Experiments with LLMs for Converting Language to Logic

Tanel Tammet[1][0000−0003−4414−3874], Priit Järv[1][0000−0001−7725−543X], Martin Verrev[1][0000−0003−4890−9283], and Dirk Draheim[2][0000−0003−3376−7489]

[1] Applied Artificial Intelligence Group,
Tallinn University of Technology, Tallinn, Estonia
{tanel.tammet,priit.jarv1,martin.verrev}@taltech.ee
[2] Information Systems Group, Tallinn University of Technology, Tallinn, Estonia
dirk.draheim@taltech.ee

Abstract. While the meaning of "semantic parsing" covers a wide spectrum, we consider converting English to first order predicate logic (FOL) with the help of large language models (LLMs). The paper focuses on experiments with different approaches of using an LLM for semantic parsing to FOL: from standalone zero-shot and multishot scenarios to the use as a specialized component in several stages of the semantic parser pipeline. The goal of the experiments is to determine promising approaches for including LLM components into a question answering pipeline built around a logical reasoner with extensions for commonsense reasoning.

1 Introduction

Large language models (LLMs) are a breakthrough technology in natural language understanding and commonsense reasoning. However – using the terminology popularized by Kahneman in [1] – they appear to exhibit primarily intuitive "system one" reasoning and are severely lacking in deliberate and logical "system two" reasoning. On one hand, LLMs tend to give incorrect/invented answers: the well-known "hallucination" phenomenon. On the other hand, LLMs are conceptually not well suited for problems requiring search, like planning (see [20], [3]), theorem proving or finding long chains of inferences. Simply put, LLMs do not contain algorithms for search. Using LLMs for converting natural language information and queries to a format usable for planning, reasoning and search software is one of the promising ways to make progress on connecting intuitive and logic-based reasoning.

Although the classic field of semantic parsing has employed machine learning for decades, the practical use of semantic parsers has been limited. As indicated in a survey [12], a growing body of research, e.g. [19], [6], [7], [10], [4] and [9], is devoted to using LLMs as specialized semantic parsers producing symbolic input for conventional software systems such as planners, reasoners or programming language interpreters, which are then employed to solve the tasks originally posed in natural language. These systems, in turn, may gain performance from

components using machine learning, which are specialized for the task. In particular, SEDAC

The focus of our paper is on the experimental use of LLMs for semantic parsing, in order to generate a representation which is usable for a wide spectrum of query answering and planning tasks, either directly or via further transformations. We use enhanced first order logic (FOL) for symbolic representation and verify the generation of representation by using an enhanced first order logic theorem prover for question answering. The goals and methods of our work are similar to [9], the authors of which use both a hand-crafted semantic parser and an LLM prompt to convert English to logic programs. The last paper introduces an algorithm called SEDAC for correcting semantic errors for a specific class of problems, while our paper considers a wider class of problems, focuses on FOL and does not introduce a concrete error-correcting algorithm.

2 The Question Answering Pipeline

We have built a full pipeline for question answering in natural language, [3] containing a semantic parser, a reasoner and a proof verbalizer. We will give only a brief overview of the whole system, referring to the papers [17], [16], [15] and [5] for describing the pipeline and the extensions implemented in our FOL reasoner. The main extension to FOL is adding numerical confidence to clauses and implementing default logic [13] by including special literals to encode exceptions, as presented in our papers [16] and [15]. For reasoning the pipeline calls our commonsense reasoner GK, written in C.

2.1 Basic Semantic Parsing and the Question of Correctness

Our approach to semantic parsing is to generate a detailed representation in FOL which is usable for a wide spectrum of query answering and planning tasks, either directly or via further transformations. The parser takes English strings of natural language text as input and outputs extended clausified first-order logic formulas encoded in JSON as proposed in JSON-LD-LOGIC [18]. As a first step it uses the external Stanza parser [11] from Stanford, giving a Universal Dependencies (UD, see [2]) graph. It then uses a series of simplification rules on the UD graph, interleaved with further Stanza calls, until the final UD graph is made. The main part of the parser takes the UD graph and runs several passes, building iteratively enhanced internal representation, resulting in an annotated FOL formula. The latter is finally converted and clausified to a FOL clause set passed to the reasoner.

As expected, the parser is able to correctly capture all the information from the relatively simple sentence structures like “Elephants have long trunks and no wings”, “If X is a father of Y, Y is a child of X”, “John smokes with a probability 90%”, “The nice bear who was white and ate a big fish also ate blue berries”,

[3] http://github.com/tammet/nlpsolver

"A man liked a car which a nice woman bought", "Mike ate berries in the forest which was bought by Mary", "The judges admired the lawyers that supported the secretaries" etc.

By *correctly* we mean the ability of the pipeline to correctly answer a wealth of natural language questions with the answers logically derivable from the rules given and the input sentences presented, like "The lawyers admired the secretaries?", "The judges admired the secretaries?", "The woman bought the car?", "Mike ate in the forest?", ""Who ate in the forest?" etc. Our regression tests consist of ca 1500 such assumptions/question pairs along with the expected answers.

However, properly handling sentences with the complexity typically found in Wikipedia, like 'Situated on a bay in north Estonia, on the shore of the Gulf of Finland of the Baltic Sea, Tallinn has a population of about 461,000 (as of 2024) and administratively lies in the Harju maakond (county)", is out of the scope of the capabilities of the parser. Although theoretically the parser could be enhanced ad infinitum, the effect/cost ratio of this work decreases very quickly.

3 Preliminaries: Test Sets and Direct Reasoning by GPT

Our regression tests currently contain ca 1300 assumptions/question/answer triplets made by us, plus ca 120 similar triplets built by taking a representative subset of the HANS test set [8] and 38 triplets built by us from the examples of the Allen AI ProofWriter system [14]. Finally we have built a set of hard problems for parsing (*wiki*): 90 questions built by selecting two or three first sentences from ten different representative articles of English Wikipedia, adding a few simple ballast sentences and asking questions with an expected answer *True*, *False* or *Unknown*, three of each for every article.

As expected, our pipeline is able to correctly answer all of the questions from the regression tests, while being unable to reasonably parse the sentences from the *wiki* set. The actual reasoning tasks for the tests are very simple, since the focus is on the correctness of parsing.

We have also built three GPT-oriented test sets *core*, *hans* and *allen* by modifying corresponding subsets of the regression tests described above. A number of our questions ask for specific objects (*who*, *what*, *where* etc), the results of which are not easy to compare. Thus we select only the problems with a boolean question, i.e. the answer is either *true*, *false* or *unknown*. In cases where the question could be possibly answered from the world knowledge like "Birds fly?" we replace the relevant words by an invented word like *crenkil*.

Although direct question answering by LLMs is not the focus of our paper, we will nevertheless provide a quick overview of the performance of GPT3 and GPT4 on the test sets. For each question we pose a suitable prompt to the GPT3 version gpt-3.5-turbo-0125 and GPT4 version gpt-4o-2024-05-13 via the API, setting GPT temperature (randomness factor) to zero. An example of a prompt: "You have to check whether a question is logically derivable from the assumptions. Answer True, False or Unknown, without any additional explana-

tions or information. Assume only the following: 'Big elephant's head is green. John is a big elephant. John has a head'. Based on only this knowledge, is it true, false or unknown that John has a green head?".

Of the 982 questions built by us (*core*) , GPT3 answers "correctly" 743, while GPT4 answers "correctly" 804. However, the "incorrect" answers often cannot be considered strictly incorrect, since some of the problems can be interpreted in slightly different ways and some of the world knowledge may creep in. Examples of clear cases of errors for GPT4: answering *true* instead of *false* to the question "Crenkils are not birds. John is a crenkil. John is a bird?", *false* instead of *unknown* to "John had a car Eve liked. Eve had a car?", *true* instead of *unknown* to "John saw the head of the elephant. John saw the tail of the elephant?". On the other hand, cases like answering *unknown* instead of *true* to "The big mouse is strong. The mouse is a big mouse?" and answering *false* instead of *unknown* to "John saw a head of an elephant. John saw a head of the bear?" are debatable.

For the 113 unmodified problems from the *hans* test set, GPT3 answers "correctly" 86, while GPT4 answers "correctly" 100. An example of a GPT4 failure is answering *false* instead of *unknown* to "The lawyers were recommended by the doctor . The lawyers recommended the doctor ?".

For the 38 selected problems from the *allen* set, inspired by the ProofWriter examples, GPT3 answers "correctly" 27, while GPT4 answers "correctly" 32. These problems are longer and require somewhat deeper reasoning than the examples described before. They are also less ambivalent, due to the lack of focus on the nuances of parsing.

Of the 90 problems from the *wiki* set, GPT3 answers "correctly" 79, while GPT4 answers "correctly" 88.

4 Two Paths of Incorporating LLMs

We consider two different paths for incorporating LLMs into the parsing component and conduct small experiments for evaluating their feasibility. The system prompts mentioned below along with the test sets and the results mentioned earlier are available from Github [4].

4.1 Direct Parsing by LLM

The first path is giving LLM the task of converting the whole input – both the assumptions and the question – to FOL. This raises the question of the syntax and the encoding style. We use the system prompt of the API to give GPT4 the necessary instructions. In our experiments the simplest system prompts like (presented partially) "You are a semantic parser from English to first order predicate logic (FOL). Convert input sentences to logic, represented in json using lists for predicates and formulas like this: ["forall","X", ...] for universal quantification of a variable X, ... etc ... ["question" ...] for wrapping a logic of the question

[4] https://github.com/tammet/nlpsolver/tree/main/gpt

with a yes/no answer" typically produce unusable and conceptually incorrect results, with a botched selection of quantifiers. As a simple example, the problem "Mary's car is clean. Mary has a car?" is converted to `["and",["clean", "car_of_Mary"],["question", ["exists", "X", ["and", ["car", "X"], ["owns", "Mary", "X"]]]]]`.

After a number of experiments with GPT4 we have built a reasonably usable system prompt, containing almost a hundred examples of converting sentences and full problems to FOL, using the encoding style which is relatively natural, yet suitable for our goals of capturing the details of meaning. Consequently, we have built a converter from the output to the actual encoding style used in our world knowledge rule systems and usable by the solver. Here is one example:

```
   "Dinosaurs had big heads" =>
["forall","X", ["implies",["isa","dinosaur","X"],
["exists","Y",["and",["isa","head","Y"],
["has property","Y","big"],["had","X","Y"]]]]].
```

The examples indicate both the intended structure and style, the suitable predicates and most importantly, the *coreference* handling and the proper selection of quantifiers (the example above). We have also incorporated wikification of proper names, while keeping situations, probabilities and exceptions out of the examples, to be handled separately. The size of the prompt is ca 14 kilobytes. We suspect that if we were to incorporate too many aspects, like the handling of situations, probabilities and exceptions, the prompt would become unwieldy.

Our experiments with this large prompt are mostly positive for short sequences of simple sentences and single complex sentences, while not so positive for sequences of complex sentences. For example, the previous example "Mary's car is clean. Mary has a car?" is correctly converted. to

```
["and", ["exists","X", ["and", ["isa","car","X"],
["has","Mary","X"], ["has property","X","clean"]]],
["question",
["exists","Y", ["and", ["isa","car","Y"], ["has","Mary","Y"]]]]].
```

Similarly, the example "The presidents paid the scientists. The presidents paid?" from HANS which was answered incorrectly by GPT4, is converted correctly and proved to be *true* by our reasoner.

This said, the quality of parsing simple assumption/question pairs is not guaranteed. For exploring the quality, we use the same four test sets described before. The sets are geared towards investigating the quality of parsing, while the reasoning tasks are trivial. The following table is a summary of testing each of these four sets. The *size* column gives the number of problems in the set and the *llm* column gives the percentage of erroneous answers when we evoke GPT-4o directly on the problem, without parsing. The *json err 1* column gives the percentage of LLM parsing results with a syntactically incorrect json result. We wrote a small program to correct four of the most common syntax errors with an obvious fix. The *json err 2* column gives the percentage of json syntax errors which did not have an obvious fix. Finally we ran our reasoner on the subset of the parsing results which were either syntactically correct or which had an

obvious fix. The final column shows the percentage of errors found by proving the problem with our reasoner. Since the reasoning tasks were easy, errors indicate semantically incorrect parsing. Since an expected answer for each problem in the set is either *True*, *False* or *Unknown*, a random answer would err for roughly 66% of the problems.

problem set	set size	llm err	json err 1	json err 2	prove err
hans	113	15%	8%	4%	12%
core	982	18%	19%	2%	28%
allen	38	18%	34%	18%	42%
wiki	90	2%	40%	10%	49%

Let us consider the types of parsing errors. To start, the rows in the table are in the rough order of the textual length of the problems, thus we can see that the errors made both in the json syntax and the logical representation cumulate as the length of output grows.

A large subset of undisputable errors is caused by choosing a wrong scope of quantifiers. For example, the sentences like "The price of the car is 3 dollars. The bike has the same price as the car." are often transformed to two separate formulas with separate existential quantifiers for the *car*, thus breaking the connection between the *car* in the first and second sentences.

Sometimes important details are missed. For example, "The hand of a man moved a wheel" was otherwise parsed correctly, but the fact that the hand belongs to *the man* (or is in any way associated with *the man*) was missing.

GPT4 employs a fair amount of creativity in deciding which phrases to split or not to split. For example, one of the results of parsing "Karivere is the capital and most populous city of Trapania" is `["rel2", "most populous city", "Karivere", "Trapania"]`, from which it is impossible to logically deduct that Karivere is a city. Similarly, "Situated on a bay in north Trapania, on the shore of the Gulf of Finland ..." gives us `["rel2", "in", "X", "north Trapania"]` and `["rel2", "on the shore of", "Karivere", "Gulf of Finland"]` as parts of the answer, from which it is again impossible to directly logically deduct that X is in Trapania or that Karivere is *on the shore*.

The problem with longer phrases being kept intact is hard to solve. Clearly, a fine-grained representation may be highly complex, and it may require an unwieldy amount of directions and examples in the prompt to reach a suitable result of parsing. On the other hand, simpler facts, like being a city, may be relatively easy to extract from phrases like "most populous city".

Our large prompt recommends a use of a small number of wrapper predicates like *rel2*, *isa*, *has attitude* and actually uses 29 wrapper predicates (*has*, *has location*, *is able*, $=$, etc) in the examples. The GPT4 output contains ca forty new wrapper predicates not included in the example, like *has beneficiary*, *has intermediary*, *has means*, *knew*, *pay*, *coextensive with*, *somewhat*, *very* etc. Although the list of logical connectives has been concretely given in the prompt, GPT4 invents a few more, like *most*. Another relatively common pattern is using predicates without a wrapper, like `["jet engine", "Y"]` instead of `["isa", "jet engine", "Y"]`.

It would not be correct to consider such invented predicates as clear "parsing errors". Many of these make good intuitive sense. However, since reasoning requires using a large set of pre-built rules in the world knowledge database, the unexpected appearance of new invented predicates inevitably creates a mismatch with the existing knowledge bases.

It is also possible that at least some of the errors described could be avoided or mitigated by using a larger and more complex prompt: this requires further investigations.

We have performed a few initial experiments with the Claude 3.5 Sonnet model from Anthropic: these indicate a surprisingly similar behaviour to that of GPT4, with errors very similar to the ones described above.

Our current conclusion is that parsing the whole problem with GPT4 or Claude 3.5 Sonnet and a suitable large prompt is reasonable in case the problem is relatively small. For larger problems it would rather make sense to parse the sentences or short paragraphs separately. This, however, requires the use of a coreference solver and other additional components, in order to connect the parsed sentences. For nontrivial sentences it is also necessary to employ a significant amount of complex post-processing to further split long phrases and normalize invented predicates and quantifiers to the signature used in the background knowledge base. Also, a certain amount of randomly occurring errors is to be expected.

4.2 Using an LLM for Specific Subtasks

The second path is giving LLM a number of smaller specific subtasks in the parsing process. Our initial experiments show that small subtasks are solved by GPT with a higher quality and more consistently, than the task of parsing everything at once. The subtasks we consider are notoriously hard – for some, practically impossible – to conduct with high quality using pure heuristics, without substantial help from machine learning.

Text simplification. Several semantic parsers, including ours, use a set of rules for text simplification. However, it is practically hopeless to write down a sufficient set of simplification rules by hand. LLMs are conceptually well suited for such a task. Consider the following simple system prompt for GPT4: "You preprocess sentences for a semantic parser to logic. Simplify,maximally shorten and split the sentence to shortest possible separate subsentences, to make it understandable for children.". Applied to the complex Wikipedia sentence about penguins "Penguins are a group of aquatic flightless birds from the family Spheniscidae of the order Sphenisciformes" we get "Penguins are birds. Penguins cannot fly. Penguins live in water. Penguins belong to the family Spheniscidae.". Similarly, the complex Wikipedia sentence "Situated on a bay in north Estonia, on the shore of the Gulf of Finland of the Baltic Sea, Tallinn has a population of about 461,000 and administratively lies in the Harju maakond (county)" is converted to "Tallinn is a city in north Estonia. It is by the Gulf of Finland. Tallinn has about 461,000 people. It is in Harju county.". Parsing these sentences to FOL is already in reach of semantic parsers. In cases where some of the output

sentences are still too complex, the same simplification operation can be applied to them separately.

Coreference resolution. This is a classic hard problem in A.I. However, LLMs are known to solve this problem with a very high quality. Our experiments with a moderately complex system prompt show good results with some instructions followed by a set of ca 30 examples like "An elephant was strong. The animal lifted a stone. =>An elephant was strong. The animal {An elephant} lifted a stone. ". and "If animal or bird is nice and simple, it is cute. John is cute? =>If animal or bird is nice and simple, it {animal or bird} is cute. John is cute?".

Quantifier selection. Again, this is a conceptually hard problem: selecting a universal or existential quantifier for a variable and determining the proper scope of quantifiers. Our experimental prompt contains a few lines of instructions, followed by ca 30 examples like "Bears like honey =>Bears * like honey *", "Americans have a capital =>"Americans * have a capital +" "Bears had a tail =>Bears + had a tail $(bears)".

Wikification. Wikification means replacing a name in the text with a globally unique identifier, typically a Wikipedia URL. Globally unique identifiers are crucial for composing separately parsed texts into a coherent knowledge base. Since this task appears to be fairly common and relatively easy for LLMs, we have achieved good results with a system prompt containing two lines of instruction and ca 10 obvious examples.

5 Summary and Future Work

Our first observation is that the behaviour of the current versions of GPT for strict logical reasoning even for very small natural language inference tasks is somewhat unpredictable.

The second observation is that applying GPT4 with complex multi-shot prompts to perform semantic parsing to a usable detailed FOL representation is possible, and is likely more productive than writing large amounts of heuristics for a non-learning parser. At the same time, multisentence text including questions should be rather parsed as separate sentences, with additional parser components performing coreference resolution and other steps to glue the results of parsed sentences into a coherent whole.

The third observation is that the arguably hardest subtasks of semantic parsing can be performed with a help of GPT4 with a seemingly higher reliability and quality than direct parsing with LLM. This would probably make it possible to compose a high-quality parser by using these components as a part of a conventional parser, either partially hand-coded or fully learned.

The experiments described in the paper were conducted with a goal of researching alternative paths of using LLMs for semantic parsing. We provide several experimental system prompts and test sets geared towards parsing English to FOL. We are planning to use our observations for enhancing our current pipeline with LLM-based components, after which it would be possible to experiment with different question answering benchmarks.

References

1. Daniel, K.: Thinking, fast and slow. macmillan (2011)
2. De Marneffe, M.C., Manning, C.D., Nivre, J., Zeman, D.: Universal dependencies. Computational linguistics **47**(2), 255–308 (2021)
3. Dziri, N., Lu, X., Sclar, M., Li, X.L., Jiang, L., Lin, B.Y., Welleck, S., West, P., Bhagavatula, C., Le Bras, R., et al.: Faith and fate: Limits of transformers on compositionality. Advances in Neural Information Processing Systems **36** (2024)
4. Gupta, G., Zeng, Y., Rajasekaran, A., Padalkar, P., Kimbrell, K., Basu, K., Shakerin, F., Salazar, E., Arias, J.: Building intelligent systems by combining machine learning and automated commonsense reasoning. In: Proceedings of the AAAI Symposium Series. vol. 2, pp. 272–276 (2023)
5. Järv, P., Tammet, T., Verrev, M., Draheim., D.: Knowledge integration for commonsense reasoning with default logic. In: Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KEOD. pp. 148–155. INSTICC, SciTePress (2022)
6. Lehmann, J., Gattogi, P., Bhandiwad, D., Ferré, S., Vahdati, S.: Language models as controlled natural language semantic parsers for knowledge graph question answering. In: ECAI 2023, pp. 1348–1356. IOS Press (2023)
7. Lyu, Q., Havaldar, S., Stein, A., Zhang, L., Rao, D., Wong, E., Apidianaki, M., Callison-Burch, C.: Faithful chain-of-thought reasoning. In: Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 305–329 (2023)
8. McCoy, T., Pavlick, E., Linzen, T.: Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 3428–3448. Association for Computational Linguistics (2019)
9. McGinness, L., Baumgartner, P.: Automated theorem provers help improve large language model reasoning. EPiC Series in Computing **100**, 51–69 (2024)
10. Olausson, T., Gu, A., Lipkin, B., Zhang, C., Solar-Lezama, A., Tenenbaum, J., Levy, R.: Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 5153–5176 (2023)
11. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: A python natural language processing toolkit for many human languages. CoRR abs/2003.07082 (2020), https://arxiv.org/abs/2003.07082
12. Qiao, S., Ou, Y., Zhang, N., Chen, X., Yao, Y., Deng, S., Tan, C., Huang, F., Chen, H.: Reasoning with language model prompting: A survey. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 5368–5393 (2023)
13. Reiter, R.: A logic for default reasoning. Artificial Intelligence **13**(1–2), 81–132 (1980)
14. Tafjord, O., Mishra, B.D., Clark, P.: Proofwriter: Generating implications, proofs, and abductive statements over natural language. arXiv preprint arXiv:2012.13048 (2020)
15. Tammet, T., Draheim, D., Järv, P.: Gk: Implementing full first order default logic for commonsense reasoning (system description). In: Blanchette, J., Kovács, L., Pattinson, D. (eds.) IJCAR 2022: Automated Reasoning. LNCS, vol. 13385, pp. 300–309. Springer (2022)

16. Tammet, T., Järv, P., Draheim, D.: Confidences for commonsense reasoning. In: Platzer A., S.G. (ed.) Automated Deduction – CADE 28. CADE 2021. LNCS, vol. 12699, pp. 507–524. Springer (2021)
17. Tammet, T., Järv, P., Verrev, M., Draheim, D.: An experimental pipeline for automated reasoning in natural language (short paper). In: International Conference on Automated Deduction. pp. 509–521. Springer (2023)
18. Tammet, T., Sutcliffe, G.: Combining json-ld with first order logic. In: 15th International Conference on Semantic Computing (ICSC). pp. 256–261. IEEE (2021)
19. Trinh, T.H., Wu, Y., Le, Q.V., He, H., Luong, T.: Solving olympiad geometry without human demonstrations. Nature **625**(7995), 476–482 (2024)
20. Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., Kambhampati, S.: Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. Advances in Neural Information Processing Systems **36** (2024)