internals. It works even with very little training data and is especially effective on models designed to follow instructions, though it can also help with models that are not instruction-tuned.

Are Large Language Models Good Prompt Optimizers?

DATASET The paper evaluates prompt optimization methods using four tasks (and according datasets) from the Big-Bench benchmark: Snarks, Navigate, Question Selection, and Object Counting. These tasks involve language understanding and reasoning and are used to test whether large language models can generate improved prompts that lead to better performance on these tasks.

METHOD The study compares several automatic prompt optimization methods: Iterative Automatic Prompt Editing (a resampling method), Automatic Prompt Optimization (explicit reflection), PromptAgent (explicit reflection with expert knowledge), OPRO (implicit reflection), and a new approach introduced in the paper called Automatic Behavior Optimization, which directly optimizes model behavior using demonstrations.
METRICS The best-performing method, Automatic Behavior Optimization, achieved up to 0.975 accuracy on Object Counting, 0.985 on Navigate, 0.905 on Question Selection, and 0.811 on Snarks when tested with the GPT-3.5-Turbo model.
RESULTS & ANALYSIS The results show that traditional reflection-based prompt optimization methods are often ineffective because large language models tend to generate repetitive and sometimes irrelevant feedback. In contrast, Automatic Behavior Optimization significantly improves performance by guiding models more explicitly and ensuring they follow instructions, making it a more reliable method for prompt improvement.

Summary of Added Paper

Can GPT-3 Perform Statutory Reasoning?

DATASET The paper uses a dataset called the Statutory Reasoning Assessment (SARA), which contains 376 carefully written legal cases based on nine sections of the U.S. tax code. Each case includes a short story (the facts), a yes/no question, and requires applying one or more statutes to determine the correct answer. The questions focus on legal reasoning — whether the facts legally "entail" or "contradict" a legal conclusion. The authors also created a separate set of synthetic cases using made-up laws and terms, to test if GPT-3 can reason over statutes it definitely has never seen before.
METHOD The authors test three main prompting methods using the GPT-3 model (specifically, the text-davinci-003 version). These are: zero-shot prompting (no examples provided), four-shot dynamic prompting (using the four most similar examples from training), and ten-shot chain-of-thought prompting (with detailed human-written reasoning). Each of these setups is tested in multiple ways — both with and without including the relevant statute text, and both with and without adding the prompt phrase “Let’s think step by step,” which is known to improve logical reasoning. These variations help explore how different prompt designs affect GPT-3’s legal reasoning accuracy.
METRICS The researchers measure GPT-3’s performance using accuracy — the percentage of correct answers out of total questions. The best-performing configuration was zero-shot prompting without including the statute text and without the step-by-step reasoning phrase, which achieved 74% accuracy on cases involving numbers and 71% overall. This outperformed previous state-of-the-art methods based on BERT, which had only reached 59% accuracy. Precision, recall, and F1 score are not used in this paper, since the task is framed more as answering binary legal questions correctly rather than classifying examples into multiple categories.
RESULTS & ANALYSIS The results show that GPT-3 can perform better than older models on statutory reasoning tasks when prompted carefully, but it still makes many consistent errors. It often struggles with numerical reasoning and sometimes misremembers the structure or wording of U.S. statutes, even when the text is included in the prompt. When tested on synthetic laws — written in plain language and using unfamiliar terms — GPT-3 fails frequently, suggesting it relies more on memorized patterns than on true logical reasoning. Overall, the findings suggest that while GPT-3 has potential, it is not yet reliable for real legal reasoning tasks and highlights the need for more robust models in this area.

Overall Discussion of Topics Raised by Added Paper with References to Prior Papers (1 page max) / (Research Proposal)

My added paper explores whether a language model can apply legal rules to factual scenarios by reasoning over the structure and logic of real or synthetic laws. It reveals that GPT-3 can achieve decent accuracy with well-crafted prompts but often fails when the task demands genuine logical interpretation, especially when the law is unfamiliar or phrased in synthetic language. This highlights a key challenge: GPT-3 can mimic legal reasoning patterns but lacks deep understanding of formal statutory logic. Nevertheless there has been significant improvements to reasoning models in recent time[1], potentially warranting a revisit of the topic of statutory reasoning with current reasoning models. This Approach strongly contrasts with the more engineering-focused success seen in the Claudette paper, where classifiers are trained to recognize unfair clauses purely based on statistical and structural features. While Claudette shows impressive performance in detecting and categorizing known clause types, it does not engage in legal reasoning or apply formal legal criteria to determine actual illegality. This gap, between detecting a clause and proving it's illegal under a clear law, is precisely where insights from the statutory reasoning paper could potentially be a rewarding exploration. My research proposal would focus on trying to bridge this gap. The task would be to assess whether a specific clause (e.g., a jurisdiction clause) from the Claudette XML data is not just potentially unfair, but actually illegal, by reasoning through a clearly defined legal rule, for example, corresponding European Union consumer protection directives that explicitly forbids such clauses in consumer contracts. First, this law would be translated into a simplified set of logical conditions, either by hand or with model assistance. Then, large language models could be prompted, or guided through Retrieval-Augmented Generation, to reason whether a given full clause from the XML satisfies or violates those logical rules. The clause text would be presented to the model, along with the logical structure of the rule (e.g., "If the clause forces the consumer to sue outside their own country, then it violates Regulation X"). Manual prompt design or automatic prompt optimization (inspired by methods like GRIPS or ProTeGi) could be used to improve the clarity and relevance of prompts, depending on the structure and complexity of each clause. The model's answer would not just be classification but also a short reasoning chain explaining the legal logic, modeled after chain-of-thought prompting, showing how the rule applies. This project would combine strengths from both streams: the classification and clause-type knowledge from Claudette and the reasoning-driven prompt design from the statutory reasoning study. The goal would be to explore a methodology that doesn't just detect clause types on a statistical basis, but also explains, in a legally informed way, why a clause potentially violates specific, clear, and enforceable rules. This exploration could also focus on documenting the reasoning process in a traceable way, as traceability of legal assesments is a key value in jurisprudence and would be expected to proof useful in many future potential applications of automatic tools in legal questions.

[1] Guo, Daya, et al. "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning." arXiv preprint arXiv:2501.12948 (2025).

References

Lippi, Marco, et al. "CLAUDETTE: an automated detector of potentially unfair clauses in online terms of service." *Artificial Intelligence and Law* 27 (2019): 117-139.

Yang, Chengrun, et al. "Large language models as optimizers." *arXiv preprint arXiv:2309.03409* (2023).

Pryzant, Reid, et al. "Automatic prompt optimization with" gradient descent" and beam search." *arXiv preprint arXiv:2305.03495* (2023).

Prasad, Archiki, et al. "Grips: Gradient-free, edit-based instruction search for prompting large language models." *arXiv preprint arXiv:2203.07281* (2022).

Ma, Ruotian, et al. "Are large language models good prompt optimizers?." *arXiv preprint arXiv:2402.02101* (2024).

Blair-Stanek, Andrew, Nils Holzenberger, and Benjamin Van Durme. "Can gpt-3 perform statutory reasoning?." *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*. 2023.

Guo, Daya, et al. "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning." *arXiv preprint arXiv:2501.12948* (2025).