NORTHWESTERN UNIVERSITY

Reasoning and Structured Explanations in Natural Language via Analogical and Neural Learning

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Danilo Neves Ribeiro

EVANSTON, ILLINOIS

September 2023

Abstract

Performing complex reasoning has been a long-standing challenge in artificial intelligence (AI). This thesis describes a class of AI systems designed to reason, extract knowledge, and answer questions on various domains such as process understanding, elementary science, and math word problems. Our approach differs from traditional logical reasoning systems since we work directly over the natural language description of problems, thus bypassing the need to manually create formal representations. The proposed systems rely on two architectures: the *Companion Cognitive Architecture* and *Transformer Language Models*. At their core, these architectures use *analogical* and *neural* learning to reason and extract patterns from the input text. Experiments on multiple language tasks show that our methods outperform strong baselines and can make predictions from just a few training examples.

In addition, we study how such AI systems can generate chains of reasoning that explain how known facts can be used to reach conclusions and answer questions. Called *structured explanations*, these chains of reasoning contain multi-premise textual entailments, where intermediate conclusions are used by subsequent entailment steps. This reasoning and explanation approach differs from existing work on natural language inference (Camburu, Rocktäschel, Lukasiewicz, & Blunsom, 2018) or multi-hop question answering (Yang, et al., 2018), where reasoning is either single-step or single-premise. In particular, these structured explanations are shown to alleviate opaqueness issues of neural language models and have the potential to help humans validate and gain more trust in the output of these systems.

Acknowledgments

I have been deeply fortunate to receive support from many collaborators and institutions. Financially, this research was supported in part by the Machine Learning, Reasoning, and Intelligence Program of the Office of Naval Research (Grant No. N000142012447). I'm also thankful for the enterprises that supported my research through internships, namely Adobe Inc., Amazon.com Inc., and Meta Platforms Inc. In particular, the research done during the Summer of 2022 and 2023 at the Amazon Web Services AI Labs proved to be immensely fruitful, with results that ended up included in this dissertation.

I would like to express my sincere gratitude to my advisor Ken Forbus, for his unwavering support and guidance throughout my doctoral research at Northwestern. Over the last five years, Ken has been a major source of inspiration, consistently providing me with insightful ideas that have ultimately shaped the very core of this thesis. He not only encouraged me to think critically and independently, but also instilled in me a broader perspective, reminding me that research trends may come and go, but what truly matters is the ability to see the big picture.

I am thankful to my thesis committee members, Lawrence Birnbaum, Douglas Downey, and Han Liu. Not only did they provide insightful feedback to improve my thesis but were also amazing educators. I very much enjoyed taking their AI classes at Northwestern. Furthermore, I would like to thank my research internship mentors Chris Tensmeyer, Shen Wang, and Jack Goetz. Research internships can be fairly fast paced, requiring quick planning and execution. They provided guidance and played an instrumental role in shaping the research projects and experiments that were carried out. I would also like to thank important industry collaborators. Maria Chang, for the Step Semantic ideas and examples. Xiaofei Ma, Rui Dong, Henry Zhu, and

William Wang, for their contributions to the reasoning and explanation projects. Juliette Burger, Anjelica Ramos, and Amazon's annotation team for their work on the STREET dataset.

When I embarked on the academic odyssey known as PhD, I had a deep desire to explore the frontiers of knowledge and contribute to the field of Artificial Intelligence. Little did I know that I would also be blessed with relationships that would extend beyond my academic life. I am extremely grateful to have worked, despaired, and laughed alongside my Northwestern colleagues Tom Hinrichs, Madeline Usher, Jason Wilson, Joseph Blass, Maxwell Crouse, Irina Rabkina, Kezhen Chen, Constantine Nakos, William Hancock, Samuel Hill, Victor Bursztyn, Taylor Olson, Mukundan Kuthalam, Wangcheng Xu, Walker Demel, and Roberto Salas. Many of them were co-authors, mentors, and even groomsmen (thank you Constantine and Victor). Your company made the PhD experience worth it. Special thank you to Tom who guided me through the perils of understanding and improving the language system, as well as to Max for his guidance on analogical training.

This work would not have been possible without the support of my family and friends. Particular gratitude goes to my parents, Renilde Ribeiro and Rivaldo Ribeiro, they were my role models and taught me the value of education, perseverance, and hard work. To my loving wife, Felicity Lu-Hill, for staying by my side in joy and in sorrow, in sickness and in health, not to mention all the time she spent proofreading my work. To my parents-in-law Priscilla Lu and Dwight Hill, for encouraging me to leave my comfort zone and take risks, without them I would likely not have joined the PhD program at all. Last but not least, to my son Caio LuHill-Ribeiro, for giving me an endless amount of joy and purpose in life.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

A long-term pursuit in *Artificial Intelligence* (AI) is to endow machines with the ability to reason and manipulate knowledge to solve tasks. Initially, some AI systems performed reasoning over symbolic knowledge, applying logical and probabilistic rules to reach conclusions (Newell & Simon, 1956; McCarthy, 1960; Siler & Buckley, 2005). These methods enabled the creation of systems that were successfully applied to solve various tasks (Buchanan & Smith, 1988; Susskind, 1987). Such symbolic manipulation systems are capable of executing multiple reasoning steps, generating a formal proof that can explain their conclusions (Forbus & De Kleer, Building problem solvers, 1993). Even with their initial success, reasoning with pure rule-based symbolic systems can be challenging when it comes to handling ambiguous concepts or learning from human modalities. Some symbolic systems are carefully tailored to solve a certain task (Musen & Van der Lei, 1988), and generalization to other tasks can be hard to achieve.

To mitigate these issues, some AI approaches can work directly over natural language descriptions of the problems. These approaches must deal with many challenges when manipulating human language including ambiguity, lack of structure, and tacit knowledge or rules (Khurana, Koli, Khatter, & Singh, 2023). In this thesis, we explore two AI architectures that can reason while directly handling natural language input, namely: the ***Companion Cognitive Architecture*** and ***Transformer Language Models***. Each of those approaches has its advantages and limitations when it comes to data-efficiency, generalization, and explainability of their outputs (Jain & Wallace, 2019; Strubell, Ganesh, & McCallum, 2020). For instance, neural networks applied to language understanding have been shown to learn from large amounts of data (Devlin, Chang, Lee, & Toutanova, 2019), but the explainability of their internal computations remains

elusive (Hanif, Zhang, & Wood, 2021). We not only run experiments with these two AI architectures separately, but we attempt to have the best of both worlds by combining these two architectures into a single language system. Thus, we propose algorithms, techniques, and modeling strategies that can be used to solve complex (often multi-step) reasoning tasks over natural language, while also systematically studying the ability of such systems to explain their answers. More specifically, we work on tasks involving *process understanding*, data-efficient *knowledge extraction*, and analytical reasoning for *question-answering*.

To mitigate the explainability issues of neural models, we train these models to output human-interpretable explanations. We show how our proposed structured explanations are desirable over other approaches as they contain multi-premise entailment steps that indicate how an answer follows from a given set of premises. We evaluate our proposed methods not only on standard question-answering metrics but also on the accuracy of their generated structured explanations. In our work, we focus mostly on generating an explanation (or line of reasoning) rather than the pragmatics of how to present the explanation to the end user. For the most part, we use annotated golden data to compare both the individual reasoning steps and the overall structure of the predicted output.

1.1 Claims and Contributions

The claims of the thesis are as follows:

1. **[Claim A] Discrete actions and processes in natural language can be accurately modeled using qualitative reasoning and structured action representations.** Our proposed framework can be used to represent various discrete processes (e.g., science phenomena such as photosynthesis or daily chores such as cooking). The representation is

independent of a particular task or domain and can facilitate reasoning in downstream tasks. Such a framework can be applied to solve real-world natural language problems containing procedural texts.

2. **[Claim B] Analogical learning can be successfully applied to solve complex reasoning tasks in natural language, while being explainable and data efficient.** Analogical Learning can solve question-answering tasks involving procedural text understanding, where the system has to track entity states over discrete time intervals. Analogical Learning is also shown to perform knowledge extraction of general-purpose facts from natural language text. In this case, analogical learning also benefits from integration with neural network language models for semantic disambiguation. One can easily inspect the query-cases and analogical matches used to solve such problems.
3. **[Claim C] Explainability issues of neural models can be mitigated by training such models to output a step-by-step structured explanation of their predictions.** While not exposing the true underlying computation done by the neural network, the step-by-step explanation can help humans validate the answer and gain more trust on models that are otherwise viewed as black box systems.

The contributions of the thesis are as follows:

1. **Created a framework (namely *Step Semantics*) for modeling discrete actions and events that occur in both physical and abstract processes.** It combines the existing Cyc and FrameNet ontologies to describe events, identifying relevant role-relations that can be used to represent properties of events such as participating concepts, locations, and duration of events.

2. **Designed and implemented systems and algorithms that use Qualitative Representations and Analogical Learning to solve natural language problems.** In the first task of procedural text understanding, we use *Analogical QA training* to predict changes in the state of participating concepts mentioned in text describing some process. The system applies common sense constraints to obtain a consistent sequence of events using dynamic programming to globally optimize scores. In the second task of Knowledge Extraction, we use distant supervision to automatically generate sentence and fact pairs as training examples. The system predicts new facts by using a mix of Analogical QA training and a set of heuristics for semantic selection. In addition, we use an encoder-only pre-trained language model trained on FrameNet and VerbNet frames to generate word sense disambiguation scores as well as scoring new extracted facts.
3. **Performed experiments showing the data-efficiency of Analogical Learning when compared to existing state of the art models in natural language understanding tasks.** In the procedural text understanding experiments, we show that analogical learning can perform inference and identify state changes in procedural text from a few training examples. In the knowledge extraction experiments we leverage the power of analogical learning and use the existing structure in a knowledge graph to show how new relations can be learned with distant supervision from just a handful of examples.
4. **Developed explanation methods using neural language models while improving existing datasets with structured natural language explanations.** Our proposed iterative retrieval and generation architecture (IRGR) can help language models fetch premises that are required to build structured explanations from a textual knowledge corpus. Furthermore, we create a new explanation and reasoning benchmark (STREET) that

contains challenging tasks and requires systems to not only answer questions, but also explain their answers systematically.

1.2 Organization

The primary contributions of this thesis are contained in Chapters 4, 5, 6 and 7. The overall structure of this document is as follows. Chapter 2 provides a literature review. Chapter 3 discusses the background needed to understand the techniques and methods used. This includes a detailed description of the Companion cognitive architecture, analogical learning, the Transformer architecture, pre-trained language models, and dense retrieval. Chapter 4 introduces our analogical approach on procedural text understanding. We show how Step Semantics can be used to model discreate actions in processes. Then, individual state changes in the procedural text are predicted using analogical learning and combined using a dynamic programming algorithm that enforces commonsense constraints. Chapter 5 describes how analogical and neural learning can be combined to extract structured knowledge from unstructured texts. We show that our method can extract high precision facts from a few distantly supervised examples. Chapter 6 and Chapter 7 explore how to improve explainability of neural language models by generating explanation structures containing multiple reasoning steps that show how premises can entail conclusions. We also describe our proposed dataset called STREET, which contains multi-step reasoning explanation on a varied set of question-answering tasks. Chapter 8 summarizes the contributions and propose avenues for future work.

1.3 Summary of Experiments

The following table contains a summarization of the experiments performed, including data and metrics used, and how they are connected to our claims.

Experiment #1 [Claim A & B]	
Approach	Predicting State Changes in Procedural Text (Chapter 4)
Data	ProPara dataset (Dalvi, Huang, Tandon, Yih, & Clark, 2018)
Description	For each paragraph containing entities and how the entity state evolves over time, there are a set of questions of different granularities designed to test if the system can track the changes in state.
Metrics	Sentence-level questions accuracy: • Cat-1: Is p created (destroyed, moved) in the process? • Cat-2: When is p created (destroyed, moved)? • Cat-3: Where is p created (destroyed, moved from/to)? Paragraph level questions, with combined Precision, Recall and F1 score: • Q1: What are the inputs to the process? • Q2: What are the outputs of the process? • Q3: What conversions occur, when and where? • Q4: What movements occur, when and where?
Experiment #2 [Claim B]	
Approach	Data-Efficient Learning via Analogical Knowledge Extraction (Chapter 5)
Data	Simple English Wikipedia as text corpus and NextKB and knowledge source

Description	We extract knowledge from a subset of the Simple English Wikipedia articles containing 2,679 articles. We perform manual evaluation of the extracted facts.
Metrics	We compare the results with baselines using the following metrics: • Estimated precision of extracted knowledge. • Number of extracted facts.
Experiment #3 [Claim C]	
Approach	Iterative Retrieval and Generation (Chapter 6)
Data	EntailmentBank (Dalvi, et al., 2021)
Description	The evaluation consists of comparing the generated entailment tree and the golden entailment trees. The evaluation metrics will measure the correctness of the generated proof along four dimensions.
Metrics	The metrics used for the four dimensions are as follows: • Leaf (F1, All-Correct): tests if the predicted and golden tree match, ignoring ordering. • Steps (F1, All-Correct): tests if the predicted entailment tree follows the correct structure. Uses an alignment algorithm based on Jaccard Similarity to assign a one-to-one mapping among nodes. • Intermediates (F1, All-Correct): tests if the generated sentences (intermediate node labels) are correct. Given that two sentences

	were matched by the alignment algorithm, the F1 score is computed by using a BERT-based textual similarity score. • Overall (All-Correct): Tests all previous metrics together.
Experiment #4 [Claim C]	
Approach	Natural Language Reasoning Graphs (Chapter 7)
Data	Uses our proposed STREET dataset, which builds upon the following datasets: • ARC (Clark, et al., 2018) • SCONE (Long, Pasupat, & Liang, 2016) • GSM8K (Cobbe, et al., 2021) • AQUA-RAT (Ling, Yogatama, Dyer, & Blunsom, 2017) • AR-LSAT (Zhong, et al., 2022)
Description	The experiments not only test the model's ability to answer questions, but also explicitly evaluate their step-by-step reasoning explanation.
Metrics	We propose the following metrics: • Answer Accuracy: measures the ability of models to predict the correct answer to a question. • Reasoning Graph Accuracy: compares the predicted and golden reasoning graphs in terms of graph structure and intermediate conclusion nodes. We use a textual similarity function to test if two reasoning step nodes are equivalent.

	• Reasoning Graph Similarity: is a “softer” metric than the previous one, as it compares the predicted and golden reasoning graph using a graph edit distance function. The graph edit distance is normalized to obtain a similarity score.

2 Related Work

We summarize the existing literature on how cognitive architectures and neural models have been applied to perform complex reasoning over natural language, while summarizing what has been done to make such systems more explainable and trustworthy. In addition, we elaborate on the methods used to solve the natural language tasks discussed in this work.

2.1 Reasoning over Natural Language

The development of AI systems capable of performing reasoning and knowledge manipulation to solve tasks expressed in natural language has been widely studied. When testing natural language capabilities, several QA datasets were proposed including SQuAD (Rajpurkar, Zhang, Lopyrev, & Liang, 2016), ARC (Clark, et al., 2018) and OBQA (Mihaylov, Clark, Khot, & Sabharwal, 2018). Unlike natural language tasks such as dialogue, translation, or summarization; question answering (or QA) tasks are often easier to evaluate. Many QA datasets became popular within the NLP community and were accepted as relevant reasoning benchmarks. Hence, we largely use QA datasets to evaluate our systems.

The techniques used to solve these QA datasets served as important milestones towards enabling complex natural language reasoning. Despite this initial success, these models had many shortcomings. For instance, stochastic models were able to exploit inherent biases in QA datasets (Gururangan, et al., 2018; Ko, Lee, Kim, Kim, & Kang, 2020), raising questions regarding the true quality of their results. Explainability of the answers are also elusive, and often systems would be able to answer the posed questions, but it was not clear why an answer was chosen over the others.

2.1.1 Multi-step Reasoning over Natural Language

As one of the simpler forms of reasoning, *Natural Language Inference* (NLI) is the task of predicting the relationship between sentences (or short texts) as either "entailment", "contradiction" or "neutral" (Bowman, Angeli, Potts, & Manning, 2015; Zellers, Bisk, Schwartz, & Choi, 2018). NLI was an important step towards testing reasoning in natural language, but this task often checks for the plausibility of a sequence with no explicit multi-step entailments required.

To address the multi-step aspect of reasoning, some initial research has been done on datasets that contain mostly synthetic texts describing some templated domain (Weston, et al., 2015; Sinha, Sodhani, Dong, Pineau, & Hamilton, 2019; Clark, Tafjord, & Richardson, 2021). These tasks had the advantage of being self-contained and were annotated with an explicit solution that goes from premises to conclusion. ProofWriter (Tafjord, Dalvi, & Clark, 2021) is a generative language model that takes as input a set of textual premises and outputs the step-by-step proof that answers a given question. The ProofWriter model shares some similarities to our proposed IRGR model (Chapter 6), however it does not perform retrieval of premises and has a fixed input context.

On the other hand, *Multi-Hop QA* (Yang, et al., 2018; Khashabi, Chaturvedi, Roth, Upadhyay, & Roth, 2018) rely on more fluid texts, often using real-world examples. The notion of "multiple hops" can be somewhat abstract and a variety of tasks can be categorized as multi-hop reasoning. Here we assume that multi-hop implies that one or more intermediate conclusions are used to reach the final answer. One common limitation among MHQA tasks is that they contain a small number of hops (around 2 or 3), and the reasoning chain is mostly linear (Mavi, Jangra, & Jatowt, 2022), which differs from our work where reasoning has multiple steps and contains more complex structures with multiple antecedents per step.

Performing multi-step reasoning over natural language has been accomplished with cognitive systems, with instances both in instruction learning (Veloso & Carbonell, 1993) and in the multimodal domain (Hinrichs & Forbus, 2014; Kirk & Laird, 2016). Notably, Analogical Chaining (Blass & Forbus, 2017) was able to answer common sense reasoning questions by learning *common sense units* from natural language instructions via analogy. It repeatedly uses analogical retrieval to make inferences, exploring alternative outcomes and explanations. The analogical cases are learned through natural language interaction with a person and some learned rules require hand-edits, which differs from our proposed approaches that learn exclusively from training examples with no human interaction.

2.1.2 Process Understanding

Although considerable advances have been made in complex natural language reasoning, understanding texts describing processes is still a major challenge. These include both processes describing continuous changes, such as flows and movement of objects, as well as discrete changes, such as creation or destruction events. Textual description of processes often depicts a world that is constantly evolving. Answering questions about such texts requires tracking how the world state changes over time, and such QA tasks often require multi-step reasoning.

From a modeling perspective, the Qualitative Reasoning community has developed frameworks and proposed formalisms to represent processes. Allen & Hayes (1989) explores a first-order theory of time and intervals, Hogge (1987) described an approach to planning in the physics domain using qualitative descriptions of processes and Drabble (1993) built a hierarchical partial order planner that used qualitative process theory to both create and execute plans involving actions and processes. These approaches differ from our proposed Step Semantics work (Chapter 4), that integrates representations of continuous and discrete representations combining qualitative

process theory (Forbus K. D., 1984) with concepts from FrameNet (Baker, Fillmore, & Lowe, 1998) and OpenCyc (Lenat & Guha, 1989) in an attempt to understand texts in natural language.

There are only a few relevant QA datasets containing questions related to textual description of processes. The SCONE dataset (Long, Pasupat, & Liang, 2016) describes synthetic worlds and how the state changes over time. Conversely, the ProPara dataset (Dalvi, Huang, Tandon, Yih, & Clark, 2018) contains a more natural description of various processes (e.g., science and procedural texts), while also providing annotation for the change in state of the entities (mostly location and existence) mentioned in the context paragraphs. The majority of the approaches used to solve ProPara employed a neural model to predict the state changes on a local level, or tie their global predictions using common sense constraints (Tandon, et al., 2018; Gupta & Durrett, 2019). The work of Clark et al. (2018) used a hybrid approach, integrating semantic role labeling with VerbNet derived rules to map each sentence to its effects on the world state. None of these prior approaches used cognitive modes and analogical learning to solve this QA task, which is the main focus of our proposed system described in Chapter 4.

2.2 Knowledge Extraction and Retrieval

Acquiring, retrieving, and applying general knowledge is essential to perform complex reasoning. For the most part, the methods proposed in this work use either *unstructured knowledge* (e.g., free-form text), *structured knowledge* (e.g., entities and their relations), or *parametric knowledge* (e.g., neural model's own parameters). In this section we briefly summarize the literature on how these different types of knowledge have been obtained and combined to solve natural language tasks.

2.2.1 Knowledge Extraction

As one of the more explicit forms of knowledge representation, structured knowledge has proven to be a useful resource in a variety of artificial intelligence applications (Lukovnikov, Fischer, Lehmann, & Auer, 2017; Tuan, et al., 2022). This knowledge is often stored in *knowledge bases* (KBs) with a pre-defined formalism used to encode facts. Some of the more well-known projects on KB construction include NELL (Carlson, et al., 2010), FreeBase (Bollacker, Evans, Paritosh, Sturge, & Taylor, 2008), and ConceptNet (Speer, Chin, & Havasi, 2017). However, many of these KBs are limited by either: (1) lack of general world knowledge and common nouns, (2) a small set of semantic role relations, (3) representation of entities and relations that use text, which can be ambiguous due to polysemy. One prominent KB that is not limited by these factors is Cyc (Lenat & Guha, 1989; Lenat & Guha, 1990). Cyc is a long-term project that includes concepts and rules representing general world knowledge. Our work described in Chapter 5 uses the open-source version of Cyc, called OpenCyc, which is also an expressive and well-structured KB.

Missing entities and relations can be an issue and limit the usefulness of available KBs. With that in mind, a considerable amount of effort has been made to automatically extract structured knowledge from the more ubiquitous sources of unstructured knowledge. This process is often called Relation Extraction (RE) or Automatic Knowledge Base Completion (AKBC). In our work (Chapter 5) we use the concept of learning new facts with distant supervision, which is a learning approach that does not rely on labeled data and was initially introduced by Mintz et al. (2009). Other methods have focused on using neural networks to perform RE from a few training examples (Han, et al., 2018; Gao, et al., 2019; Qu, Gao, Xhonneux, & Tang, 2020), even though they still required many hundreds of training examples and mostly extract facts about named entities, instead of general knowledge.

Similarly, other systems attempted to extend the contents of a knowledge base by reading texts and building formally represented cases (Forbus, et al., 2007) The learn by reading systems often create cases that contain multiple statements instead of simple triples (Barbella & Forbus, 2010) and may also learn from multiple modalities (Lockwood & Forbus, 2009; Forbus, Lockwood, Sharma, & Tomai, 2009) such as pictures, sketches and diagrams.

2.2.2 Knowledge Retrieval

In order to answer questions, many approaches have used external sources of knowledge, both in structured (Crouse, McFate, & Forbus, 2018; Zhang, Dai, Kozareva, Smola, & Song, 2018; Yasunaga, Ren, Bosselut, Liang, & Leskovec, 2021) or unstructured (Pan, et al., 2019; Guu, Lee, Tung, Pasupat, & Chang, 2020) formats. The work of Lewis et al. (2020) combines a pre-trained retriever with an encoder-decoder language model. Their model performs intertwined steps of retrieval and generation. Other works used previously retrieved passages to improve the retrieval of new passages (Cartuyvels, Spinks, & Moens, 2020; Zhao, Xiong, Boyd-Graber, & Daumé III, 2021; Xiong, et al., 2021). These iterative retrieval-generation methods share some similarities to our proposed system described in Chapter 6, but none of them perform explicit multi-step reasoning and generate explanations to answers.

2.3 Explainability of AI Systems

As AI systems become more pervasive and are used in high-stake applications, it is crucial that humans can understand how they generate predictions. In the context of natural language understanding, many initial AI methods were inherently inspectable such as rule-based systems (Allen J. , 1995), decision trees (Schmid, 2013; Boros, Dumitrescu, & Pipa, 2017) and others. In fact, the cognitive system used in many of our proposed approaches produces outputs where one

can, with some effort, trace the provenance of each decision. On the other hand, many of the current machine learning approaches are considered black box systems (most notably, neural networks) and producing an explanation for their outputs remains a formidable challenge (Hanif, Zhang, & Wood, 2021).

To address this issue, the NLP community came up with multiple notions of explanation and justification in natural language. These include outputting free-form natural language explanations for a given answer (Camburu, Rocktäschel, Lukasiewicz, & Blunsom, 2018; Rajani, McCann, Xiong, & Socher, 2019), using explanation graphs (Jansen, Wainwright, Marmorstein, & Morrison, 2018) or creating reasoning chains in multi-hop that explains an answer (Jhamtani & Clark, 2020). With the advent of very large language models, other explanation approaches such as chain-of-thought prompting (Wei, et al., 2022; Kojima, Gu, Reid, Matsuo, & Iwasawa, 2022; Madaan, et al., 2023) have shown that these models can generate step-by-step explanations (as plain text) that can also help them answer questions more accurately. Unlike these approaches, our work (Chapter 6 and Chapter 7) generates explanations in the form of *entailment trees* (Dalvi, et al., 2021) or *reasoning graphs* (Ribeiro D. N., et al., 2023), which are structured explanations that show explicitly how premises and intermediate conclusions can be combined to explain the predicted answer.

3 Background

3.1 Companion Cognitive Architecture

Our cognitive approaches in Chapter 4 and Chapter 5 were built on the *Companion Cognitive Architecture* (Forbus & Hinrichs, 2006), referred here as "Companion". The Companion architecture is designed to integrate multiple cognitive processes including analogical learning, visual processing capabilities, and language understanding. Similar to other cognitive systems, Companion has the distinguishing characteristic that it is designed to handle multiple human skills and capabilities, instead of focusing on a narrow set of tasks. For instance, Companion has been used for a variety of purposes including learning from analogies (Barbella & Forbus, 2010), understanding multi-modal instructional analogies (Chang, 2016), data-efficient learning for question-answering (Crouse, McFate, & Forbus, 2018), and a deployed information kiosk system (Wilson, et al., 2019).

The architecture contains many components and tools that are designed to bootstrap the development of software as social organisms. There are among these components (1) a powerful reasoning engine called FIRE (Forbus K. D., Hinrichs, De Kleer, & Usher, 2010), which provides analogical learning and reasoning in addition to back-chaining (2) a large-scale knowledge base called NextKB, which integrates various materials from sources including OpenCyc and FrameNet (Baker, Fillmore, & Lowe, 1998) (3) and a general-purpose natural language parser called CNLU, that can convert unstructured input text to structured semantic representations. We briefly describe some of Companion's components in the following sections.

3.1.1 Ontology and Knowledge Base

The structured representations used in Companion can be interpreted as *logical expressions* in predicate calculus. They follow the formalizations defined by the CycL ontology language (Lenat & Guha, 1989). Throughout this thesis we will use lisp-style syntax (nested lists also known as s-expressions) to represent logical assertions. For example, the expression `(properPhysicalPartTypes Fish Gill)` represents the general assertion "*Fish have physical parts called gills*", where concepts `Fish` and `Gills` are arguments of the relation `properPhysicalPartTypes`. In this representation we assume that symbols are case-sensitive.

In CycL a *collection* refers to a kind or type of thing (roughly approximated to a set of concepts) whose instances share a certain property, attribute, or feature. For instance, the term `Fish` refers to all the fishes that ever existed, and will ever exist, and could possibly exist. We can specify that an entity is an instance of that collection with `isa` statements such as `(isa fish3154 Fish)`, where `fish3154` refers to a single entity in that collection. One can express inheritance among collections using the `genls` relation such as `(genls Fish NonHumanAnimal)`, meaning that `Fish` is a subcollection of `NonHumanAnimal`. The knowledge resources in Companion contain a rich hierarchical structure organizing the existing collections.

Predicates can be used to make statements and are another important concept in CycL. Both *relations* and *functions* are kinds of predicates. The terms `properPhysicalPartTypes` and `isa` are relations. Functions are used to denote entities such as `(SeedFn PeanutPlant)`, which refers to the "*seed of peanut plants*" and their symbols are commonly suffixed with `Fn`. All predicates can have a specific number of arguments (called arity) as well as specific argument

types which are represented by `argIsa` structural relations such as `(arg1Isa likes-Generic Agent-Generic)`, meaning that the relation `likes-Generic` expects an instance of `Agent-Generic` as its first argument.

We call the repository of knowledge in Companion a *knowledge base* (or KB), which is a term used to distinguish between both databases and knowledge graphs. The term KB differentiates from databases since it contains general reusable knowledge instead of just specific facts like in "*Mark's year of birth is 1992*". It also differentiates from a knowledge graph since it is not limited to expressing knowledge using triples or binary relations (analogous to an edge in the graph). As mentioned previously, the KB in Companion is called NextKB and is an open-license[1] knowledge base built to support research and development of knowledge-rich systems.

3.1.2 Companion Natural Language Understanding

One key component within Companion is the CNLU language understanding parser (Tomai & Forbus, 2009), which based on Allen's bottom-up parser (Allen J. , 1995) and uses grammatical features and rules to map from the textual input into structured semantic representations in CycL notation. The output of CNLU is task-agnostic, meaning that a semantic interpretation for a certain task can be learned for downstream tasks. The CNLU pipeline consists of many sub-routines that are responsible for breaking the input text into tokens, generating plausible parse trees, and producing a consistent structured semantic representation for each entity introduced in the text. To illustrate this parsing process, we will use the sentence "*Cats drink water*." as a running input example.

[1] Available at https://www.qrg.northwestern.edu/nextkb/index.html

First, the tokenizer identifies the list of tokens from the sentence, e.g., `(cats drink water punc-period)`. It is possible that the same token will appear in different parts of the sentence. For this reason, CNLU will assign a *discourse variable* to uniquely identify each token. For instance, the word "*drink*" will be assigned the discourse variable `drink8304`, while the word "*water*" will be assigned the variable `water8324`. The bottom-up parser will use the entries from its lexicon (McFate & Forbus, 2011), grammar rules, and semantic frames to produce a set of syntactic parse trees representations. The lexicon contains information about parts of speech and other syntactic features associated with words. The semantic frames, many of which were derived from FrameNet, are called *semtrans* statements. These semantic frames specify *word senses* (i.e., the different meanings of a given word) as well as the possible role relations connecting constituents in the input utterance. Each constituent in the parse tree (i.e., span of tokens used in the hierarchical structure) may be assigned a predicate calculus expression representing its semantics.

```
(and (isa drink8304 DrinkingEvent)
     (consumedObject drink8304 water8324)
     (performedBy drink8304 cat8297))
```

Figure 1: Neo-Davidsonian representation for the drinking event.

In CNLU, most of the lexical rules for verbs will represent the semantics using *Neo-Davidsonian representations*. In these representations, the events are reified (i.e., made concrete) and role relations are used to identify the entities that are involved in them. For instance, in the utterance "*cats drink water*" the verb "*drink*" can be represented by the logical form shown in Figure 1. The discourse variable `drink8304` is an instance of `DrinkingEvent`, and the

relations `consumedObject` and `performedBy` describe roles of the entities `water8324` and `cat8297` in this event.

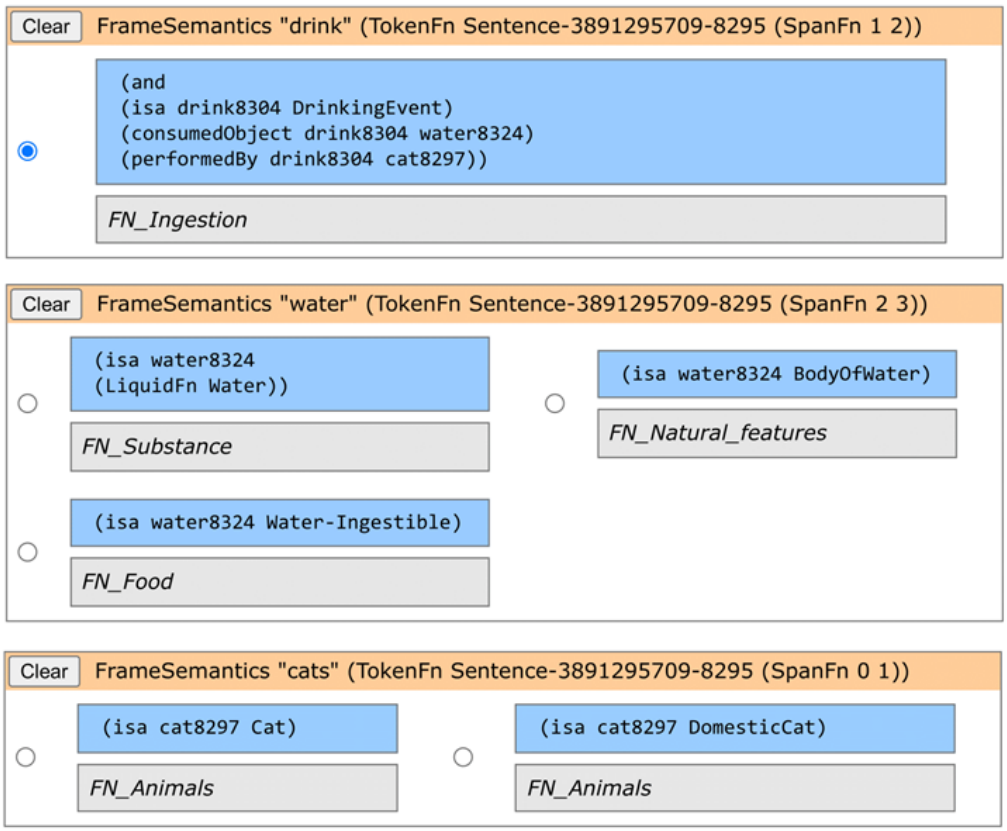

Figure 2: Choice sets generated by CNLU for the sentence "C*ats drink water*.".

Multiple intermediate parse trees can be generated due to different interpretations of the sentence. For instance, the word "*water*" could either be a noun or a verb depending on the context. Furthermore, a given word is likely to have multiple possible interpretation frames due to polysemy and homonymy. To properly represent these ambiguities, CNLU uses distinct *choice sets* of logical statements to represent the different valid semantic frames and parse trees. The choice sets generated for the example sentence as well as the FrameNet frame (e.g., `FN_Ingestion`) associated with them are shown in Figure 2. Each element in the choice set will

be referred as a *choice*, and pairs of choices will be marked as conflicting according to the rules used by the parser. For instance, the pairs of choices that include `(isa cat8297 Cat)` and `(isa cat8297 DomesticCat)` cannot be asserted as true simultaneously. Therefore, making an interpretation of a given input (e.g., word senses) consists of choosing a set of logical statements as correct and ruling out the other remaining conflicting choices.

3.1.3 Analogical Learning

In our cognitive systems we work under the hypothesis that analogical reasoning is central to cognition, and similarity processes play an important role in human learning and how humans compare representations (Gentner & Forbus, 2011). In Companion, the computational implementation of analogy refers to the comparison of two *structured* representations often stored in *cases* containing logical statements. Therefore, analogy differs from other similarity metrics purely based on feature vectors.

The analogical capabilities in Companion are provided by the *Structure Mapping Engine* (Forbus, Ferguson, Lovett, & Gentner, 2017), or SME for short. SME takes as input two relational representations (e.g., the semantic representation of a sentence) and outputs one or more *mappings* between concepts in these representations. These mappings are composed of (1) the correspondences between the concepts and predicates between two structures (2) a score indicating the degree of similarity between the two structures (3) candidate inferences that propose new information that can be derived from the mapping.

To illustrate this process, consider that we are trying to compare the semantic representation of the sentence *A* "*Cats drink water*" with the sentence *B* "*Leaves in plants are eaten by herbivores*". We can see that the word overlaps among *A* and *B* is non-existent, however they are both referring to the same core idea: *consumption of matter by living beings*. The semantic

representation for both sentences are shown in Figure 3, where the disambiguation choices are manually made according to the context.

"*Cats drink water*"	**"*Leaves in plants are eaten by herbivores*"**

`(isa drink8304 DrinkingEvent)`
`(consumedObject drink8304`
`  water8324)`
`(performedBy drink8304 cat8297)`
`(isa water8324`
`  Water-Ingestible)`
`(isa cat8297 DomesticCat)`
	`(isa be8826 Situation)`
	`(performedBy be8826 plant8768)`
	`(isa eat8886 EatingEvent)`
	`(performedBy eat8886`
	`  herbivore9251)`
	`(consumedObject eat8886`
	`  leaf8740)`
	`(doneBy eat8886 herbivore9251)`
	`(objectFoundInLocation leaf8740`
	`  plant8768)`
	`(isa plant8768 Plant)`

Figure 3: Side by side comparison of the manually disambiguated semantic representations for the sentences "*Cats drink water*" and "*Leaves in plants are eaten by herbivores*".

When creating a mapping between the representations for *A* and *B* (where *A* is the *base* and *B* is the *target*) SME will use the constraints defined by *Structure Mapping Theory* (Gentner, 1983). These constraints will guide how pairs of logical statements (e.g., `(isa plant8768 Plant)`) and entities in these expressions (e.g., `plant8768`) should be mapped from base representation to the target representation. These constraints also indicate which mappings should take preference. The theory asserts that matches must follow the following properties:

- **Tiered identicality:** only allows matches between logical statements with identical predicates. For example, the logical forms `(performedBy drink8304 cat8297)` and `(performedBy eat8886 herbivore9251)` could be included as a pair in the

mapping since they share the predicate `performedBy`. If the predicates are semantically similar (meaning that they share a superordinate in the predicate hierarchy) they may also be allowed to be matched.

- **Parallel connectivity:** ensures that two logical statements can only match if their arguments also match. In that instance, `(performedBy drink8304 cat8297)` and `(performedBy eat8886 herbivore9251)` can only be matched if `drink8304` is matched to `eat8886` and if `cat8297` is matched to `herbivore9251`.
- **One-to-one connectivity:** Each element of the base and target can be a part of at most one correspondence. This means that if `cat8297` is matched to `herbivore9251`, then it cannot be matched to some other element such as `plant8768`.
- **Systematicity:** mappings with more deeply nested expressions are given preference over other mappings.

Analogical similarity is widely applicable and it has been used to generate analogical generalizations (McLure, Friedman, & Forbus, 2015) as well as perform retrieval from stored cases in the KB (Forbus, Gentner, & Law, 1995). In general, the problem of finding a mapping that satisfies all the Structure Mapping Theory's properties is known to be NP-Hard (Veale & Keane, 1997). However, SME uses a procedure that produces close to optimal results and runs in $O(n^2 * log(n))$ time, where n is the size of the larger set between target and base representations.

3.1.4 Analogical Question-Answering Training

Given that analogy provides a cognitively plausible and efficient way to compare two structured semantic representations, the goal is to use analogy as a way to facilitate learning from training examples and then making future predictions. The analogical learning in this thesis is inspired by

the approach called *analogical question-answering training* (AQAT) first introduced by Crouse, McFate, & Forbus (2018) and further described by Crouse (2021). This AQAT approach amounts to finding patterns in the input semantic representation that are related to the desired output for a given training example.

More concretely, consider the procedural text understanding problem introduced in Chapter 4 where the objective is to classify if a given input text implies that a certain object of interest has moved. The target prediction can be modeled as an uninstantiated logical expression using Neo-Davidsonian representation: `(and (isa ?object Thing) (isa ?event MovementEvent) (objectMoving ?event ?object))`. If we want the target prediction to include additional information such as where the object of interest moved from, we could also add the logical form `(fromLocation ?event ?object)` to that representation. Therefore, given a training example specifying that the sentence "*roots absorb water from the soil.*" implies that the object of interest "*water*" moved from the "*roots*", we are interested in finding which parts of the semantic representation around `AbsorptionEvent` map to the target representation around `MovementEvent`.

To this end, AQAT creates mappings between source and target representations called *query cases*, which are rule-like constructs that treat the input semantics as antecedents and target logical forms as consequents. During testing, AQAT retrieves the generated query cases using analogical retrieval and combines the best query cases to make a prediction (e.g., answer a question) given a new input semantic representation. One example of a generated query case for this training example is shown in Figure 8. Note that in our case the automatically generated input semantics are task-agnostic (i.e., the outputs of CNLU) while the target logical forms are task-

specific. By including a semantic parser that can be used in various domains, the system can avoid learning from scratch for every new task. This leads to improved performance and data efficiency.

When constructing query cases AQAT needs to select which part of the input semantics justifies which parts of the target logical form. In its core, the query case construction procedure uses SME to create an initial mapping between all the logical forms in the semantic representation and target logical forms. One key problem is that the output of CNLU contains conflicting choices which will render some SME mappings invalid (i.e., the mapping may contain two logical statements that are from conflicting choices in a choice set). To circumvent this issue, AQAT uses a procedure called *structure-aware alignment*, which is a hill-climbing local search algorithm that iteratively improves the initial SME mappings by exploring conflict-free alterations that differ by at most two edges from the current mapping. This algorithm explores the space of distinct mappings by giving priority to alternate mappings according to a given set of scoring functions. The algorithm will keep running until it can no longer find a mapping with a higher score than the previous search step. We further describe the structure-aware alignment algorithm as well as the scoring functions used throughout our experiments in Appendix A.1.

Once query cases are constructed, they are stored in the KB and later retrieved to make predictions. We use MAC/FAC (Forbus, Gentner, & Law, 1995) in order to efficiently retrieve the query cases relevant to an input question. It performs analogical retrieval in two steps, first by performing a cheaper feature-vector match to generate candidates, then uses SME on the most promising candidates to select the best retrieval cases.

3.2 Transformer Language Models

Neural networks have recently become a popular approach in natural language processing. They are capable of leveraging large amounts of data and computation to solve a wide range of language tasks. More specifically, the Transformer architecture (Vaswani, et al., 2017) was shown to learn deep representations from unlabeled texts by predicting tokens and sentences given its context (Devlin, Chang, Lee, & Toutanova, 2019). The Transformer architecture has gradually replaced previously popular recurrent models since they are more suitable for parallelization and can train on larger amounts of training data. The approaches proposed in Chapter 5, Chapter 6 and Chapter 7, use language models either in classification tasks (e.g., word sense disambiguation) or for generative tasks (e.g., generating multi-step explanations).

3.2.1 Neural Network Preliminaries

Here we show the general machine learning recipe for training neural networks. We assume a given *training set* $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{D_{train}}$ with D_{train} data points where $\boldsymbol{x}_i$ is the input vector and $\boldsymbol{y}_i$ is the output or expected vector. These vectors can be used to encode a variety of data formats, but we mostly use these vectors to encode input and output text (i.e., a sequence of symbols or tokens) or classification labels. We also assume the existence of a held-out *development set* (also referred here as evaluation data) of size D_{dev} such that $D_{dev} < D_{train}$ and consists of the examples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{D_{dev}}$. The neural network can be interpreted as a differentiable function $f_{\boldsymbol{\theta}}$ where $\boldsymbol{\theta} \in \mathbb{R}^K$ is the set of model parameters (or neural network weights) that can be applied to make a prediction $\hat{\boldsymbol{y}}_{\boldsymbol{i}} = f_\theta(\boldsymbol{x}_i)$ given some input $\boldsymbol{x}_i$.

The training of neural networks (Rumelhart, Hinton, & Williams, 1986) is done using *gradient descent* by minimizing a *loss function* $\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}_{\boldsymbol{i}}) \in \mathbb{R}$ that intuitively represents the cost (or

penalty) associated with the prediction. These model parameters change over multiple iteration steps t during gradient descent, and the initial model weights $\boldsymbol{\theta}_{(\mathbf{0})}$ are either a vector of zeros or initialized at random. While the loss on the development set has not converged, the stochastic version of gradient descent will select a random sample of the training data $(\boldsymbol{x}_d, \boldsymbol{y}_d)$ and take a step in the direction of the gradient of the loss such that: $\boldsymbol{\theta}_{(\boldsymbol{t+1})} = \boldsymbol{\theta}_{(\boldsymbol{t})} - \eta_t \nabla \mathcal{L}(\boldsymbol{y}_d, f_\theta(\boldsymbol{x}_d))$, where η_t is the learning rate parameter for that gradient step.

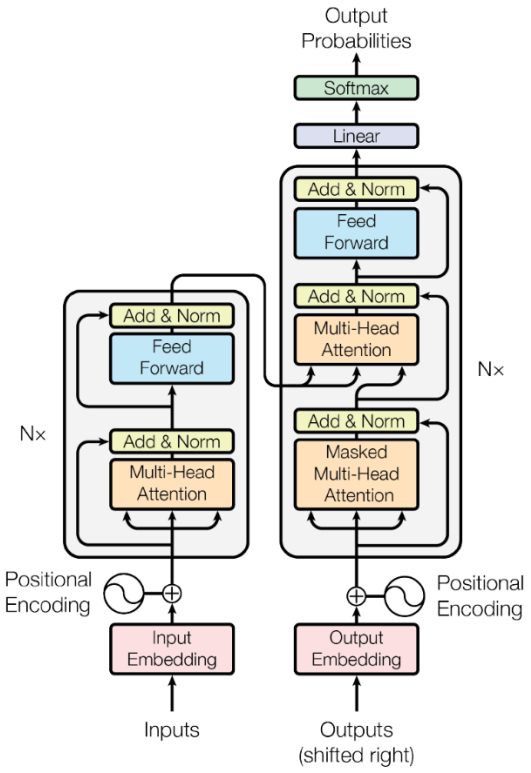

Figure 4: The Transformer architecture. Image taken from Vaswani et al. (2017).

When training these networks, we will normally use a *mini batch* of training examples (a subset with size often between 2 and 1,000) to perform the gradient steps where the gradient is averaged across examples. Other details such as (1) how the gradient steps are computed (i.e., what

optimizer is used), (2) how the learning rate schedule modifies η_t across different t, and (3) the maximum number of total passes over the training data (often called epochs) will vary depending on the task.

3.2.2 The Transformer Architecture

Initially designed for textual sequence modeling such as machine translation, the Transformer architecture (Vaswani, et al., 2017) is an *encoder-decoder* model heavily based on *attention mechanisms*. Given an input of n symbols, it follows the encoder-decoder formulation (Cho, et al., 2014) and maps the input $\boldsymbol{x} = (x_1, \ldots, x_n)$ to an intermediate representation $\boldsymbol{z} = (z_1, \ldots, z_n)$. Afterwards the decoder auto-regressively generates an output sequence $\boldsymbol{y} = (y_1, \ldots, y_m)$ of symbols one element at a time, where generated symbols are used in subsequent generation steps. An overview of the Transformer architecture is shown in Figure 4.

The Transformer's attention mechanism is believed to detect semantic relations between words rather than treating each word simply as tokens in a sequence. For instance, in the sentence "*willows lined the bank by the river*" the semantics of the word "*bank*" is heavily dependent (or weighted) on the context word "*river*". The abstract idea behind the attention revolves around mapping a query $\boldsymbol{q}$ and a set of key-value pairs ($\boldsymbol{k}$ and $\boldsymbol{v}$) to an output. In this case, the query, key, and values are vectors, and the output is a weighted sum of the value vectors. In this section we describe the Transformer's attention mechanism and direct the reader to Vaswani et al. (2017) for further details on the encoder, decoder, and positional encodings.

In the Transformer architecture, the attention mechanism is called *Scaled Dot-Product Attention*. More formally, assuming that the query and key are of dimension d_k and the values are of dimension d_q, and assuming that the vectors are packed together into matrices Q, K and V (for query, key and vector), then the attention output is given by:

$$Attention(\mathrm{Q}, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Where $softmax$ converts a vector of real numbers into a probability distribution such that $softmax(\boldsymbol{a})_i = e^{\boldsymbol{a}_i}/(\sum_{j=1}^{|\boldsymbol{a}|} e^{\boldsymbol{a}_i})$ for $i \in \{1, \dots, |\boldsymbol{a}|\}$ and will serve to compute the weighted sum of V. The scaling factor $\sqrt{d_k}$ is used to help assign a more equally distributed probability among the values of V and prevent small gradients. This attention function is used by the Transformer's *multi-head attention* mechanism that linearly projects the query, key and value h times using separately learned linear projects into dimensions d_k, d_k and d_v, respectively. Assuming that all sub-layers and embedding layers in the model produce outputs of size d_{model} the multi-head attention is given by:

$$MultiHead(\mathrm{Q}, K, V) = \bigoplus_{i=1}^{h} (Attention(QW_i^Q, KW_i^K, VW_i^V))W^O \quad (2)$$

Where $\bigoplus$ is the concatenation operator, and $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$ are parameter matrices. The sizes d_k and d_v are set to d_{model}/h such that the total amount of computation for the multi-head attention is similar to the computation for the single-head attention.

3.2.3 Pre-trained Language Models

The initial *natural language processing* (NLP) methods using neural networks were tailored to solve a single problem and required large amounts of labeled data, where models needed to learn from scratch for every new task (Qiu, et al., 2020). In recent years, the NLP community has shifted the training paradigm towards an approach called *pre-training*, where neural models learn universal language representations from a large corpus of text using *self-supervision*. The early

generations of pre-trained language models aimed to learn *word embeddings* (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013), which learned vector representation for words from a vocabulary. Although word embeddings were shown to be useful in many downstream NLP applications, they were designed to be context-free and often failed to capture the full meaning of a word in a sentence.

To address issues with polysemy and contextual dependencies, the newer generation of pre-trained language models (Peters, et al., 2018; Devlin, Chang, Lee, & Toutanova, 2019; Raffel, et al., 2020) were designed to learn contextual representations of words and became more widely applicable. In particular, Transformer-based pre-trained language models (which we simply refer to as *language models*) were shown to be powerful at learning universal language representations and can better capture long-range dependencies since their self-attention mechanisms can directly represent the relationship between any two words in the input sequence. Different language models may use different pre-training tasks when learning universal language representations. Two popular self-supervised pre-training tasks are called *language modeling* (sometimes called auto-regressive language modeling) and *masked language modeling* which are probabilistic density estimation problems. We discuss these two pre-training approaches below.

Auto-Regressive Language Modeling Pre-Training

Given a sequence of input tokens $\boldsymbol{x} = (x_0, x_1, \ldots, x_{|\boldsymbol{x}|})$ where $\boldsymbol{x}_{i:j} = (x_i, x_{i+1}, \ldots, x_{j-1}, x_j)$ and x_0 is a special token indicating the beginning of the sequence, then the joint probability of the whole sequence $P(\boldsymbol{x}_{1:|\boldsymbol{x}|})$ can be decomposed as:

$$P(\boldsymbol{x}_{1:|\boldsymbol{x}|}) = \prod_{i=1}^{|\boldsymbol{x}|} P(x_i|\boldsymbol{x}_{0:i-1}) \tag{3}$$

Since predicting the conditional probability $P(x_i|\boldsymbol{x}_{0:i-1})$ amounts to computing a probability distribution over the vocabulary of tokens, a language model $f_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ can be used to estimate the probability distribution such that $f_{\boldsymbol{\theta}}(\boldsymbol{x}_{0:|\boldsymbol{x}|}) = P(x_i|\boldsymbol{x}_{0:i-1}; \boldsymbol{\theta})$. Therefore, assuming that $\boldsymbol{x}$ is a large input corpus of tokens, the auto-regressive language modeling task dictates that we can pre-train $f_{\boldsymbol{\theta}}$ with maximum likelihood estimation where the loss $\mathcal{L}_{ARLM}$ is given by:

$$\mathcal{L}_{ARLM}(\boldsymbol{x}, \boldsymbol{\theta}) = -\sum_{i=1}^{|\boldsymbol{x}|} \log\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i-k:i})\right) \tag{4}$$

In practice, the transformer-based language model $f_{\boldsymbol{\theta}}$ will have an input size limit (which is usually between 512 and 2048 tokens) and pre-training is done using a context window of size k such that $f_{\boldsymbol{\theta}}(\boldsymbol{x}_{k:|\boldsymbol{x}|}) = P(x_i|\boldsymbol{x}_{k:i-1}; \boldsymbol{\theta})$. This pre-training approach is particularly useful for models tasked with language generation since the models are trained to predict the most likely tokens given the previous tokens (i.e., auto-regressive language generation). The GPT-3 model (Brown, et al., 2020) used in Chapter 7 is a decoder-only transformer and is pre-trained using this auto-regressive language modeling approach.

Masked Language Modeling Pre-Training

One clear drawback of the unidirectional auto-regressive language modeling is that the neural model can only see the tokens to the left of the predicted token, therefore model's latent representation of a token can't account for the full context. To overcome this issue, Devlin et al. (2019) adapted the Cloze prediction task as a pre-training task. In the Cloze task, some of the input tokens in the sequence are masked out as in the sentence "*My little ___ likes to bark*". In this example, this blank token "___" is likely to be associated with words such as "*dog*" due to the

context word "*bark*" that appears after it. Therefore, in the masked language modeling approach, models are trained to predict every masked-out token in the input sequence.

In practice, Devlin et al. (2019) simply masked 15% of the input tokens from the corpus at random. Out of these masked tokens, they replaced the blanks with either placeholder token "`[MASK]`" (80% of the time), a random token (10% of the time) or the true unchanged token (10% of the time). In that formulation, given a large corpus with token $\boldsymbol{x}$, the set of masked tokens $m(\boldsymbol{x})$, the modified corpus with the masked-out tokens $\boldsymbol{x}_{/m(\boldsymbol{x})}$ and a model $f_{\boldsymbol{\theta}}$ that outputs probability distributions for the masked tokens $x' \in m(\boldsymbol{x})$. Then the masked language modeling loss $\mathcal{L}_{MLM}$ is given by:

$$\mathcal{L}_{MLM}(\boldsymbol{x}, \boldsymbol{\theta}) = -\sum_{x' \in m(\boldsymbol{x})} \log\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{/m(\boldsymbol{x})}, x'\right)\right) \tag{5}$$

For encoder-only transformer models such as BERT (Devlin, Chang, Lee, & Toutanova, 2019), the final hidden vectors corresponding to the mask tokens are fed into an output $softmax$ classifier that predicts the probability distribution over the vocabulary. Alternatively, encoder-decoder models such as T5 (Raffel, et al., 2020) can feed the sentences with masked tokens to the encoder module, while the decoder is expected to output the original tokens from $m(\boldsymbol{x})$ in an auto-regressive fashion. For example, one of the language modeling tasks used to train T5 consisted of taking sentences such as "*Thank you for inviting me to your party last week*", then masking some randomly chosen tokens to create "*Thank* `<x1>` *inviting me to your party* `<x2>` *week*" (consecutive token spans are replaced with sentinel tokens "`<x1>`" and "`<x2>`") and expecting the model to output the tokens dropped-out spans as in "`<x1>` *you for* `<x2>` *last* `<x3>`" where "`<x3>`" is used as a delimiter. Many other language modeling objectives have been used including

next sentence prediction and de-shuffling the words in a sentence, each with their advantages and drawbacks.

Fine-Tuning and Few-Shot Prompting

A pre-trained language model $f_{\boldsymbol{\theta}}$ that has hopefully learned general language representations during pre-training, can be fine-tuned (or adapted) to perform a new specific task. The way that the model is fine-tuned may depend on the model's architecture and on the specific task at hand, but it will always involve performing gradient updates on $f_{\boldsymbol{\theta}}$.

In the case of encoder-only transformer models such as BERT, each input token x_i will be associated with a final hidden state $H_i \in \mathbb{R}^{d_{model}}$ containing a distributed representation of that token. Given a new task, one can add a classification head $W \in \mathbb{R}^{d_{model} \times d_{out}}$ on top of these hidden states to make predictions about the tokens themselves, where d_{out} is the vector size to be used for the output prediction. BERT also defined the first token x_0 as the classification token (which the authors call the "`[CLS]`") that should contain a latent representation for the whole sentence. In the case of encoder-decoder models such as T5, the model is fine-tuned using the maximum likelihood estimation as in Equation (4), however, the task-specific data $\boldsymbol{x}'$ is used instead of the pre-training corpus $\boldsymbol{x}$.

When the language models are scaled up to significantly larger parameter sizes (usually when $|\boldsymbol{\theta}| > 10^8$) and are pre-trained on massive amounts of data (e.g., corpus of books and web crawled data), they improve task-agnostic performance and are able to solve various NLP tasks just from a few input examples, without any specific fine-tuning. This learning approach called *few-shot prompting* (sometimes called few-shot learning) was first introduced by Brown et al. (2020).

In the case of large language models such as GPT-3, one uses few-shot prompting and includes a few question-answers pairs as examples in the input context while expecting the desired output to be generated by the model. For instance, if using a 2-shot prompting the input to the model would follow the template: "<example-question-1>; <example-answer-1>; <example-question-2>; <example-answer-2>; <prompt-question>", and the expected output should be the answer to "<prompt-question>". In the case of machine translation from English to Portuguese, then the example question could be "*translate to Portuguese: I have a dog;*" and the example answer could be "*Eu tenho um cachorro;*". The few-shot prompting approach has the clear advantage that it does not require any labeled data or costly fine-tuning of these large models, however the number of examples that can be fed to the model is still limited by the model's maximum number of input tokens.

3.2.4 Dense Retrieval

One useful application of pre-trained language models is performing textual retrieval (Zhao, Liu, Ren, & Wen, 2022). When compared to classical retrieval methods such the *bag-of-words* or TF-IDF, the language models have the advantage of computing representations that consider how tokens are semantically related to their context (Karpukhin, et al., 2020). A language model $f_{\boldsymbol{\theta}}$ can be fine-tuned to construct a dense vector representation of the input text. These representations are often used to quickly compute the similarity between two pieces of text $sim(a, b)$ by computing their vector dot product such that $sim(a, b) = f_{\boldsymbol{\theta}}(a)^T f_{\boldsymbol{\theta}}(b)$ or cosine similarity $sim(a, b) = f_{\boldsymbol{\theta}}(a) \cdot f_{\boldsymbol{\theta}}(b) / \|f_{\boldsymbol{\theta}}(a)\|_2 \|f_{\boldsymbol{\theta}}(b)\|_2$.

The dense encoder models can be trained from general-purpose language models using *contrastive learning*. Given pairs (a, b^+) of semantically similar texts and pairs (a, b^-) of semantically dissimilar texts, then the contrastive learning loss $\mathcal{L}_{CL}(\boldsymbol{x}, \boldsymbol{\theta})$ is given by:

$$\mathcal{L}_{CL}(\boldsymbol{x}, \boldsymbol{\theta}) = \mathbb{E}_{a,b^+,b^-}\left[-\log\left(\frac{\exp(sim(a, b^+))}{\exp\big(sim(a, b^+)\big) + \exp\big(sim(a, b^-)\big)}\right)\right] \quad (6)$$

Usually, the set of semantically similar pairs (a, b^+) are part of a dataset (e.g., answers marked as duplicate in internet forums) while the set of semantically dissimilar pairs (a, b^-) are automatically created by randomly sampling b^- from the set of available sentences.

4 Predicting State Changes in Procedural Text

The world around us is constantly changing due to the effects of processes. These processes are often communicated in natural language through various genres of text including recipes, scientific articles, and manuals. These genres are referred to as *procedural texts*. AI systems need to understand how the world's state evolves over time in order to perform complex reasoning over natural language.

One important task in understanding processes from procedural texts is to predict how the state of entities changes over time. In Ribeiro, et al. (2019), we present a novel approach to reading comprehension of procedural texts (e.g., answering questions about a paragraph describing the stages of photosynthesis). We use analogical learning capabilities of the Companion Cognitive Architecture to predict discrete state changes testing our method on the ProPara (Dalvi, Huang, Tandon, Yih, & Clark, 2018) dataset. The ProPara dataset contains 488 crowdsourced paragraphs describing processes and how the state of the world evolves after each step. One example paragraph and its accompanying question is shown in Table 1.

"Chloroplasts in the leaf of the plant traps light from the sun. The roots absorb water and minerals from the soil. This combination of water and minerals flows from the stem into the leaf. Carbon dioxide enters the **leaf**. Light, water and minerals, and carbon dioxide all mix together. This mixture forms **sugar** (glucose) which is what the plant eats. Oxygen goes out of the leaf through the stomata."
Question: Where is sugar produced? **Answer:** In the leaf.

Table 1: Example of question and context paragraph from the ProPara dataset describing how the world state changes over time during the process of photosynthesis.

4.1 Modeling Discrete Actions and Processes with Step Semantics

Even though processes often involve changes in continuous values such as space and time, they are colloquially described as discrete events such as "*moving a ball*" or "*baking bread*". Here we introduce *Step Semantics* (Forbus, et al., 2019), a framework we designed to describe such discrete actions and events as found in procedural texts. Step Semantics provides a set of representation conventions for processes and discrete state changes described in natural language. Initially designed to model processes from ProPara questions, we believe Step Semantics could also be applied in other similar tasks such as modeling recipes (Kiddon, Ponnuraj, Zettlemoyer, & Choi, 2015). For instance, consider the event of cooking a meal, which includes many continuous processes such as mixing, heating, chopping, etc. Ultimately, the cooking action can be viewed as a discrete event with changes of states and well-defined inputs and outputs. In our framework, continuous changes can be represented as multi-step changes (Rickel & Porter, 1994) of states, using a coarser granularity of descriptions.

4.1.1 Step Semantics Ontology

The goal is to build a formal representation of the states, steps and entities involved in a natural language description of a process. The states can be represented by a set of propositional statements that describe entities of interest. The steps represent a set of events that describe the transition between their initial and final states. The temporal ordering of states is represented using before/after relations. Therefore, the ordering of events can be represented by a graph that may contain cycles (e.g., oscillating, or recurring events).

We assume that the textual description of a process is a list of sentences which reference the entities of interest. The mapping between sentences and steps is assumed to be many-to-many,

meaning that a single step can be described by multiple sentences while a single sentence can describe multiple steps. Taking inspiration from previous work (McFate & Forbus, 2016), we use both FrameNet and Cyc to bridge the gap between natural language and a formal description. We first examine how verbs in a sentence can be linked to simple steps and summarize the representations as follows:

- Creation Steps: are represented by the FrameNet frame `FN_Creating` and the respective Cyc event type `CreationEvent`. The lexemes associated with this frame include "*create*", "*form*", "*make*", "*produce*", and others.
- Destruction Steps: are represented by the FrameNet frame `FN_Destoying` and the respective Cyc event type `DestructionEvent`. The lexemes associated with this frame include "*destroy*", "*demolish*", "*terminate*", and others. Note that certain verbs such as "transform" are often represented by both a Destruction Step (of inputs) and a Creation Step (of outputs).
- Property Change Steps: Many frames and lexemes can represent such steps. For instance, `FN_Change_of_phase_scenario` covers phase changes such as "*freezing*", "*boiling*", and others.
- Quantity Change Steps: These include frames that represent quantitative change of a property. For instance, `FN_Change_of_temperature` covers verbs such as "*heat*", "*warm*", "*cold*", and others.
- Subprocess / Event Steps: Examples include `FN_Motion` and `FN_Fluidic_motion`, and the respective Cyc event `MovementEvent`. They represent phrases often describing the source and destination of a certain entity.

Note that the natural language description of processes can be ambiguous. For the most part, we assume that the existence and properties of objects don't change over time (inertia) unless directly stated. However, this might not be true depending on the context. For instance, a textual description of a block of dry ice might not reference the sublimation process, and the reader has to understand that the solid becomes gas over time by using common sense.

4.1.2 Recipe Example

The following procedural text contains a recipe for cooking "French Toast"[2]. We use this recipe to illustrate how Step Semantics can be used to represent the states, steps and entities involved in this cooking process:

1. Whisk milk, eggs, vanilla, cinnamon, and salt together in a shallow bowl.
2. Melt butter on skillet over medium to high heat.
3. Dunk bread in the egg mixture, soaking both sides. Transfer to the hot skillet and cook until golden, 3 to 4 minutes per side.

LEXEME	FRAMENET FRAMES	ENTITIES
WHISK	`FN_Self_motion`, `FN_Amalgamation`, `FN_Creation`	milk, eggs, vanilla, cinnamon, salt, shallow bowl, egg mixture
MELT	`FN_Change_of_temperature`, `FN_Change_of_phase_scenario`	butter, skillet
DUNK & SOAK	`FN_Soaking`, `FN_Fluidic_motion`, `FN_Dunking`, `FN_Cause_to_be_wet`	bread, egg mixture
TRANSFER	`FN_Motion`	bread, egg mixture, skillet
COOK	`FN_Apply_Heat`, `FN_Change_of_temperature`, `FN_Amalgamation`	bread, egg mixture, skillet, French toast

Table 2: Lexemes, frames, and entities for French toast recipe.

[2] https://www.allrecipes.com/recipe/7016/french-toast-i/

The entities can be identified as follows: milk, eggs, vanilla, cinnamon, salt, shallow bowl, butter, bread, egg mixture, skillet, French toast. The sequence of discrete steps from the recipe are found in Table 2. Note that some of the entities are implicitly referenced by the text containing the lexeme, but they are still represented in the semantics of the step. We use Step Semantic representations and formalisms to predict state changes in procedural text from ProPara, as described below.

4.2 ProPara Problem Definition

PROCESS PARAGRAPH (simplified)	STATE CHANGES				
	Light	Water	CO2	Mixture	Sugar
leaf traps light from the sun.	**sun**	**soil**	?	-	-
The roots absorb water the soil.	MOVE **leaf**	**soil**	?	-	-
water and minerals flows from the stem into the leaf.	**leaf**	MOVE **root**	?	-	-
Carbon dioxide enters the leaf.	**leaf**	MOVE **leaf**	?	-	-
Light, water, and the carbon dioxide all mix together.	**leaf**	**leaf**	MOVE **leaf**	-	-
This mixture forms sugar.	DESTROY -	DESTROY -	DESTROY -	CREATE **leaf**	-
Oxygen goes out of the leaf through the stomata.	-	-	-	DESTROY -	CREATE **leaf**
	-	-	-	-	**leaf**

Figure 5: State change annotations for the entities of interest (participants) in the paragraph.

More formally, the ProPara paragraphs consists of a list of sentences $S = s_1, \ldots, s_N$ (e.g., "Chloroplasts in the leaf of the plant traps light from the sun") containing a number of participants $P = p_1, \ldots, p_M$ (e.g., "leaf" or "plant") that are mentioned in such sentences. The dataset provides annotations containing the state of each participant before and after each sentence. The participant's information tracked includes both *location* (e.g., in the leaf) and *existence* (e.g., if it was created or destroyed). The evolving state of the world can be described by a matrix S with

$N + 1$ rows and M columns. Each matrix entry describes the state of the M participants through the $N + 1$ steps. The data assumes three possible states for participants: does not exist (labeled as "—"), location is unknown (labeled "?"), or location is known (label is a string expressing a location). An example of annotation for a single paragraph is shown in Figure 5.

We use Step Semantics to represent the possible changes in the state (namely: `Creation`, `Destruction`, and `Movement`), identifying the participants, locations, and relations associated with the event. Step Semantics is used to map from textual lexemes to FrameNet descriptions. Such descriptions are in turn mapped to the OpenCyc ontology. The state change descriptions are referred to as *target logical forms* (or Target LFs) and are shown in Table 3.

Creation	**Destruction**	**Movement**
(isa participant1 Participant) (isa event1CreationEvent) (isa tolocation1 Location) (outputsCreated event1 participant1) (outputsCreatedLocation event1 tolocation1)	(isa participant2 Participant) (isa fromlocation2 Location) (isa event2 DestructionEvent) (inputsDestroyed event2 participant2) (inputsDestroyedLocation event2 fromlocation2)	(isa participant3 Participant) (isa event3 MovementEvent) (isa fromlocation3 Location) (isa tolocation3 Location) (objectMoving event3 participant3) (fromLocation event3 fromlocation3) (toLocation event3 tolocation3)

Table 3: Target logical forms (or Target LF). Each token in the logical forms contains relations and collections from OpenCyc, or open variables (ending in numbers).

4.3 Approach

Taking inspiration from Analogical QA training (Crouse, McFate, & Forbus, Learning from unannotated qa pairs to analogically disambiguate and answer questions, 2018), the logical forms from the semantic representation of the text (produced by CNLU) are matched to one of the Target LFs to generate query cases. An overview of the system is shown in Figure 6. During the prediction

phase these query cases are analogically retrieved and used to predict the state changes given in a sentence (sentence level prediction) from an input paragraph. A dynamic programming algorithm is applied to enforce some commonsense constraints, generating a more coherent set of state changes for the whole paragraph (global level prediction). The training (query case construction), inference (state change prediction), and commonsense constraints are described below.

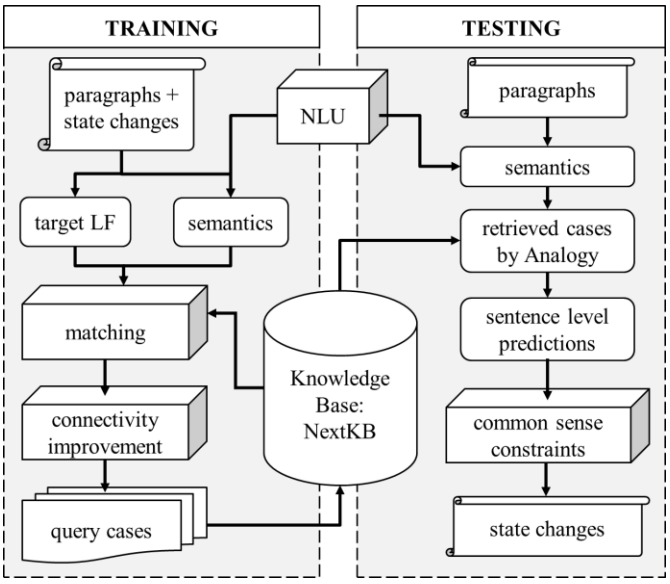

Figure 6: Overview and sub-components of the state change prediction system.

4.3.1 Query Case Construction

During the training phrase, the system pairs the most relevant input logical forms (from the semantic choices of the parsed input sentences $s_i \in S$) to their respective Target LFs, while preventing any possible choice conflicts. For example, consider the case when the input sentence is "*The roots absorb water from the soil.*". First, the system identifies the tokens of all the participants p_i (i.e., "*water*") and their annotated state changes (i.e., movement from "*soil*" to the "*roots*"). Afterwards it adds logical forms to the semantic interpretation such as `(isa`

`water4336 Participant)` and `(isa soil4486 Location)` to signal that these discourse variables should be considered when matching the semantic to a Target LF.

The objective of the graph matching phase is to select a subset of logical forms from the parsed semantics that justify the annotated state change. This is done by executing a bipartite matching algorithm that finds the best one-to-one assignment of logical forms with respect to three signals: *ontological similarity*, *structural overlap*, and *conflict avoidance*. While computing the matching, the system verifies that no two chosen semantic forms are conflicting. Here, conflicts are represented by the *semantic choices* and correspond to distinct word senses or syntax parse trees associated with the input sentence.

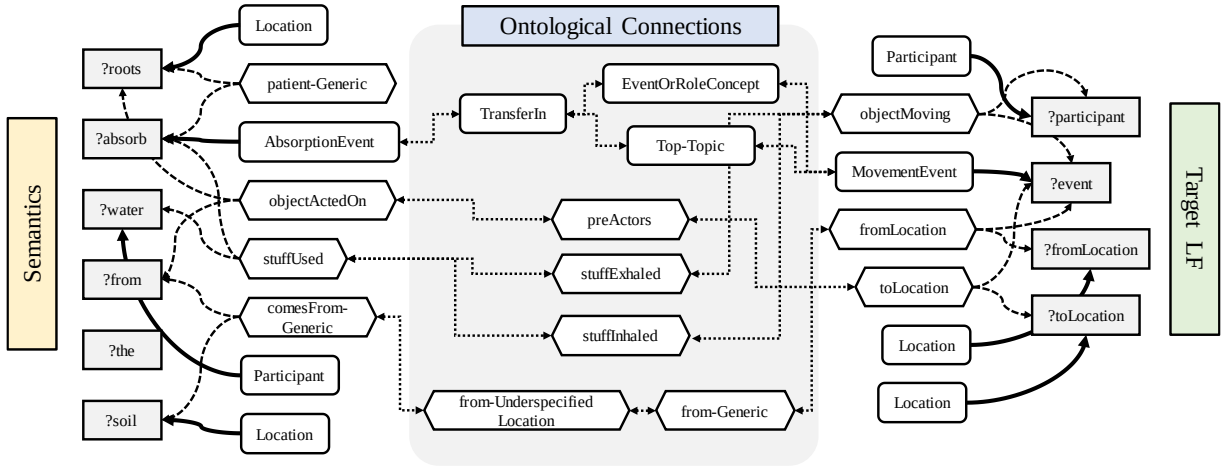

Figure 7: Example of ontological connection between semantics and target logical forms for the sentence "Roots absorb water from the soil". Gray squares contain logical variables. The full, dashed, and dotted arrows represent isa statements, role relations, and ontological structural relations, respectively.

The ontological similarity between two expressions represents how well connected they are in the knowledge base, as depicted in Figure 7. The score is computed by extracting concepts (e.g., predicates, role relations, and entities) and finding connection subgraphs (Faloutsos, McCurley, & Tomkins, 2004) defined by structural relations in NextKB. In OpenCyc ontology,

structural relations include `argIsa` argument type constraints, `isa` instance relations, and `genls` or `specs` type hierarchies. More concretely, one possible path between "*absorption*" and "*movement*" concepts from both the Semantics and Target LFs could consist of the following nodes: `AbsorptionEvent` ↔ `TransferIn` ↔ `EventOrRoleConcept` ↔ `MovementEvent`.

In general, given two a in the knowledge base, the set of paths *P* that connect such concepts through the connection subgraph formed by structural relations, and $v_{u,w}$ the vertices in these paths. Let $deg(v_{u,w})$ be the number of incoming edges to the vertex $v_{u,w}$. The ontological similarity $OntCon$ is defined as:

$$OntCon(\phi_1, \phi_2) = \sum_{p_w \in P} \frac{\prod_{v_{u,w} \in p_w} deg(v_{u,w})^{-1}}{|P|} \quad (7)$$

Note that this $OntCon$ similarity gives higher weights to paths that go through more sparse regions of the knowledge graph, penalizing nodes that are more generic (i.e., high vertex degree). The similarity is computed for all the possible connection subgraphs between the entities ϕ_1 in the semantic logical forms and ϕ_2 in the target logical forms. When ϕ_1 and ϕ_2 have no paths between them, their ontological similarity score is set to zero.

Considering the semantics and target logical forms as graphs, their structural overlap is the number of times two expressions in a matching are neighbors in their respective graphs. For instance, mapping `MovementEvent` to `AbsorptionEvent` is desirable over other mappings since their discourse variables are both connected to the `Participant` concepts in their respective logical forms. Last but not least, the conflict avoidance signal disregards matchings with a higher number of expressions in the semantic parse that have conflicting meanings (e.g., due to alternative word senses or parse choices).

These three signals are then combined using a weighted sum of their values (weights are hyperparameters). Finally, the matching procedure executes a hill-climbing algorithm that searches through the space of possible matchings, selecting the one with the best combination of signals and stopping when no better matching can be found. The query case construction algorithm is explained in more details in Appendix A.1. The optimized matching is used to create *query cases*, which are then stored in the KB for future inference. A query case example is shown in Figure 8.

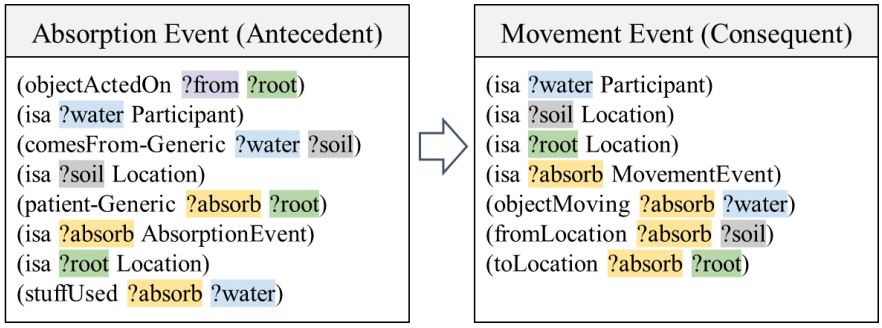

Figure 8: Example of constructed query case for the sentence "Roots absorb water from the soil". The corresponding discourse variables are highlighted with the same colors.

4.3.2 State Change Prediction

Given an input paragraph, the participants (e.g., "plant" or "leaf", which are *a priori* known) are identified by finding their tokens in the input sentences. Similar to the training phase, the semantic representation of the input sentences is obtained from the CNLU and augmented with the logical forms of the participants that are found in the sentence. For instance, if the participant "water" is found in the input sentence with associated discourse variable `water4336`, then `(isa water4336 Participant)` is added to the semantic representation. Afterwards, the MAC/FAC module retrieves a set of potential query cases in which antecedents are analogically

similar to the semantic logical forms. More concretely, considering that the query case example from Figure 8 was retrieved for the input sentence "*Neurons absorb nutrients from the blood*". Given that discourse variables from an input sentence will be analogically bound to elements in the antecedents and consequent of the query case (e.g., the discourse variables `nutrient4453`, `blood4485`, and `neuron4440` from the input sentence will bind to `?water`, `?soil`, and `?root` from the query case, respectively), then the system predicts the movement of `nutrient4453` from `blood4485` to `neuron4440`. This query case instantiation procedure is applied to infer the state changes of each sentence $s_i \in S$ in the input paragraph.

One obvious problem is that multiple query cases can be retrieved by the MAC/FAC module. The question then becomes: which query case would best predict the state change described in the input sentence? To this end, the system ranks each retrieved query case by computing some statistics over all stored query cases created during training. Let E_c and E_a be the sets of collections (e.g., "`AbsorptionEvent`" or "`Location`") from both consequent and antecedents in a stored query case, respectively. Let $\Psi = \{R_1, \ldots, R_{|\Psi|}\}$ be the relations in the antecedents, and λ be a hyper-parameter. The ranking score RS is for a given sentence-level prediction is:

$$RS(\Psi, E_c, E_v) = |\Psi| + \lambda \left(\sum_{R_i \in \Psi} P(E_c \mid E_a, R_i) \right) \quad (8)$$

This ranking score can be viewed as a sentence-level prediction, only considering the current and past sentences at a certain step in the input paragraph. However, it is possible that the sentence level prediction is not optimal according to a global level interpretation of the paragraph. For instance, if the scoring function predicts a participant was destroyed at a step t, but in the step $t + 1$ such participant is moving from one location to another, it is possible that the destruction

event was mis-classified. To mitigate this issue, the system stores all the top-ranked retrieved cases in a table P with N rows and M columns (where each entry $P[n][m]$ is a list of predicted state changes and their scores for sentence s_n and participant p_m) and feeds P the next prediction step as described below.

4.3.3 Common Sense Constraints

To obtain a globally consistent output, the system applies common sense constraints using dynamic programming. Given the set of retrieved query cases (from the sentence level predictions) and their respective ranking scores, the algorithm (shown in Figure 9) uses the following constraints to predict the output with optimum global prediction:

- **Inertia**: A participant will not move until some `Movement` event occurs involving that participant.
- **Collocation**: When a `Creation` and `Destruction` event occurs during the same discrete time step, then we assume the participant destroyed was transformed into the created participant. Therefore, the created participant will be assigned the same location (if known) as the destroyed participant.
- **Existence**: if a participant already exists, it cannot be created.
- **Absence**: if a participant does not exist, then it cannot be moved or destroyed.
- **Presence**: the algorithm penalizes predictions that assume a participant does not exist even if it is mentioned in a sentence. The presence penalty cost is represented by the hyper-parameter *P-penalty*.

- **Re-existence**: the algorithm penalizes predictions where participants are destroyed and created in the following sentence (to avoid spurious predictions). The re-existence penalty cost is represented by the hyper-parameter *R-penalty*.

Function: COMMONSENSE-OPTIMIZATION
Input: Table P with size (N, M). Each entry $P[n][m]$ is a list of predicted state changes for sentence s_n and participant p_m
Parameters: *P-penalty*, *R-penalty*
Output: State change grid

```
Create dynamic programming table D with size (N + 1, M, 2)
for n from 0 to N do
    for m from 0 to M − 1 do
        D[n][m][0] ← (null, null, 0)
        D[n][m][1] ← (null, null, 1)
for n from 1 to N do
    for m from 0 to M − 1 do
        let predictions ← P[n][m + 1] and let w ← 0
        for p in predictions do
            if event(p) = CreationEvent then
                w ← 1 if R-penalty criteria is met else 0
                UPDATE-TABLE(D, n, m, 0, 1, event(p), score(p) - (w * R-penalty))
            if event(p) = DestructionEvent then
                UPDATE-TABLE(D, n, m, 1, 0, event(p), score(p))
            if event(p) = MovementEvent then
                UPDATE-TABLE(D, n, m, 0, 0, event(p), score(p))
            if event(p) = ParticipantFound then
                w ← 1 if P-penalty criteria is met else 0
                UPDATE-TABLE(D, n, m, 1, 1, event(p), −1 * (w * P-penalty))
            end if
        end for
        UPDATE-TABLE(D, n, m, 0, 0, null, null) /* propagate prev. row */
        UPDATE-TABLE(D, n, m, 1, 1, null, null) /* propagate prev. row */
    end for
end for
return RECONSTRUCT-OUTPUT-GRID(D)
```

Figure 9: Global level state change prediction algorithm. Commonsense constraints are applied using a dynamic programming approach.

The input to COMMONSENSE-OPTIMIZATION is the input table P containing a list of the sentence-level state changes together with their ranking scores. The table $D[n][m][s]$ will store the dynamic programming values where the n index corresponds to the state change position, the

m index corresponds to the participant position, and the s index represents if participant exists ($s = 0$) or doesn't exist ($s = 1$). Each entry in the dynamic programming table $D[n][m][s]$ stores three values: (1) the best possible score that can be reached up to that step (2) the prediction event (e.g., `CreationEvent`) that that was chosen for that state, and (3) the previous state for that participant given the prediction event chosen. The *Existence* and *Absence* constraints are mandatory, but the *Presence* and *Re-existence* are just treated as an extra state change "cost" (i.e., *P-penalty* and *R-penalty*, respectively). The UPDATE-TABLE and RECONSTRUCT-OUTPUT-GRID sub routines are shown in Appendix A.3. In particular, the RECONSTRUCT-OUTPUT-GRID takes the final dynamic programming table and reconstructs the predicted state change grid, which is then used to answer the questions about the state changes involving the participants in the input paragraph as described in the following section.

4.4 Experiments

In this experiment, the system reads a paragraph containing a textual description of a process (e.g., photosynthesis) and answers templated questions regarding the state changes involving a participant p_i mentioned in the paragraph. The first set of metrics, called *Sentence-Level* metrics, which were proposed by Dalvi et al. (2018). This first set of metrics answers the following three categories of questions:

- **Cat-1**: Is p_i created (destroyed, moved) in the process?
- **Cat-2**: When (step #) is p_i created, destroyed, or moved?
- **Cat-3**: Where is p_i created, destroyed, moved?

The accuracy for these questions is derived simply from the predicted state change grid which is the output of RECONSTRUCT-OUTPUT-GRID (prediction for each sentence in the paragraph). Cat-

1 questions are asked for every participant. On the other hand, Cat-2 and Cat-3 questions are only asked for participants that have been moved, been destroyed or been created. For Cat-3, the predicted location is regarded as correct if it is either identical or a substring of the golden annotated location. The second set of metrics are called *Document-Level* and were proposed by Tandon et al. (2018). The task is to answer the following four templated questions:

- **Q1**: What are the inputs to the process?
- **Q2**: What are the outputs of the process?
- **Q3**: What conversions occur, when and where?
- **Q4**: What movements occur, when and where?

In these questions the inputs correspond to the participants that existed during the start of the process, but not at the end, while the outputs did not exist during the start but did at the end. Conversions are when some participants are created while others are destroyed. Movements correspond to events when participants that had their location changed. Since each question can have multiple answers, the metrics use precision, recall and F1 score to compare the gold and predicted answers.

4.4.1 Results

In Table 4 we report our system results, which we call Analogical Procedural Text Understanding (or APTU), together with previously published results on the ProPara dataset. The majority of the baselines are based on neural networks: QRN (Seo, Min, Farhadi, & Hajishirzi, 2017), EntNet (Henaff, Weston, Szlam, Bordes, & LeCun, 2017), PROLOCAL and PROGLOBAL (Dalvi, Huang, Tandon, Yih, & Clark, 2018), PROSTRUCT (Tandon, et al., 2018), KG-MRC (Das, Munkhdalai, Yuan, Trischler, & McCallum, 2019), LACE (Du, et al., 2019), and NCET (Gupta &

Durrett, 2019). ProComp (Clark, Dalvi, & Tandon, 2018) used a hybrid approach relying on separate sources of knowledge.

Technique	Model	Sentence-Level					Document-Level		
		Cat-1	Cat-2	Cat-3	Micro-avg	Macro-avg	Prec.	Recall	F1
Hybrid	PROCOMP	57.14	20.33	2.40	26.24	26.62	-	-	-
Artificial NN	QRN	52.37	15.51	10.92	26.49	26.26	55.5	31.3	40.0
	EntNet	51.62	18.83	7.77	25.96	26.07	50.2	33.5	40.2
	PROLOCAL	62.65	30.50	10.35	33.96	34.50	**77.4**	22.9	35.3
	PROGLOBAL	62.95	36.39	35.90	45.37	45.08	46.7	52.4	49.4
	PROSTRUCT	-	-	-	-	-	74.2	42.1	53.7
	LACE	-	-	-	-	-	75.3	45.4	56.6
	KG-MRC	62.86	40.00	38.23	46.62	47.03	64.5	50.7	56.8
	NCET	**73.68**	**47.09**	**41.03**	**53.93**	**53.97**	67.1	**58.5**	**62.5**
Analogy	APTU	61.58	40.14	18.59	39.38	40.10	62.0	45.1	52.3

Table 4: Results for ProPara dataset on both sentence-level and paragraph level evaluations.

The results show strong performance for APTU against many of the baselines. Notably, for Cat-3 (predicting when the state changes occur) we outperform all the previously reported results except for the NCET model. On the other hand, our Cat-3 results (predicting the location changes of participants) are not as strong as other baselines. After manually inspecting the state change predictions, we notice that many errors are due to incomplete semantic parses or due to the fact that we make a simplifying assumption that the location should be contained within the tokens of the sentence describing the state changes, which is not always the case. The results in the document-level evaluation seem to be suffering from a similar problem: our approach cannot always find the location of the participants in the paragraph. Nonetheless, we believe that these are strong results considering that analogical learning has other desirable properties including data

efficiency and explainability (Crouse, McFate, & Forbus, Learning from unannotated qa pairs to analogically disambiguate and answer questions, 2018; Chen & Forbus, 2021).

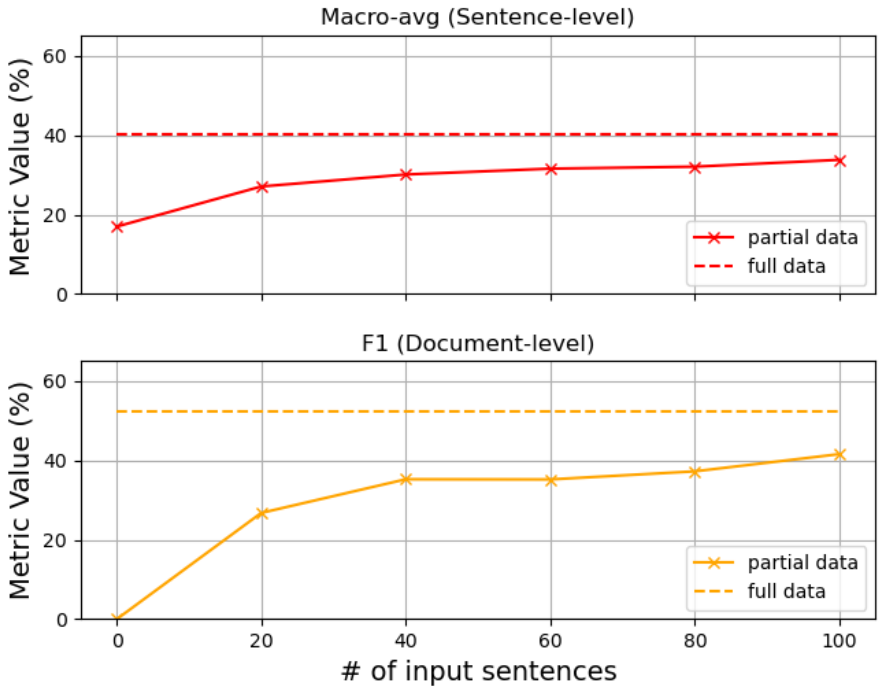

Figure 10: Graphs showing the learning curve of the APTU system for different number of training examples and evaluation metrics.

To illustrate the data-efficiency properties of analogical learning, we evaluate the system on a subset of the ProPara training data. We plot the sentence-level macro-average of Cat-1, Cat-2 and Cat-3 as well as the document-level F1 scores when the system is trained on a subset from 0 to 100 training sentences, as shown in Figure 10. The subset of training input sentences $s_i \in S$ are taken from the pool of 2,639 total training sentences. We automatically select the sentences that are likely to be relevant to a large number of test examples. This is done by looking at the generated query cases from the full training data and counting occurrences of concept and role-relations in the antecedents which appear most often, then selecting the sentences associated with such query

cases. The results show that with only 100 training examples, the system already performs close to the results with the full training data (84.3% for macro-avg and 79.5% of the F1 score), even outperforming other baseline such as QRN, EntNet, and PROLOCAL that were trained on the full data.

4.4.2 Error Analysis

To further understand the strengths and weaknesses of our approach, we randomly selected five paragraphs from the development set in ProPara, categorizing common errors made by the system. We found that the most common errors (33%) were due to implicit state changes. For instance, if a paragraph is describing some garbage collection process where a bag is added to a trash truck, then a sentence "*The trash truck travels to the landfill*" will imply that the trash bag inside the truck also moved to the landfill, even if that is not explicitly stated in the sentence. Such cases are very challenging to predict and would require embedding more commonsense reasoning and rules into the system. The second most common type of errors (28%) are due to CNLU incomplete or inaccurate parses. These errors can stem from missing syntactic and grammatical rules, incorrect coreference-resolution involving nouns, or finding the relationship between participants and their locations when they are far apart in the paragraph. The third most common error (21%) was due to incorrectly retrieved cases. This happens either when the training data does not contain certain events (e.g., *bag up* as in the sentence "*trash is bagged up*", which implies that the trash moved to the bag) or when a sentence implies a different state change due to its context. The remaining errors were due to incorrect ranking of scores (14%) and noisy data (4%), which we believe were mislabeled by the human annotators.

4.5 Conclusion

Understanding how the world evolves over time is a key aspect of human-level reasoning. In this chapter we discuss how analogical learning can be applied to answer questions about procedural texts describing various processes (e.g., photosynthesis). We build upon the Step Semantic formalisms and develop a system that can extract patterns from the training data which are then analogically retrieved to answer questions. Predictions that only consider a subset of the input sentences in the paragraph may be globally inconsistent. To mitigate this issue, we apply a set of commonsense constraints using dynamic programming to arrive at a more plausible sequence of state changes. The resulting system is shown to have strong results against other baselines on the ProPara dataset, while having other desirable properties such as data-efficiency (achieving around 80% performance from only 100 examples) and producing inspectable outputs.

5 Analogical Knowledge Extraction

To perform complex reasoning and draw new conclusions, intelligent agents need to use and manipulate knowledge. Within the context of natural language processing, structured knowledge has also proven to be useful in many downstream tasks including dialogue systems, question-answering and language generation (Lukovnikov, Fischer, Lehmann, & Auer, 2017; Wilson, et al., 2019; Lin, Tseng, & Byrne, 2021; Forbus, et al., A Prototype System that Learns by Reading Simplified Texts., 2007). With that in mind, in Ribeiro & Forbus (2021) we tackle the long-standing problem of automatically extracting structured knowledge from natural language text (Lao, Subramanya, Pereira, & Cohen, 2012; Distiawan, Weikum, Qi, & Zhang, 2019). The goal is to not only extract information about named entities (e.g., countries or companies), but also general world knowledge (e.g., everyday objects or elementary science facts) by leveraging the knowledge already contained in the knowledge base (KB).

In our approach, which we call *Analogical Knowledge Extraction* (or AKE for short), we use the relations and concepts defined in Companion's NextKB as the starting point, expanding the KB by extracting information from Simple English Wikipedia[3] articles. One important aspect of NextKB is that it uses OpenCyc as its main source of knowledge, which has three desirable properties:

1. It contains general world knowledge covering various topics including elementary sciences, human activities, and everyday objects, contrasting with other sources such as NELL (Carlson, et al., 2010) and FreeBase (Bollacker, Evans, Paritosh, Sturge, & Taylor, 2008) that mostly focus on named entities.

[3] https://simple.wikipedia.org/

2. It contains a large set of semantic relations, as opposed to other sources such as WordNet (Miller, 1998) which has less than a dozen relations.
3. The concepts and relations are represented by unique symbols, as opposed to plain textual representations. For instance, the term "*mouse*" in OpenCyc could be represented by either `Mouse-Rodent` or `ComputerMouse`, which have distinct meanings. Knowledge resources that represent their concepts or relations with plain text include ConceptNet (Speer, Chin, & Havasi, 2017), Aristo Tuple KB (Dalvi Mishra, Tandon, & Clark, 2017), ATOMIC (Sap, et al., 2019), and Ascent (Nguyen, Razniewski, & Weikum, 2021).

In order to train our system, we apply *distant supervision* (Mintz, Bills, Snow, & Jurafsky, 2009) to automatically create training examples for analogical training and extract knowledge from a text corpus. This learning paradigm is non-supervised, meaning that we do not use any human-annotated data directly labeling text to their structured representation. To the best of our knowledge, AKE is the first to combine Companion's analogical learning capabilities with Transformer neural models. The system uses the *Bidirectional Encoder Representations from Transformer* (or BERT) (Devlin, Chang, Lee, & Toutanova, 2019) to help disambiguate between distinct word sentences (e.g., differentiate between `Mouse-Rodent` or `ComputerMouse`) from the textual semantic representations. Furthermore, BERT is used in a post-processing step to rank predicted facts according to their plausibility.

In our experiments around 94.8% of relations in the selected KB are associated with less than 100 facts. Therefore, this distant supervision will generate relatively fewer training examples per relation compared to other knowledge sources. Even in low resource settings, our results show

that AKE can extract high precision facts compared to other baselines, demonstrating the data-efficiency of analogical learning.

5.1 Problem Definition

The knowledge extraction task consists of predicting facts from natural language while following the schemas, concepts and relations defined by an existing KB. Previous works (Weston, Bordes, Yakhnenko, & Usunier, 2013; Dalvi Mishra, Tandon, & Clark, 2017) on knowledge or relations extraction often represent facts as `(h,r,t)` triples such as `(Brazil, borders, Argentina)`. On the other hand, we use the CycL ontology language (Lenat & Guha, 1989) which allows for a more expressive representation of facts.

In CycL all concepts are represented using disambiguated constants (as in `Mouse-Rodent` or `ComputerMouse`). Concepts are represented by collections, where `Mouse-Rodent` is a collection, while `mouse123` denotes an instance or member in that collection. All the relations are of arbitrary arity (not just binary), where relations such as `between` can naturally have three arguments, as in the example `(between ContinentOfEurope ContinentOfAsia UralMountains)`. Furthermore, relations can be of higher order and take collections and predicates as arguments such as `(relationAllExists eatsWillingly Omnivore Meat)`. Logical functions can produce new terms from existing ones, where `(FruitFn AppleTree)` represents the "*fruit of an apple tree*".

More formally, the knowledge extraction task can be defined as follows. Two inputs are expected, a text corpus $TC = (s_1, \dots, s_{|TC|})$ composed of sentences s_i and a knowledge base $KB = (C, R, F)$. In KB, the term C represents the set of all available concepts (in CycL, they can be either collections, entities, or logical functions), R represents the set of all relations and predicates while

F is the set of facts (which is a nested tuple containing elements from C and R). The objective is to learn new facts $f' \notin F$ which are not already present in the KB that are expressed in the sentences $s_i \in TC$.

5.2 Neural Word Sense Disambiguation

Because of homonymy and polysemy, lexemes might have multiple meanings and interpretations. Therefore, a key challenge in knowledge extraction is disambiguating the meaning from the words in the corpus. For this reason, we augment the CNLU parser by including a word sense disambiguation module that scores different parse choices according to the context. This module uses BERT trained on the FrameNet[4] data, which resembles the approach by Tan & Na (2019). FrameNet contains annotations mapping from parts of sentences to their corresponding frames. For instance, the word "*know*" may be associated with either the `[Awareness]` frame or the `[Familiarity]` frame.

Given an input sentence s with tokens $t_1, \ldots, t_{|s|}$, we want to assign a probability that a FrameNet frame FN_k is associated with a given subsequence $t_i, \ldots, t_j$ of such tokens such that $1 \leq i, j \leq |s|$. We fine-tune BERT by adding a classification head with weights $W \in \mathbb{R}^{K \times 4|H|}$. Here, the value H represents the model's output hidden state vector for each token with size $|H|$, while the value K is the number of distinct FrameNet frames. This module maps all sequences of four (i.e., $j - i = 3$) consecutive tokens to a probability distribution over the possible frames. The probability that a subsequence of tokens in s is associated with a frame F_k is represented by $Fra\_Pro(s, i, j, FN_k)$ and is given by the following formula:

[4] https://framenet.icsi.berkeley.edu/

$$Fra\_Pro(s, i, j, FN_k) = softmax\left(\left[H_i, : H_j\right]W^T\right)_k \quad (9)$$

This module is used to assign probability score for each choice from choice sets within CNLU's semantic parse. Given a logical form $\Upsilon_a \in \Upsilon$ from CNLU's semantic parse which is part of a choice with frame FN_k, we will use the notation $Fra\_Pro(\Upsilon_a)$ to represent the frame probability score given to the logical form Υ_a. All choices in a choice set which are associated with no FrameNet frame will receive probabilities from a uniform distribution. This disambiguation module is used when parsing sentences in both the training and testing phases.

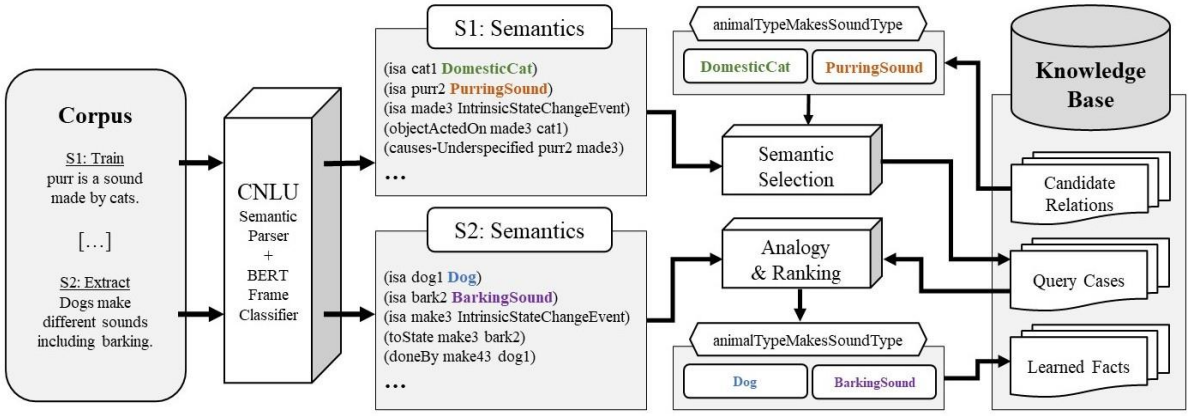

Figure 11: Overview of system components of the Analogical Knowledge Extraction (AKE) system. The input data consists of a corpus describing general knowledge (e.g., "*purr is a sound made by cats*") and a knowledge base with facts which are used to extract training examples using distant supervision.

5.3 Analogical Knowledge Extraction

The AKE pipeline consists of four main phases. First, the system parses the sentences from the text corpus TC and finds facts in the knowledge base KB in which entities are referenced by the sentences. These sentences are selected as distant supervision training examples. Second, the

system uses analogical learning to identify patterns in the training examples. These patterns are stored as *query cases* and will help extract new facts from the corpus afterwards. Third, the system performs a second pass over the text corpus TC, this time trying to learn new facts by using the stored query cases. Finally, AKE performs a post-processing step where all the learned facts are assigned a score, discarding ones which are considered less plausible. We detail each of these steps in the following sections. An overview of the system components and data flow among them is shown in Figure 11.

5.3.1 Learning with Distant Supervision

The *distant supervision* assumption is that if concepts from a fact $f \in F$ are referenced by a sentence s_i, then that sentence might express that fact. We use that assumption to generate training examples without any explicitly labeled data. First, the sentences s_i from the input corpus are parsed by CNLU to produce a predicate calculus semantic representation. Afterwards all facts containing two or more concepts in the semantic representation are retrieved from the KB. For instance, consider the sentence "*Cats use many different sounds for communication, including meowing and purring.*". The semantic representation will contain logical forms such as `(isa cat158446 Cat)` and `(relationInstanceExists waveEmitted purr158909 PurringSound)`. Since both `Cat` and `PurringSound` concepts are present in the semantic representation, then AKE will assume that this particular sentence expresses the existing KB fact `(animalTypeMakesSoundType Cat PurringSound)`.

In general, all the facts that are retrieved from the KB this way will be referred to as *candidate facts*. The collection of all candidate facts is represented by the set Ψ where $\Psi_i \in F$.

Likewise, all the concepts in the candidate facts (e.g., `Cat` or `PurringSound`) will be referred to as *anchor concepts*. Ω represents the set of all anchor concepts where $\Omega_i \in C$.

Function: SEMANTIC-SELECTION
Input: Sentence semantics Υ, retrieved candidate fact Ψ_i, anchor concepts Ω for that fact.
Parameters: linear combination weights λ_1, λ_1, λ_1, λ_1, and score threshold ε_1
Output: Subset of semantics $\Upsilon^* \subseteq \Upsilon$

Select a set of logical forms $\Phi_{a,b} \subseteq \Upsilon$ from paths connecting two anchor concepts Ω_a and Ω_b through the semantic representation graph with depth of up to 3 nodes.
foreach $\Upsilon_a \in \Upsilon$ **do**
  if $\Omega_c \in \Upsilon_a$ for some $\Omega_c \in \Omega$ **then**
    $con \leftarrow |\Phi_{a,b}|$
    $fac \leftarrow \sum_c Ont\_Con(\Omega_c, \Psi_i)$
    $gen \leftarrow Gen(\Omega_c)$
    $src[\Upsilon_a] \leftarrow \lambda_1 * con + \lambda_2 * \log_{10}(1 + fac) + \lambda_3 * \log_{10}(1 + gen)$
  else
    $\Omega' \leftarrow$ set of all collections from Υ_a
    $src[\Upsilon_a] \leftarrow \lambda_2 * \sum_c Ont\_Con(\Omega'_c, \Psi_i)$
  end if
  $src[\Upsilon_a] \leftarrow src[\Upsilon_a] + \lambda_4 * Fra\_Pro(\Upsilon_a)$
end for
$\Upsilon_a \leftarrow$ sort Υ by $src[\Upsilon_a]$
$\Upsilon^* \leftarrow \emptyset$
foreach $\Upsilon'_a \in \Upsilon'$ **do**
  if $\forall\ \Upsilon^*_c \in \Upsilon^* \neg\ Conflicts(\Upsilon'_a, \Upsilon^*_c)$ **and** $src[\Upsilon_a] > \varepsilon_1$ **then**
    $\Upsilon^* \leftarrow \Upsilon^* \cup \Upsilon'_a$
  end if
end for
return Υ^*

Figure 12: Analogical KE semantic selection algorithm.

5.3.2 Construction of Query Cases

Given the sentences in the corpus that are associated with and candidate facts, the AKE system uses a modified version of Analogical Question-Answering Training (Crouse, McFate, & Forbus, Learning from unannotated qa pairs to analogically disambiguate and answer questions, 2018) to generate *query cases*. Such query cases can be thought of as functions that can map the semantic representation (antecedents) to the given generalization of a fact in the KB (consequent). When

producing query cases, the system needs to consider how to best select a subset of the parsed semantic forms such that these can be later used to infer new facts.

The algorithm that performs such selection is shown in Figure 12. It takes as input sentence semantics Υ produced when parsing sentence s_i, a retrieved candidate fact Ψ_i and all their anchor concepts Ω. To illustrate this process, we take as example the sentence "*Bees also have a stinger at the back of the abdomen*". The semantic representation of the sentence contains the anchor concepts `Bee` and `Stinger` that relate to the candidate fact `(properPhysicalPartTypes Bee Stinger)`. First, the system selects set of logical forms $\Phi_{a,b} \subseteq \Upsilon$ connecting any two anchor concepts Ω_a and Ω_b through a path in Υ containing three or fewer nodes. In this case, the logical forms `(isa have2924551 StaticSituation)`, `(fe_possession have2924551 stinger2924603)` and `(possesses bee2924540 have2924551)` would be part of the set since it connects the discourse variables `stinger2924603` and `bee2924540`. Furthermore, the algorithm relies on three signals to rank the logical forms from Υ called *semantic connectivity*, *ontological connection* and *type generality*.

The *semantic connectivity* signal represents the logical forms that connect the anchor concepts of a fact together and is simply defined as the size of the set $\Phi_{a,b}$. The second signal, *ontological connection*, helps disambiguate the semantic choices by prioritizing concepts that are more closely related to the candidate fact. The ontological connection Ont_Con (Ribeiro, et al., 2019) defined in Equation (7) in Chapter 4 computes a correlation between two concepts, ϕ_1 and ϕ_2, by finding paths connecting them in the graph defined by the KB.

The third and last signal, *type generality*, gives lower priority to more specific concepts. Given a concept x, the score computes the number of facts F_x in the KB that contain such concept. The formula is as follows:

$$Gen(X) = |\{F_x \, | F_x \in F \wedge X \in F_x\}| \quad (10)$$

For instance, the concept `Water` will have a higher generality score than `Water-Saline` since it appears in more facts in the KB. Finally, all three signals are combined to select a subset of the parsed semantics and create a query case. A query case example is shown in Figure 13. The `isa` statements from the parsed semantics associated with the arguments of the candidate fact are used as non-abducible antecedents (necessary condition) while the remaining are set as abducible antecedents (additional evidence).

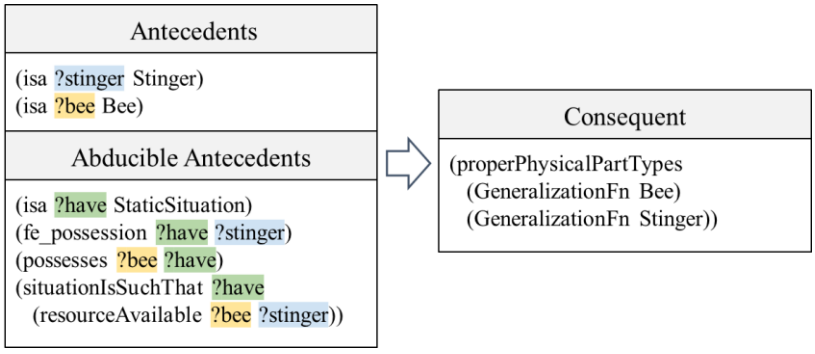

Figure 13: Query case example for the sentence "Bees also have a stinger at the back of the abdomen". The corresponding discourse variables are highlighted with the same colors.

5.3.3 Predicting New Facts

The third phase consists of going through the corpus TC and extracting structured knowledge (facts) that could be expressed using the relations and concepts already defined in the KB. The Analogical KE system parses each sentence using CNLU. Afterwards, the system uses the sentence's semantic representation as probes to analogically retrieve the stored query cases using

MAC/FAC. In our implementation, we further process the query cases retrieved by MAC/FAC and select up to 20 cases per probe[5].

The goal is to decide if the semantic representation of the input sentence contains sufficient evidence so that new facts can be predicted. To this end, the system uses a fact scoring formula to compute scores for each instantiated query case. This scoring formula is designed to ensure that (1) the predicted facts are relatively similar to the query case consequent (i.e., similar to the corresponding candidate facts from training) (2) the word senses in the semantic representation have high probability and (3) the concepts inside the predicted fact are not too generic (i.e., avoid concepts with a large number of descendants in the genls hierarchy such as `Event` or `Thing`). Suppose that the instantiated query case has antecedents A and consequent C. Given the anchor concepts Ω^c from the input sentence and the anchor concepts of the original fact Ω^o from the consequent C, then, the fact scoring formula is given by:

$$\lambda_5 * \sum_{A_u \in A} FRA\_PRO(A_u) + \lambda_6 * |A| - \lambda_7 * log_{10}\big(GEN(\Omega^c)\big) + \lambda_8 * \log_{10}\Big(\sum_{u,v} ONT\_CON(\Omega_u^c, \Omega_v^c)\Big) \tag{11}$$

In this formula λ_5 through λ_8 are system hyper-parameters. Any predicted facts that are already in the KB or have fact scores lower than a given threshold ε_2 are filtered out. Finally, the learned new facts, the provenance (sentence that entailed the new fact) and the fact scores are stored in the KB.

[5] Experiments showed that 20 cases per probe was a number that was small enough that didn't slow down the system but large enough to not compromise the prediction quality.

5.3.4 Fact Scoring

In the last phase of the pipeline all the output facts (extracted knowledge) are stored using a separate BERT model. Since BERT is pre-trained on a large text corpus, we assume that the model's weights will implicitly store some general world knowledge that will help rank the predicted facts. We take inspiration from Yao, Mao, & Luo (2019) and predict the plausibility of new facts by fine-tuning BERT on a "linearized" textual encoding of the predicted facts. For instance, the fact `(pathogenCausesConditionType Borrelia LymeDisease)` is linearized to "*[CLS] pathogen causes condition type [SEP] lyme disease [SEP] borrelia [SEP]*", where "*[CLS]*" is the special classification token and "*[SEP]*" is a separation token used to mark boundaries between sentences.

The training examples are generated from the full set of facts F in the KB. All facts $f \in F$ are used as positive examples. Negative examples are automatically generated either by shuffling the order of concepts in f (ignoring facts in which relations are part of the set of symmetrical relations $R°$ such as `bordersOn`) or by switching a concept within f with a random concept in C. Without loss of generality, considering all facts $f \in F$ are triples in the format (h, r, t) where $r \in R$ and $h, t \in C$, the set of negative training examples F^- is given by:

$$\begin{aligned} F^- = \{(t,r,h) | (h,r,t) \in F \wedge r \notin R°\} \cup \\ \{(h,r,t') | t' \in C \wedge t' \neq t \wedge (h,r,t') \notin F\} \cup \\ \{(h',r,t) | h' \in C \wedge h' \neq h \wedge (h',r,t) \notin F\} \end{aligned} \tag{12}$$

Both F and F^- are used to fine-tune BERT which is used as a binary classifier. The combined set $F \cup F^-$ has a total of 39,447 training examples. A classification head $W' \in \mathbb{R}^{2\times|H|}$ is added, where $|H|$ is the model's hidden state size. The output score $s^i \in \mathbb{R}^2$ for a fact $f_i \in F \cup F^-$ is

computed as $s^i = sigmoid(T^{CLS}W'^T)$ where $T^{CLS} \in \mathbb{R}^H$ is the hidden vector for the special classification token `[CLS]`. The training is done by optimizing the cross-entropy loss $\mathcal{L}$ as in:

$$\mathcal{L} = \sum_{f_i \in F \cup F^-} (y^i * \log(s_0^i) + (1 - y^i) * \log(s_1^i)) \quad (13)$$

Here, the value y^i represents the expected output, where $y^i = 1$ if $f_i \in F$ or $y^i = 0$ if $f_i \in F^-$. After training, the fine-tuned model achieves 89% fact classification accuracy on a held-out development set.

5.4 Experiments and Results

In this section, we describe the corpus and data used to extract facts as well as baseline models that are used for comparison. All the training details and hyper parameters used during experiments are shown in Appendix A.4.

5.4.1 Corpus and Knowledge Base Details

We select a subset of Simple English Wikipedia[6] articles such that the article titles contain tokens in a list of common nouns including animals (e.g., "*alligator*"), house objects (e.g., "*towels*"), places (e.g., "*California*"), and science related terms (e.g., "*zygote*"). Articles with titles such as "*20th Century Classical Music*" or "*Harry Potter*" are excluded. All the article names from Simple English Wikipedia used for these experiments are listed in Appendix A.4. The final corpus contains a total of 2,679 articles.

Likewise, we selected a subset of NextKB which we believe expresses general facts. This is done by selecting all facts from a list of microtheories. In Cyc, a microtheory represents a group

[6] https://simple.wikipedia.org/

of similar assertions and facts. The list includes microtheories such as `AnimalActivitiesMt`, `BiologyMt` or `HumanActivitiesMt`. All the facts containing relations such as `quotedIsa` and `givenNames` are excluded. The final set of facts is available online[7] and contains 66,649 facts covering 3,745 distinct relations and 21,462 concepts.

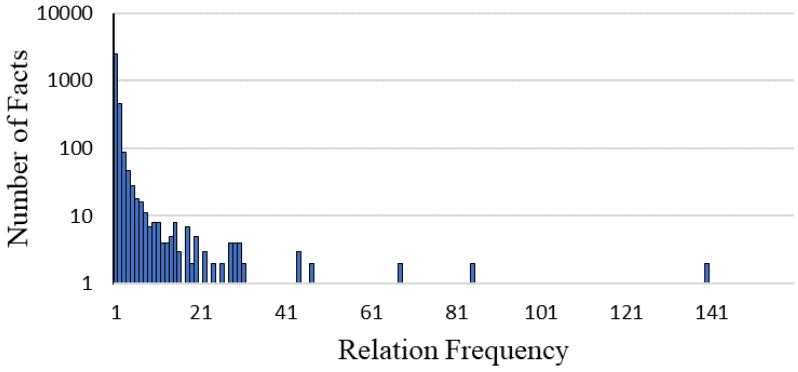

Figure 14: A histogram showing the distribution of relations according to their frequency (number of occurrences in distinct facts) in the knowledge base.

To further analyze this KB subset, we plot in Figure 14 a histogram depicting the frequency of relations. In this plot, the values in the horizontal axis refer to the number of times (frequency) a relation appears in the KB, while the values in the vertical axis refer to the total number of facts with relations in that frequency. For instance, there are 2,497 facts in the KB in which relations appear once in the KB, while only 8 facts have relations that appear 10 times. This is evidence that the selected facts are relatively sparse, and the frequency of relations follows a long-tail distribution. This fact further justifies the use of data-efficient approaches such as analogical learning to solve this task.

[7] https://github.com/dnr2/analogical-ke

5.4.2 Baselines

We compare our results against three other baselines, where the distant-supervised examples from AKE are used to create a training and development set. The first is based on the Text-to-text Transfer Transformer (or T5) by Raffel et al. (2020), where we fine-tune the model to take as input a sentence and output a linearized version of the structured knowledge. The linearized encoding is similar to AKE's Fact Scoring module, where a fact such as `(typicalMainConstituent-TypeType BookCopy Paper)` gets encoded as "*typicalMainConstituent-TypeType [S] BookCopy [S] Paper*".

Method	**Estimated Precision**
Relation Extraction * (OpenRE + CNN)	17.1%
Relation Extraction * (OpenRE + BERT)	20.8%
Text-to-text (T5 [`base`])	26.0%
Analogy (AKE)	45.7%
Analogy (AKE + BERT fact scoring)	**71.4%**

Table 5: Evaluation results showing the estimated precision of AKE and other baselines. Models with * can only produce facts with binary relations.

Two other baselines use the relation extraction model OpenNRE by Han et al. (2019). We use both BERT and CNN (Zeng, Liu, Lai, Zhou, & Zhao, 2014) as sentence encoders. Note that open implementation of OpenNRE can only predict facts with binary relations, which is limiting since the facts in NextKB contain multi-arity relations. For this reason, we exclude any facts with more than two arguments while training the OpenNRE baselines. During testing, the head and tail arguments for the relation extraction baselines are identified from the surface form of the concepts in the KB (taken from CNLU lexical mappings).

5.4.3 Results

After performing knowledge extraction on the Simple English Wikipedia corpus, we evaluate the predicted facts (excluding the ones already present in the KB). Unlike other knowledge extraction tasks (Mintz, Bills, Snow, & Jurafsky, 2009; Han, et al., 2018), we do not have a held-out test split. Instead, we randomly sample 8% of the predicted facts and manually evaluate their correctness. The estimated precision averaged across different relations are shown in Table 5.

When it comes to the number of output facts, the text-to-text (T5) model extracted 6,772 new facts, while the adapted relation extraction models (OpenRE) using the CNN and BERT encoder extracted 2,448 and 2,804 facts, respectively. Interestingly, the OpenNRE models are able to perform well on a held-out development set from the distant supervision examples (achieving over 80% F1 scores when it comes to predicting the relation given between two concepts) but their performance drops on the proposed knowledge extraction task, possibly because some facts and relations are outside the distribution of the training data.

The AKE estimated precision outperforms baselines in terms of precision, predicting a total of 4,458 facts covering 58 distinct relations. Adding the BERT fact scoring phase greatly increases the precision, but the total number of predicted facts drops to 976 items. Figure 15 shows the estimated precision for a subset of AKE predicted facts broken down by relation. After a more careful inspection of the results, we see that the system is able to perform reasonably well even when the distant supervision data is sparse. For instance, in the initial KB there are only four examples of facts in the format `(relationAllExists eatsWillingly ?x ?y)`, but a total of 156 new facts are predicted in the end.

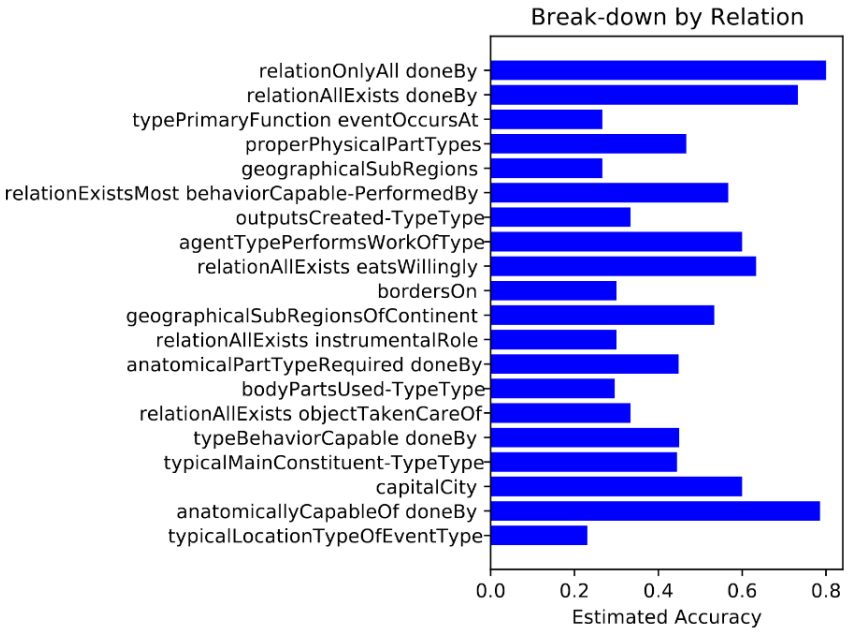

Figure 15: Estimated precision of the AKE system broken down by relation types.

Furthermore, we performed an error analysis of AKE's outputs and found that roughly 42% of the failures were due to semantic parsing errors. The majority of the remaining erroneous predictions were due to classification errors from BERT's word sense disambiguation component and incorrect semantic selection during the creation of query cases. We believe that improving the coverage of the training data from FrameNet would greatly mitigate the issues caused by the word sense disambiguation component.

5.5 Conclusion

Knowledge acquisition is an important task for any intelligent agent that needs to perform general reasoning. In this chapter we present AKE, a hybrid approach based on both analogical and neural learning that can automatically extract general world knowledge from natural language text. The AKE system is built on the Companion Cognitive Architecture, and it extracts patterns from the

semantic representation of the input text, which are later used to analogically predict new facts. Two BERT-based modules are introduced to help with both word sense disambiguation and fact scoring. Our experiments show that our system is able to outperform other baselines, producing high-precision facts from a small number of distantly supervised training examples.

6 Natural Language Explanations with Entailment Trees

With the goal of making the output of neural language models less opaque, in Ribeiro et al. (2022) we propose a question-answering system that can create natural language explanations by both (1) retrieving supporting evidence from a corpus of textual facts and (2) generating a systematic step-by-step chain of reasoning from the retrieved textual evidence. In this work, the chain of reasoning is represented using *entailment trees*, which were first introduced by Dalvi et al. (2021).

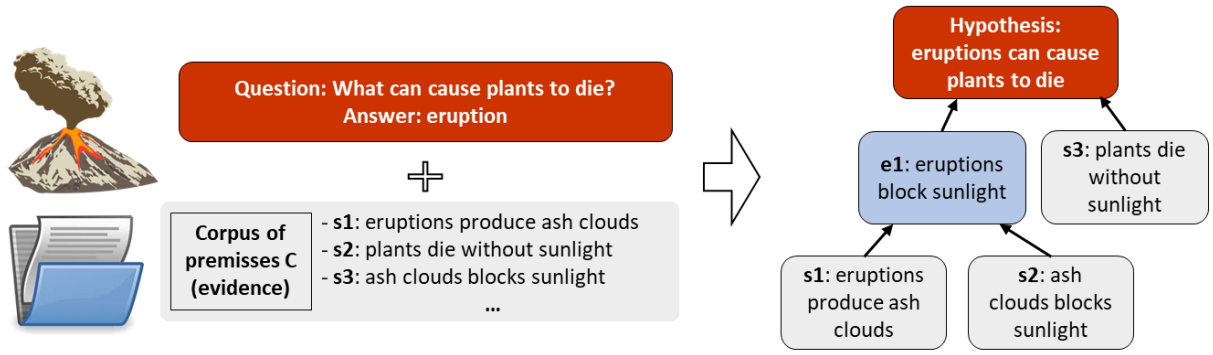

Figure 16: Example from EntailmentBank dataset. Given a hypothesis H (a combination of question and possible answer), the system retrieves a set of textual premises from a corpus C and constructs an entailment tree (on the right-hand side) that explains the hypothesis H using such premises. Gray nodes represent sentences from the corpus, while blue nodes represent generated intermediate conclusions.

Entailment trees are structures comprised of multi-premise entailment steps that show how a hypothesis (or an answer to a question) can be explained from simpler textual facts, as depicted in Figure 16. The goal is to take as input a textual hypothesis and a corpus of textual premises and output such entailment trees as a way to explain the input hypothesis. Entailment trees are more expressive than other natural language explanation methods such as retrieval of passages (DeYoung, et al., 2020) or multi-hop chains (Jhamtani & Clark, 2020) since it contains multi-premise textual entailment steps.

Previously, the generation of entailment trees was done by first retrieving a subset of the premises from the corpus and then using an encoder-decoder language model to output a textual encoding of the entailment trees contaning such premises (Dalvi, et al., 2021; Tafjord, Dalvi, & Clark, 2021; Bostrom, Zhao, Chaudhuri, & Durrett, 2021). These approaches had two drawbacks. (1) The input size limit of the language models reduces the number of possible retrieved premises from the corpus. If incorrect premises are retrieved, the output entailment tree will contain inaccurate inferences. (2) The retrieval of premises is done in a single step, and the model cannot not recover from a situation where an incorrect set of premises are retrieved.

To mitigate these issues, we propose the ***I**terative **R**etrieval and **G**eneration **R**easoner* (IRGR), a system that can iteratively retrieve premises while generating a single intermediate conclusion at a time. The system leverages intermediate conclusions to help with the retrieval of premises for the following entailment steps. We show that this approach greatly improves the retrieval of premises and consequently generates better explanations for the hypothesis. Our results show a 300% gain in the overall correctness of generated entailment trees over previous baselines.

6.1 Problem Definition

Formally, the input consists of a corpus of premises C (with simple textual facts and rules such as "*eruptions produce ash clouds*" or "*a human is a kind of animal*") and a hypothesis h. The goal is to output an entailment tree T that explains the hypothesis h using a set of retrieved premises from C as nodes. The entailment tree $T = (h, S, E, P)$ can be represented as a tree data structure. The hypothesis h is always the root node. The leaf nodes are sentences $s_i \in S$ which are retrieved from the corpus C (i.e., $S \subseteq C$). The internal tree nodes $e_i \in E$ represent intermediate conclusions and are new sentences generated by the system. Finally, $p_i \in P$ are entailment steps representing one

inference step from a conjunction of nodes to an intermediate conclusion (e.g., "s_1: eruptions produce ash clouds" and "s_2: ash clouds block sunlight" => "e_1: eruptions block sunlight).

6.2 Iterative Retrieval and Generation Reasoner

In order to generate such entailment trees, IRGR intertwines the generation of conclusions with the retrieval of premises through multiple iteration steps $t \geq 1$. The IRGR system has two main modules, namely the retrieval module or $IRGR\_retriever$ and the generation module or $IRGR\_generator$. At each iteration step, the $IRGR\_retriever$ selects a subset of premises from the corpus $S_t \subseteq C$, which are then used by the generation module ($IRGR\_generator$) to output a single entailment step p_t per iteration, which is appended to the partially generated tree in a bottom-up fashion. This process continues until the full entailment tree T is generated. The retrieval module can be formally represented as follows:

$$S_t = IRGR\_retriever(h, p_{t-1}) \tag{14}$$

Given the set of previously generated intermediate steps $P_{1:t-1} = (p_1, \dots, p_{t-1})$, the generator module can be formally described as:

$$p_t = IRGR\_generator(h, S_t, P_{1:t-1}) \tag{15}$$

The retrieval-generation process has two stop conditions: (1) when the generator produces an entailment step that contains the hypothesis h or (2) when $t \geq t_{max}$, where t_{max} is the hyper parameter limiting the maximum number of generation steps. We give a more detailed description of both modules in the following sections.

6.2.1 Dense Retrieval of Premises

The $IRGR\_retriever$ module searches over the corpus of premises C using *dense passage retrieval* (Karpukhin, et al., 2020). Differently from previous work, the $IRGR\_retriever$ fetches a different set of premises for each generation step. The $IRGR\_retriever$ module may return a variable number of premises, meaning that $|S_t|$ can change for different iteration steps t. The size $|S_t|$ ensures that the concatenation of sentences $h \oplus S_t \oplus P_{1:t-1}$ (the generator module's input) is smaller than the language model input token size limit. In our experiments $|S_t| \ll |C|$ and $|S_1| = 25$. To perform dense passage retrieval, the module uses a sentence encoder function ϕ that maps sentences to a M-dimensional vector representation in $\mathbb{R}^M$. The sentence encoder tries to map semantically similar sentences to vectors that are close together in this M-dimensional embedding space according to some similarity function. Therefore, the retrieval probability for a given premise $c \ \ in \ C$ given the hypothesis h and the previous entailment step p_{t-1} is defined as:

$$P(c \mid h, p_{t-1}) = \frac{\exp(\langle \phi(c),\ \phi(h \oplus p_{t-1}) \rangle)}{\sum_{c' \in C} \exp(\langle \phi(c'),\ \phi(h \oplus p_{t-1}) \rangle)} \tag{16}$$

In principle, the sentence encoder ϕ can by implemented with any neural network architecture. In our experiments we use the *Siamese Network* architecture (Reimers & Gurevych, 2019) as the underlying model. The embedding space size $M = 768$ and the embeddings vectors are normalized such that $\|\phi(\cdot)\|_2 = 1$.

The training is done by selecting N positive and M negative examples $\{(q_i, c_i, y_i)\}_{i=1}^{N+M}$ where q_i represents the query and is the concatenation of the hypothesis h and some previous entailment step s_{t-1}, the value c_i is some premise from the corpus C, and y_i is a label given to the example. The N positive examples are taken from the golden entailment trees from training data, meaning that $c_i \in L$. We use the golden steps as queries and their respective leaf nodes as expected

premises. The negative examples are created by both randomly selecting premises in C or by using premises incorrectly retrieved by a non-fine-tuned version of the model (hard negatives).

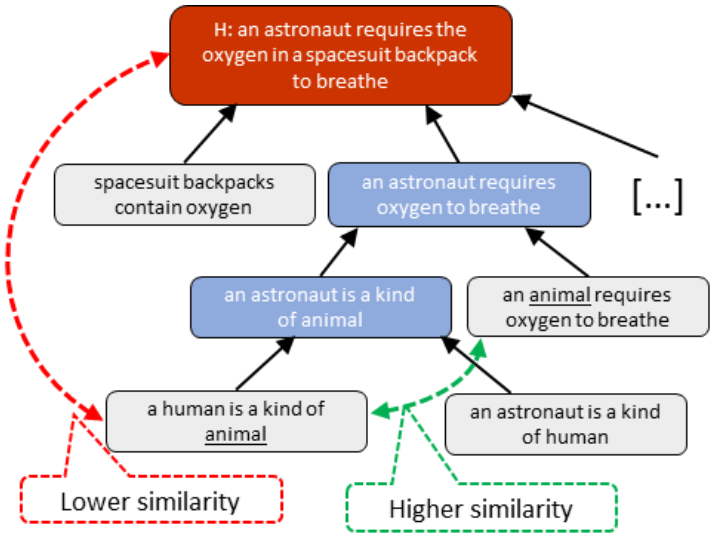

Figure 17: Entailment tree showing a challenging example when retrieving premises for the first entailment step. Some leaf nodes have lower similarity to the hypothesis (root) node.

The expected label y_i of an example depends on how close the leaf node $c_i \in L$ is from the intermediate step s_t in the golden entailment tree. Given that $ant(s_t)$ is the set of antecedents (or child nodes) of the entailment step s_t, then the value of the label y_i is given by:

$$y_i = \begin{cases} 0, & if\ c_i \notin L \\ \lambda, & if\ c_i \in L\ and\ c_i \notin ant(s_t) \\ 1, & if\ c_i \in L\ and\ c_i \in ant(s_t) \end{cases} \quad (17)$$

Which means that $y_i = 1$ if it is a positive example of a premise that is directly used by the entailment step, or $y_i = \lambda$ if the premise is in the entailment tree but is not directly related to the entailment step s_t. Essentially, the value $\lambda \in [0{:}\,1]$ gives lower priority to leaf nodes not relevant to the current entailment step (in our experiments we use $\lambda = 0.75$). More concretely, consider the entailment tree example from Figure 16. If "*int1: eruptions block sun light*" is the entailment

step s_t and "*sent1: eruptions produce ash clouds*" is the premise of interest c_i, then $y_i = 1$ since c_i is an antecedent (or child node) of s_t. On the other hand, if the premise is "*s3: plants die without*

Function: CONDITIONAL-RETRIEVAL
Input: Hypothesis h, corpus C, number of initial retrieved premises k_0.
Parameters: Conditioning factor ω.
Output: Retrieved premises S_1.

$Q \leftarrow \{h\}$ /* set of queries */
$S_1 \leftarrow \{\}$
for i **from** 0 **to** k_0 **do**
  $C' \leftarrow \{c \in C : c \notin S_0\}$
  if $i \geq \omega$ **then**
    $s_i \leftarrow argmax_{(c \in C')} P(c|Q)$
    $Q \leftarrow Q \cup \{s_i\}$
  else
    $s_i \leftarrow argmax_{(c \in C')} P(c|h)$
  end if
  $S_1 \leftarrow S_1 \cup \{s_i\}$
end for
return S_1

Figure 18: Conditional Retrieval Algorithm.

sunlight" then $y_i = \lambda$ since c_i is not a direct descendent of the node s_t. Finally, the encoder ϕ is then trained by minimizing the cosine similarity loss $\mathcal{L}_\phi$ which is given by:

$$\mathcal{L}_\phi = \frac{1}{N+M} \sum_{i=1}^{N+M} \left(y_i - \frac{\langle \phi(c_i),\ \phi(q_i) \rangle}{\|\phi(c_i)\|\ \|\phi(q_i)\|} \right) \tag{18}$$

6.2.2 Conditional Retrieval

The first retrieval step is a special case since there are no previously entailed steps, and the retrieval depends solely on h. When the entailment tree is deep, the retrieval is often more challenging because leaf nodes have lower semantic similarity to the hypothesis. A challenging case is depicted in Figure 17, where the leaf node "*a human is a kind of animal*" is very dissimilar relative to the hypothesis, "*An astronaut requires the oxygen in a spacesuit backpack to breathe*". To improve

the retrieval quality, we perform a conditional retrieval on the first step when $t = 1$, which consists of retrieving a set of premises using the hypothesis h, then concatenating h.

Input: h, **and** S_1 **for** $t = 1$

`hypothesis: Eruptions can cause plants to die;`
`sent1: eruptions produce ash clouds;`
`sent2: ash clouds block sunlight;`
`sent3: plants die without sunlight;`

Output: P_1 **for** $t = 1$
`sent1 & sent2 -> int1: Eruptions block sunlight;`
Input: h, S_1, **and** P_1 **for** $t = 2$

`hypothesis: Eruptions can cause plants to die;`
`sent3: plants die without sunlight;`
`sent1 & sent2 -> int1: Eruptions block sunlight;`

Output: P_2 **for** $t = 2$
`sent3 & int1 -> hypothesis;`

Table 6: Input and output example for the $IRGR\_generator$ module during generation for two iteration steps. The entailment tree structure is encoded using plain text.

with the partial retrieved premises $S'_t \subseteq S_t$ themselves. The conditional retrieval algorithm is shown in Figure 18.

The conditional retrieval algorithm assumes that leaf nodes (premises) are more similar among each other than to the root node (hypothesis). First the algorithm builds a set Q containing the sentences that will be used as queries. Assuming that k_0 premises will be retrieved, then the first ω retrievals use only the hypothesis h as the query. Afterwards, all the remaining retrieval results are saved in Q and are later concatenated to the hypothesis and are used as a query in order to retrieve more premises. In our experiments we set $\omega = 15$ and $k_0 = 25$.

Function: ITERATIVE-RETRIEVAL-GENERATION
Input: Hypothesis h, corpus C, number of initial retrieved premises k_0.
Parameters: Conditioning factor ω, maximum number of entailment steps t_{max}.
Output: Predicted entailment tree T.

$T \leftarrow$ "" /* entailment tree as empty string */
$P \leftarrow ()$ /* empty list of intermediate steps */
for t **from** 1 **to** t_{max} **do**
  $C' \leftarrow \{c \in C : c \notin S_0\}$
  if $t > 1$ **then**
    $S_t \leftarrow IRGR\_retriever(h, p_{t-1})$
  else
    $S_t \leftarrow$ CONDITIONAL-RETRIEVAL(h, C, k_0)
  end if
  $p_t \leftarrow IRGR\_generator(h, S_t, P_{1:t-1})$
  $P \leftarrow P \oplus (p_t)$
  $T \leftarrow$ UPDATE-ENTAILMENT-TREE(T, p_t, S_t)
  if $h \in p_t$ **then**
    return T
  end if
end for
return T

Figure 19: Iterative Retrieval and Generation Reasoner Algorithm.

6.2.3 Explanation Generation

The proposed $IRGR\_generator$ module outputs one entailment step per iteration. It is built on an encoder-decoder language model and handles the structured nature of entailment trees by encoding the edges and nodes as plain text. Both the input and output trees are encoded using any valid topological order, from leaves to the root node (i.e., the hypothesis node h). We follow Dalvi et al. (2021) and encode the leaf nodes $s_i \in S$ using the symbol "`sent`", the inner nodes $e_i \in E$ are using the symbol "`int`" and the root node h uses the symbol "`hypothesis`". For instance, the input and output of the $IRGR\_generator$ for the example in Figure 16 at iterations $t = 1$ and $t = 2$ is shown below in Table 6. In this example, it is assumed that the corpus consists of only the leaf nodes from that specific entailment tree (i.e., `s1`, `s2` and `s3`). Note that a leaf node already used

in the previously generated entailment steps $P_{1:t-1}$ are removed from the input context and when the model produces an entailment step that has the hypothesis as conclusion, then the whole iterative retrieval and generation process stops. The algorithm combining the retrieval and generation steps is shown in Figure 19. The UPDATE-ENTAILMENT-TREE sub routine will update the string representing the entailment tree T similar to the updates shown in Table 6. The $\oplus$ operator is used to represent concatenation of lists.

The generator module is based on the Text-to-text Transfer Transformer (or T5) by Raffel et al. (2020) and is trained in an auto-regressive fashion, where the model predicts future values from past predicted values. The training examples are generated from the golden entailment trees, where an example is generated for each entailment step. We select the model weights with the highest number of perfectly generated entailment trees (the Overall All-Correct metric described below) on the development set to be used for tests.

6.3 Experiments and Results

In this experiment, we evaluate how well our system can generate truthful explanations that can show the entailment of a hypothesis from a set of textual premises. We evaluate our system on the ENTAILMENTBANK dataset (Dalvi, et al., 2021), where the hypothesis are accompanied by answer-question pairs taken from the ARC grade school science dataset (Clark, et al., 2018), while the textual premises are part of the WorldTree V2 corpus (Jansen, Wainwright, Marmorstein, & Morrison, 2018). The ENTAILMENTBANK dataset contains 1,840 questions with a total of 5,881 entailment steps. The corpus of premises C will contain the sentences from the WorldTree V2 corpus plus a few additional premises created by the ENTAILMENTBANK annotators and contains around 11 thousand sentences covering various grade-school level science topics.

6.3.1 Implementation Details

When running experiments, we used machines with four Tesla V100 GPUs, each with 16GB of VRAM memory. For the retrieval module, we use the Sentence Transformer (Reimers & Gurevych, 2019) library. We fine tune the `all-mpnet-base-v2`[8] encoder, which is based on the MPNet language model (Song, Tan, Qin, Lu, & Liu, 2020) and fine-tuned on over one billion training sentence pairs, ultimately designed to be general purpose models. The generator module used the HuggingFace's Transformers (Wolf, et al., 2020) implementation of the `t5-large`[9], which is an encoder-decoder model with 770 million parameters and 512 tokens of context size limit. The T5 model itself was pre-trained on the "Colossal Clean Crawled Corpus" with mixture of unsupervised and supervised tasks where each task is converted to text-to-text format. Further implementation details with training hyper parameters and settings are available in Appendix A.6.

6.3.2 Retrieval Evaluation

Before evaluating the whole IRGR pipeline, we test the $IRGR\_retriever$ separately to understand how our proposed approach impacts the retrieval of premises. We chose two different metrics to evaluate the system and baselines. The first metric is the well-known information retrieval metric *recall at k* (or R@k) which computes the recall given the top-k retrieved entries from the corpus. Given the set of retrieved premises $S \subseteq C$ and the set of golden (or expected) premises $S^* \subseteq C$ such that $|S| = k$, then the R@k is given by $|S \cap S^*|/|S^*|$. The second metric, called *All-Correct* is a strict metric that expects that all the premises from S^* are retrieved from the set of retrieved premises. Therefore, the value of this metric is equal to 1 if and only if $S^* \subseteq S$. For our experiments

[8] Available at https://huggingface.co/sentencetransformers/all-mpnet-base-v2
[9] Available at https://huggingface.co/t5-large

we set $k = 25$, since this is roughly the number of premises that can fit inside the generator module limited input context.

Two approaches are used as baselines. The first uses the python implementation of the Okapi BM25 ranking function[10], which is commonly used to estimate the relevance of documents given a query. This ranking function uses bag-of-words and ranks entries in a corpus according to the frequency of occurrences of query terms in such entries. The second baseline uses the retrieval module from EntailmentWriter, which is based on Tensorflow-Ranking-BERT (Han, Wang, Bendersky, & Najork, 2020) and constructed on top BERT representations.

In addition to these baselines, we include three variations of the $IRGR\_retriever$ module. The first performs the conditional retrieval algorithm as shown in Figure 18. The second IRGR variation ("*w/o conditional ret.*") does not rely on the conditional retrieval, only using the hypothesis h as the input query to retrieve premises. The third and final variation ("*full pipeline*") tries to simulate the full pipeline where retrieval is combined with generation through multiple iterations. It performs conditional retrieval for the first iteration, and then uses the intermediate nodes from the golden entailment tree for the remaining iterations. Note that this variation retrieves more than 25 premises total, but we include these results as an upper bound performance for the IRGR system as a whole.

The retrieval evaluation results for both R@25 and All-Correct are shown in Table 7. The results show that our retrieval module outperforms the baselines by a large margin. Notably using the conditional retrieval improves the IRGR's R@25 results by 6.0% and All-Correct results by 6.8%, showing that retrieving premises using only the hypothesis is not as effective as using the hypothesis plus other intermediate retrieved premises as the query.

[10] Available at https://pypi.org/project/rank-bm25/

Method	R@25	All-Correct
Okapi BM25	45.01	22.35
EntailmentWriter	59.76	34.70
IRGR	**68.28**	44.70
- w/o conditional ret.	64.41	40.29
- full pipeline*	—	**51.47**

Table 7: Results for the retrieval module. Methods with * retrieve more than 25 premises.

6.3.3 Entailment Tree Generation Evaluation

We follow the same metrics defined by Dalvi et al. (2021) when evaluating our system on entailment trees generation. These metrics compare the predicted entailment tree $T = (h, S, E, P)$ with the golden entailment tree $T^* = (h, S^*, E^*, P^*)$ from the ENTAILMENTBANK dataset. The evaluation metrics use an algorithm to find a mapping between the nodes from T and T^*, similar to the metrics from Inoue, Stenetorp, & Inui (2020). The algorithm gathers the ancestor leaf sentences for each intermediate node $p \in P$ and $p^* \in P^*$, then uses Jaccard similarity on the set of leaf sentences to find the best pairings. Any predicted intermediate node with zero similarity to any golden intermediate nodes gets assigned to a dummy empty node. The metrics evaluate the correctness of generated entailment trees according to four categories, testing if the model (1) used the correct leaf nodes (2) generated correct entailment steps (3) generated reasonable sentences for the intermediate conclusions (4) overall correctness of the entailment tree. For each of these four categories, we compute *F1* metrics *All-Correct* metrics. The All-Correct metric is more restrictive and is only equal to 1 if and only if F1 = 1, otherwise it is set to 0. The metrics are summarized below:

Method	Leaves		Steps		Intermediates		Overall
	F1	All-Cor.	F1	All-Cor.	F1	All-Cor.	All-Cor.
EntailmentWriter	39.9	3.8	7.4	2.9	35.9	7.1	2.9
IRGR	45.6	**11.8**	**16.1**	**11.4**	**38.8**	**20.9**	**11.5**
- w/o iter.	**46.6**	10.0	11.3	08.2	38.7	**20.9**	08.2
- w/o iter. & cond.	36.1	3.8	6.1	3.2	30.5	10.3	3.2

Table 8: Scores for our proposed IRGR method on the EntailmentBank dataset. Each produced entailment tree is evaluated among four dimensions on the test set. The F1 scores measure the overlap between predicted and golden data. On the other hand, for each data point the all-correct (abbreviated to "All-Cor.") metrics are one when the F1 score is equal to one, and zero otherwise.

Task	Method	Leaves		Steps		Intermediates		Overall
		F1	All-Cor.	F1	All-Cor.	F1	All-Cor.	All-Cor.
Gold	EntailmentWriter*	98.7	86.2	50.5	37.7	67.6	36.2	33.5
	IRGR	99.6	97.6	51.1	37.6	66.8	34.1	32.1
Gold + Distr.	EntailmentWriter*	84.3	35.6	35.5	22.9	61.8	28.5	20.9
	IRGR	69.9	23.8	30.5	22.3	47.7	26.5	21.8

Table 9: Scores for our proposed IRGR method on the EntailmentBank dataset for task not requiring retrieval. The "Gold" task contains all gold leaf nodes, while "Distr." task contains gold leaf nodes and distractors randomly selected from the corpus. The EntailmentWriter* reported has equivalent model size to IRGR.

- **Leaves (F1, All-Correct):** Tests if the set of predicted leaf nodes S matches the set of golden leaf nodes S^*.
- **Steps (F1, All-Correct):** Tests if entailment steps follow the correct tree structure by comparing set of premises of the paired nodes p and p^*.
- **Intermediates (F1, All-Correct):** Tests if sentences of the generated intermediate conclusions are correct. This is done by computing the textual similarity between the conclusions $e \in E$ and $e^* \in E^*$ from the paired nodes p and p^*. The textual similarity uses

$BLEURT$ (Sellam, Das, & Parikh, 2020), a learned evaluation metric based on BERT. Whenever $BLEURT(e, e^*) \geq 0.28$ then the generated intermediate conclusions are considered correct.

- **Overall (F1, All-Correct):** The overall correctness, meaning that all other metrics have All-Correct equal to one.

We use EntailmentWriter (Tafjord, Dalvi, & Clark, 2021) as a baseline, which is an encoder-decoder model that learns to output the entailment trees as linearized texts. EntailmentWriter uses the 11 billion parameter version of T5, which is around one order of magnitude larger than the T5 model used in IRGR. As shown in Table 8, our method outperforms the baselines on all the metrics. Noticeably, the "Overall All-Correct" metric has an increase in value of over 300.0%. The gains are mostly due to IRGR's improved retrieval-generation capabilities, as suggested by the ablation results "- *w/o iter. & cond.*" (when IRGR does not iteratively retrieve premises or perform conditional retrieval), which performs similarly to EntailmentWriter. Furthermore, the "- *w/o iter.*" results (when IRGR does not iteratively retrieve and generate) are also better than EntailmentWriter and serve as further evidence that conditional retrieval is helpful when finding premises in the corpus.

To evaluate if the iterative generation alone has any impact in the results, we run experiments on two other tasks that do not require retrieval. In the "Gold" task the leaf nodes are taken from the golden entailment trees. The "*Gold + Distr.*" task mixes the leaf nodes from the golden entailment trees with some distractor premises taken at random from the corpus. The experiment results are shown in Table 9. In this table the results are for a version of EntailmentWriter that has the same number of parameters as IRGR, where they both use the `t5-large` model. The results

show negligible differences in the "Overall All-Correct" metrics, which means that iterative generation can produce accurate entailment trees when compared to generation in a single pass.

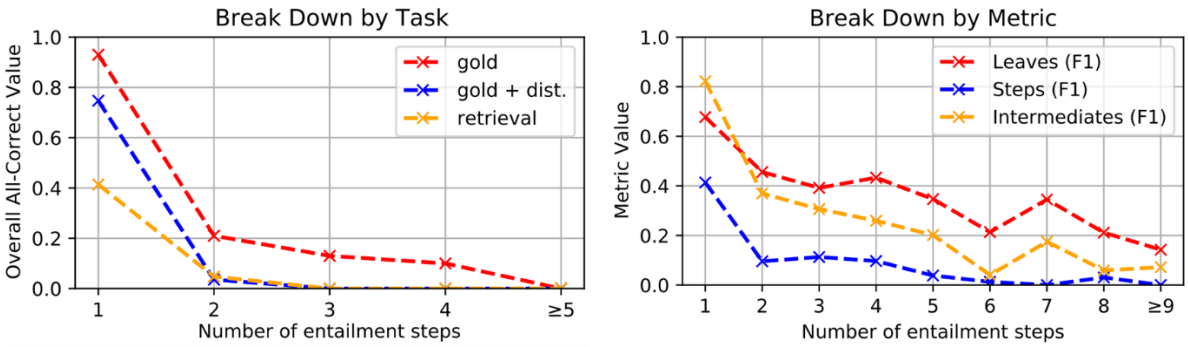

Figure 20: Results broken down by number of entailment steps in the golden entailment tree.

To understand how the size of the entailment tree influence the generation results, we plot in Figure 20 the results for different metrics and tasks according to the number of entailment steps in the golden entailment trees. The graph on the left-hand side shows the All-Correct metric for the three evaluation tasks ("*gold*", "*gold + distractors*", "*retrieval*"), while the graph on the right-hand side shows the F1 values (for "*leaves*", "*steps*", and "*intermediate*" metrics) for the "*retrieval*" task. The results indicate that the problem becomes more challenging as the number of entailment steps increases. For instance, the IRGR model cannot perfectly predict any entailment tree with five or more entailment steps. This phenomenon is likely due to the nature of the problem, where errors are compounded across entailment steps and a single mistake will cause the predicted entailment tree to receive an All-Correct score of zero.

6.3.4 Retrieval Error Analysis

To obtain some insights on the types of retrieval errors made by the $IRGR\_retriever$ module on the ENTAILMENTBANK dataset, we fetch 25 premises for each data point in the development set.

These premises are split into a set of true positives (correctly retrieved) and false negatives (expected premises that failed to be retrieved). We compute the number of overlapping unigrams and bigrams between the hypothesis and the premises in these two sets. The numbers show that true positives contain 28.5% more unigrams overlap and 68% more bigrams overlap to the hypothesis when compared to false negatives. This suggests that the retrieval module struggles with premises that are syntactically *dissimilar* to the hypothesis. Furthermore, we investigate how the tree depth (number of nodes in the path from the tree node to the leaf node) influences retrieval results. We notice that true positive nodes have an average tree depth of 2.3 when compared to the average depth of 3.0 of false negatives, which indicates that leaf nodes for larger entailment trees tend to be harder to retrieve.

6.3.5 Generation Error Analysis

To identify issues and possible improvement avenues, we manually annotate 50 predicted entailment trees generated by IRGR that differ from the golden entailment tree. We categorize these errors as follows: **Incorrect or Missing Leaves** (52%) are caused by errors in the retrieval of premises from the corpus. **Invalid Entailment Steps** (32%) happen when the generated conclusion (or intermediate nodes) is incorrect and should have not been entailed from the premises. Among those errors there are cases when the IRGR generation module simply copies one of the premises as conclusion, which might be related to hallucination problems associated with generative language models (Ji, et al., 2023). **Imperfect Evaluation** (12%) are cases where the generated entailment tree differs from the golden entailment tree but still had reasonable premises and entailment conclusions. **Disconnected Entailment Trees** (4%): is when the final output does not form a tree structure or follows the specified output format.

6.4 Conclusion

It is desirable that language systems output human-interpretable explanations instead of simply giving a discrete answer to a question. In this Chapter, we propose IRGR, a new neural network architecture that can generate structured natural language explanations in the form of entailment trees, showing how a hypothesis follows from a given set of retrieved premises. The IRGR architecture iteratively retrieves premises while generating inferences from such premises, one step at a time. We show that this approach has advantages over previous baselines, where the retrieval of premises can leverage intermediate conclusions generated by the model.

7 Structured Reasoning and Explanation Benchmark

Some recent question-answering research has demonstrated that it is possible to use generative language models to perform reasoning directly over natural language input (Clark, Tafjord, & Richardson, 2021). Furthermore, it was shown that producing an intermediate natural language *chain-of-thought* can improve very large language models' accuracy when answering questions in a few-shot prompt settings (Wei, et al., 2022). Several datasets and benchmarks were proposed to evaluate the reasoning and explanation capabilities of such systems. Most noticeably, the ENTAILMENTBANK dataset (Dalvi, et al., 2021) discussed in Chapter 6 contains multi-step explanations in the form of entailment trees. However, ENTAILMENTBANK has three shortcomings. First, it requires the creation of a set of textual *hypotheses* that represent the question-answer pair and are used as the root of entailment trees. Second, it requires a hand-crafted corpus of textual facts. Third, it only contains questions from a single domain, namely science exam questions.

To bridge the gap in resources for training and evaluating general multi-step reasoning capabilities over natural language, we introduce in Ribeiro et al. (2023) the ***ST****ructured* ***RE****asoning and* ***E****xplanation Multi-****T****ask* benchmark (or STREET for short). The STREET benchmark is a multi-task and multi-domain dataset. It contains a collection of problems involving quantitative reasoning (math questions), analytical reasoning (logic puzzle questions) and commonsense reasoning (science and process understanding questions). We build upon existing QA datasets by creating annotations that explain *how* an answer can be derived from the information contained within the questions. These structured multi-step and multi-premise explanations are named *reasoning graphs*, as shown in Figure 21. There is a total of 35.8 thousand questions in STREET and each question is annotated with a reasoning graph. When combined, all the reasoning graphs

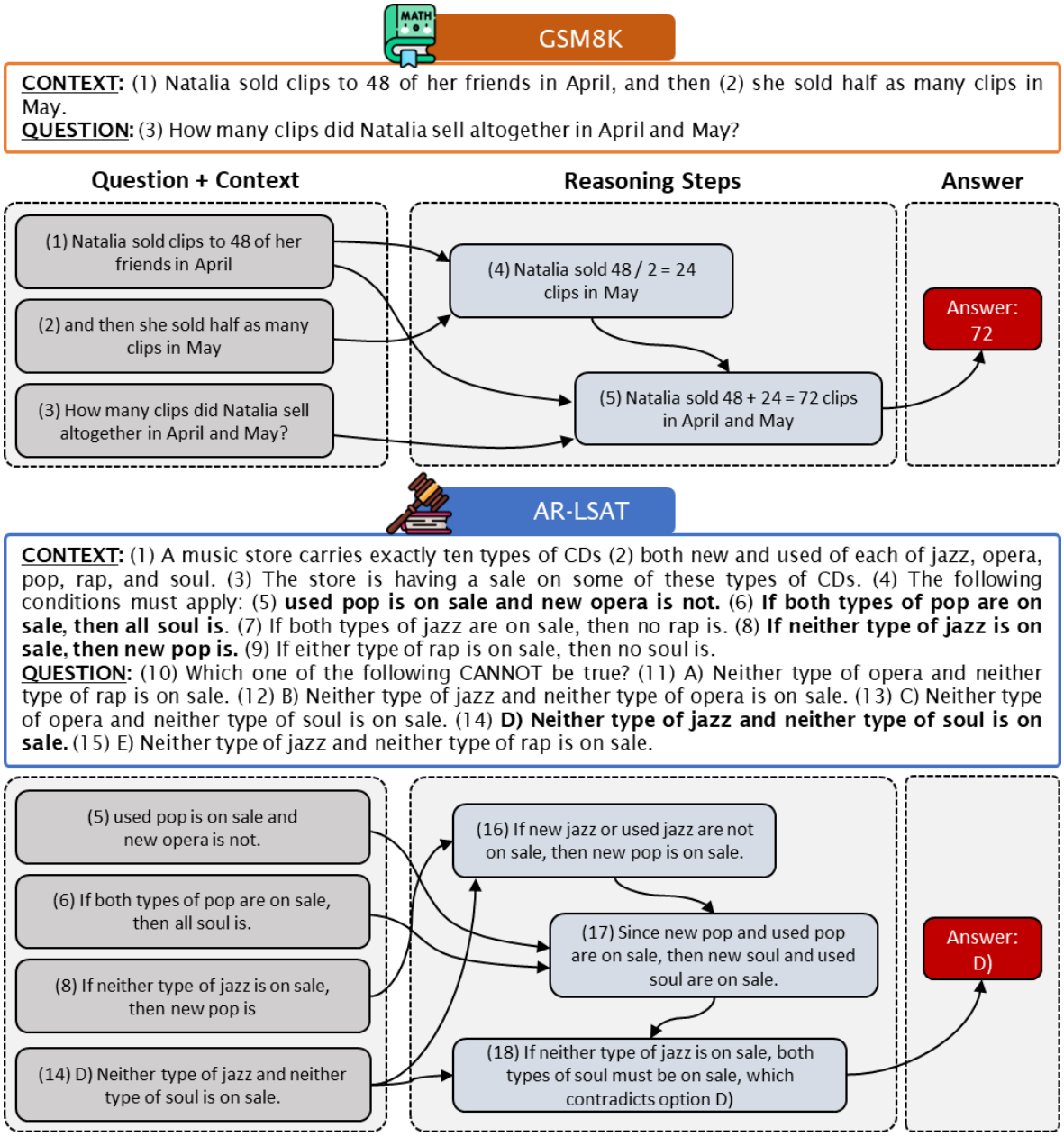

Figure 21: Examples of questions, explanations, and answers from the STREET benchmark. The questions are taken from the Grade School Math (GSM8K) dataset and from the Analytical Reasoning – Law School Admission Test (AR-LSAT). The reasoning graphs containing structured step-by-step explanations for the given answers are created by our expert annotators. The question, explanation and answers are broken down into textual logical units (or TLUs), and form nodes with entailments expressed as directed edges.

in STREET amount to a total of 151.1 thousand reasoning steps (or multi-premise textual entailments), of which 14.7 thousand were generated by human annotators. All the tasks in STREET were carefully selected such that no entity-centric knowledge (e.g., details such as "What film won the most Oscars in 1992?") needs to be retrieved, and most of the information is contained within the problem description itself. Therefore, we add more emphasis on the reasoning problem, with an average of 7.8 reasoning steps per question-answer pair and more complex reasoning structures than previous datasets.

We evaluate a fine-tuned version of T5 (Raffel, et al., 2020) and few-shot prompting version of GPT-3 (Brown, et al., 2020) on the STREET data. Given a question, the models are expected to output the answer together with the corresponding reasoning graph. Our proposed metrics evaluate the correctness of the predicted reasoning graphs compared to the expected golden data in terms of the generated conclusions and the structure of the graph. Interestingly, the experiment results show that even when achieving high answer accuracy, these baseline models still struggle to produce coherent reasoning graphs and perform much worse than humans (around 200% worse graph similarity scores) on more challenging STREET tasks.

7.1 Task Definition

The tasks in STREET can be formally defined as follows. Consider the standard Machine Reading Comprehension (MRC) task $T = (Q, C, O, A)$, consisting of a question Q, an optional context C containing a passage or question details, a list answer options $O = \{o_1, \dots, o_K\}$ in the case of K-way multiple choice questions, the expected answer A, where $A \in O$ if answer options are provided. Some MRC tasks (Ling, Yogatama, Dyer, & Blunsom, 2017; Camburu, Rocktäschel,

Lukasiewicz, & Blunsom, 2018) will also include a rationale R, which is a plain (unstructured) textual explanation for the answer A.

To build the reasoning graphs we break down all the textual content from the MRC components into chunks, which we call *textual logical units* (or TLUs). The TLUS are essentially a consecutive subsequence of tokens from the elements in T that are used as premises for the entailments in the reasoning graph. Formally, assume some component $\Psi \in T$ from the MRC task with tokens $\Psi = (t_1, \ldots, t_{|\Psi|})$. The TLUs of Ψ are the list of token spans $TLU(\Psi) = ((\mathrm{t}_1, \ldots, \mathrm{t}_{b_1}), (\mathrm{t}_{b_1}, \ldots, \mathrm{t}_{b_2}), \ldots, (\mathrm{t}_{b_{(|TLU(\Psi)|-1)}}, \ldots, \mathrm{t}_{|\Psi|}))$ such that the TLU span boundaries follow $1 \leq b_i \leq |\Psi|$ and for any two indexes $b_i < b_j$ then $i < j$. Examples of TLUs from the GSM8K question in Figure 21 are "*Natalia sold clips to 48 of her friends in April*" and "*Natalia sold 48 + 24 = 72 clips in April and May*". The TLUs are automatically extracted by breaking down paragraphs or sentences using boundaries such as punctuation marks or key tokens such as "*and*" or "*then*". The pseudocode of the algorithm used is shown in Appendix B.2.

Each task T will be accompanied by a reasoning graph which is encoded as a *directed acyclic graph* data structured. The reasoning graphs G can be formally defined as a set of vertices and edges $G = (V, E)$ such that the set of vertices $V \subseteq (TLU(Q) \cup TLU(C) \cup TLU(O) \cup TLU(A) \cup TLU(R))$ and the set of edges $E \subseteq V \times (TLU(O) \cup TLU(A) \cup TLU(R))$ contain pair of nodes. The direction of the edges always points from the premise nodes (or antecedents) to an intermediate conclusion node. All the premises and edges connected to a conclusion node represent an entailment or reasoning step. Note that nodes and edges in the Reasoning Graph don't contain any other attached labels, which means that logic concepts such as negation, conjunction and disjunction must be explicitly stated in the text of the TLU as in "*If new jazz **or** used jazz are **not** on sale, **then** new pop is on sale*".

TASK NAME	TASK DOMAIN	# ORIGINAL QUESTIONS	# USED QUESTIONS	# REASONING STEPS	ANSWER TYPE
ARC	Science	7,787	1,840	5,881	4-Way MC
SCONE	Processes	14,574	14,574	130,482	State Pred.
GSM8K	Math	8,792	1,030	4,666	Number
AQUA-RAT	Math	101,449	1,152	7,179	5-Way MC
AR-LSAT	Logic	2,046	500	2,885	5-Way MC
TOTAL	---	134,648	19,096	151,093	---

Table 10: A summary of the tasks in the STREET benchmark. The questions with multiple choice answers are labeled “K-Way MC”, which stands for “K-way multiple choice”.

7.2 STREET Dataset Details

The STREET benchmark contains a total of 35.8 thousand questions in various domains. A reasoning graph is provided along with each question-answer pair. With all reasoning graphs combined, the benchmark contains 151.1 thousand reasoning steps (or textual entailments), of which 14.7 thousand were created by expert annotators while the remaining were generated programmatically. A summary of STREET’s five tasks can be found in Table 10.

The question-answer pairs are taken from the following five existing datasets: the AI2 Reasoning Challenge or ARC (Clark, et al., 2018), Sequential Context-Dependent Execution dataset or SCONE (Long, Pasupat, & Liang, 2016), Grade School Math or GSM8K (Cobbe, et al., 2021), Algebra Question Answering with Rationales or AQUA-RAT (Ling, Yogatama, Dyer, & Blunsom, 2017), and Analytical Reasoning – Law School Admission Test or AR-LSAT (Zhong, et al., 2022).

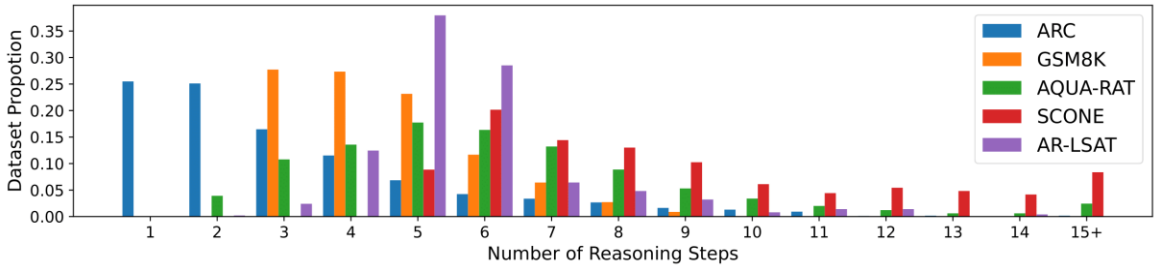

Figure 22: Proportion of reasoning graphs in each STREET task according to the number of reasoning (or entailment) steps.

For the first two datasets (ARC and SCONE) the reasoning graphs are programmatically extracted from the intermediate states provided by the original datasets. The ARC dataset contains multiple-choice grade school science questions about various topics including geography, biology and astronomy. We include the data provided by ProPara (Dalvi, et al., 2021) to build the reasoning graphs, where the premises don't need to be retrieved and will be provided as part of the MRC context C. On the other hand, the SCONE dataset contains questions about processes and how a toy world evolves over time. There are three subtasks in SCONE, each starts with a description of a toy world. In SCONE's *Alchemy* subtask, a set of beakers containing chemicals of different colors are manipulated (e.g., "*pour the content of the third beaker into the last beaker, then mix it.*"). In SCONE's *Scene* subtask, there are a set of positions, some of which people with different outfits are standing on, and a sequence of movements occur (e.g., "*a man in a red hat moves to the right of the man with a blue shirt*"). In SCONE's *Tangrams* subtask has different figures that aligned sequentially, then some of these figures move or are removed from the list (e.g., "*swap the 1st and 5th figure*"). In the end, the initial toy world will change, and the system is expected to predict the final state of the world.

The third and fourth datasets (GSM8K and AQUA-RAT) contain grade school and high school level math questions. Only unstructured explanations (or rationales) are provided in the original datasets, and we convert them into reasoning graphs by identifying the fact nodes and connecting them using entailment edges. The final dataset (AR-LSAT) contains complex analytical reasoning questions that involve multiple entities and rules described in its context. For ARLSAT we create the reasoning graphs from scratch. We ask expert annotators to create a step-by-step textual explanation of the answer given to the question and the context paragraph. As shown in Figure 22, the STREET benchmark contains a greater number of reasoning steps than most of the existing reasoning and explanation datasets with an average of 7.8 reasoning steps per answer. In contrast, most multi-hop QA datasets contain two or three steps per question (Yang, et al., 2018).

7.2.1 Annotation Details

Since some of the STREET questions are relatively challenging to solve, we chose expert annotators with undergraduate or graduate level education as opposed to randomly selected online workers. Furthermore, human annotators were given an usnlimited amount of time to complete each annotation task. For quality control we perform two annotation passes for each reasoning step and a third pass to break ties when needed. We use these extra annotations to compute annotation agreement using Fleiss Kappa κ (Fleiss, 1971), where each directed edge inside a reasoning graph is treated as a binary question. The computed value of $\kappa = 0.79$ indicates *substantial agreement* among annotators. We estimate that a total of 1,467 (paid) work hours were required to annotate the STREET data. Further annotation and dataset details including examples for each STREET task and annotation user interface can be found in Appendix B.1 and Appendix B.3.

```
$question$ = (1) Natalia sold clips to 48 of her friends
in April, and then (2) she sold half as many clips in
May. (3) How many clips did Natalia sell altogether in
April and May?

$proof$ = (1) & (2) -> (4): Natalia sold 48/2 = 24 clips
in May; (1) & (3) & (4) -> (5): Natalia sold 48+24 = 72
clips altogether in April and May; (3) & (5) -> (6): The
answer is 72;
```

Figure 23: Example of textual encoding of a reasoning graph from GSM8K task.

7.3 Experiments

7.3.1 Baseline Models

We use two popular generative language models as baselines that take as input the questions broken down in textual logical units (TLUs)s and output the reasoning graph in textual format. We choose generative language models as baselines since they have been shown to have relatively strong reasoning capabilities (Wei, et al., 2022; Wang, et al., 2023). The first baseline uses the T5 [`large`] with 770 million parameters which is fine-tuned on the full training data for each STREET task. The second baseline uses GPT-3 [`davinci`], more specifically the `text-davinci-002` with 175 billion accessed through OpenAI's API[11]. The GPT-3 [`davinci`] model takes as input only a few input examples (few-shot learning) instead of fine-tuning on STREET data. Further fine-tuning and few-shot learning details are shown in Appendix A.7.

The textual encoding of reasoning graphs follows a similar approach to (Dalvi, et al., 2021). First, we assign consecutive number IDs for each TLU as in "`(X)`" (where `X` is a number) for the

[11] Available at: https://platform.openai.com/docs/models/gpt-3

context, question and answer options. Afterwards, all the reasoning steps are sorted according to some valid topological order such that any intermediate conclusions will appear before other intermediate conclusions that have them as ancestor nodes. These reasoning steps are also assigned consecutive number IDs. The entailments from the reasoning step are encoded as "`(X) & (Y) -> (Z)`", assuming that the TLUs with id "`(X)`" and "`(Y)`" are the antecedents of the TLU with id "`(Z)`". To illustrate this encoding method, the textual representation of the GSM8K example from Figure 21 is shown in Figure 23.

7.3.2 Evaluation Metrics

The baseline models are evaluated according to both the predicted answer A_p and predicted reasoning graph G_p. We propose three main metrics which are described below.

Answer Accuracy: The first metric evaluates if the predicted answer A_p is the same as the golden answer A_g. For the tasks with multiple-choice or numerical answers, the answers are compared using exact match, meaning that $A_p = A_g$. For SCONE the predicted answer is a concatenation of the end state of each object tracked by the process.

Reasoning Graph Accuracy: The second metric compares the predicted reasoning graph $G_p = (\mathrm{V}_p, \mathrm{E}_p)$ and golden reasoning graph $G_g = (\mathrm{V}_g, \mathrm{E}_g)$ in terms of both graph structure and the content of the intermediate conclusion nodes. It starts by aligning the nodes from V_p and V_g using the nodes' premises as anchors. Given two matched reasoning step nodes $\mathrm{v}_p \in \mathrm{V}_p$ and $\mathrm{v}_g \in \mathrm{V}_g$ then their textual values are compared using a similarity function $\sigma(\mathrm{text}(\mathrm{v}_p), \mathrm{text}(\mathrm{v}_g))$ to determine if the nodes are equivalent. If any two aligned nodes differ in terms of their textual values or set of nodes

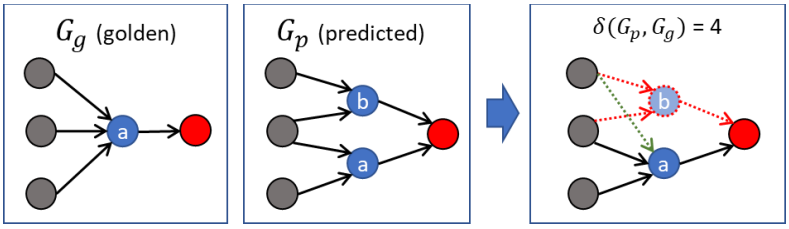

Figure 24: Edit distance between two reasoning graphs. Gray nodes are premises, blue nodes are intermediate conclusions, and the red node is the answer. Dotted red edges and dotted red nodes were removed, while the dotted green edge was used as a substitution to one of the removed edges.

from their antecedents, then the reasoning graph accuracy is set to zero, otherwise it is set to one. The textual similarity function σ varies depending on the STREET task being evaluated. The details regarding the matching algorithm and similarity functions used are shown in Appendix B.4. Note that small dissimilarities between G_p and G_g can cause the value of this metric to be zero, which means this is a relatively strict metric.

Reasoning Graph Similarity: The last metric is a "softer" version of Reasoning Graph Accuracy and uses a graph edit distance $\delta(G_p, G_g)$ function to compute the similarity between the predicted and golden reasoning graph. The allowed operators in the graph edit distance function δ are *insertion*, *deletion* and *substitution*, where each operation has a fixed cost equal to one. The textual similarity function σ is used to test if two nodes are equal. The Reasoning Graph Similarity metric $sim(G_p, G_g)$ is normalized (meaning that $sim(G_p, G_g) \in [0: 1]$) and computed as follows:

$$sim(G_p, G_g) = 1 - \left[\frac{\delta(G_p, G_g)}{\max(|\mathrm{V}_p| + |\mathrm{E}_p|, |\mathrm{V}_g| + |\mathrm{E}_g|)}\right] \quad (19)$$

We add a special case when computing the graph reasoning similarity and assign $sim(G_p, G_g) = 0$ if the answers don't match $A_p \neq A_g$. We show an example in Figure 24, illustrating the edit distance value for a small graph. In that specific example, the reasoning graph

Method	ARC	SCONE	GSM8K	AQUA-RAT	AR-LSAT
Answer Accuracy					
Random Guess	17.1	---	---	20.0	20.0
T5 [large] (fine-tuned)	93.5	69.6	10.4	28.7	28.0
GPT-3 [davinci] (few-shot)	20.8	02.3	34.8	30.2	19.0
Reasoning Graph Accuracy					
T5 [large] (fine-tuned)	17.1	60.0	00.7	00.0	00.0
GPT-3 [davinci] (few-shot)	01.7	01.2	00.7	00.0	00.0
Graph Similarity					
T5 [large] (fine-tuned)	44.1	67.0	05.4	00.9	00.3
GPT-3 [davinci] (few-shot)	15.1	01.9	16.0	05.2	01.1

Table 11: Results of different models across all five defined reasoning tasks. Numbers in percentage. The "Random Guess" results are included to facilitate visualization since different tasks have different answer types.

similarity can be computed as $sim(G_p, G_g) = 1 - 4/\max(6+7, 5+4) \cong 0.6923$. In general, computing the edit distance between any two graphs is computationally expensive as the problem is NP-Complete (Abu-Aisheh, Raveaux, Ramel, & Martineau, 2015; Abu-Aisheh, Raveaux, Ramel, & Martineau, 2018). To circumvent this issue, we compute an approximation of this value using the edit distance implementation from the `networkx` python library[12].

7.3.3 Results

The main results are shown in Table 11. The results for SCONE are averaged across the different sub tasks (namely Alchemy, Scene, and Tangrams). From the table we can see that T5 [`large`] outperforms or is on par with GPT-3 [`davinci`] on both ARC and SCONE, while the opposite is true for GSM8K and AQUA-RAT. Even though T5 [`large`] is a much smaller model relative to GPT-3 [`davinci`], we believe that there are enough examples in ARC and SCONE that the

[12] https://networkx.org/

fine-tuning gives the T5 [large] model an edge. On the other hand, it seems that both baselines struggle with AR-LSAT and their results are not much better than the random guess results. The AR-LSAT results are consistent with Zhong et al. (2022) which has demonstrated that generative models still struggle with more complex analytical reasoning tasks.

It is interesting to see that for certain tasks the Reasoning Graph Similarity metric has similar values to the Answer Accuracy metric. For instance, the Reasoning Graph Similarity for T5 [large] on SCONE is 96.2% of the Answer Accuracy value, which means that whenever the model outputs the right answer it also produces a reasonable multi-step explanation. Conversely, the Reasoning Graph Similarity for T5 [large] on AQUA-RAT is only 2.9% of the Answer Accuracy, which means that the predicted reasoning graph does not align well with the human annotated data. This is evidence that sometimes a model can predict the correct answer but can't generate a reasonable explanation for that answer.

To better understand the quality of the predicted reasoning graphs generated by the baselines, we plot the Reasoning Graph Similarity values for questions which were accurately answered in Figure 25. In addition to the baseline results, we evaluate the human performance on a subset of data for each STREET task. To obtain human results, we ask other human annotators with graduate level education to build the reasoning graphs from scratch for 100 randomly sampled questions from the test set, where they are also given the correct answer to the questions.

The results show that on both SCONE and GSM8K, the baseline models perform on a similar level to the estimated human performance, where T5 [large] is even able to outperform human results on the SCONE task. We believe this is the case because SCONE questions are based on a synthetic world and the reasoning steps required to answer questions are more standardized across data points, which allowed fine-tuned models to accurately predict the reasoning graphs.

On the other hand, both baseline models perform significantly worse than the estimated human performance on AQUA-RAT and AR-LSAT, where human-generated reasoning graphs achieve 200% higher reasoning graph similarity compared to the language models. Nonetheless, the overall reasoning graph similarity for AQUA-RAT and AR-LSAT are relatively lower than the other tasks. This could be due to the fact that for these tasks there are more valid outputs (there are multiple ways one can explain the answer to a question). Therefore, we believe that automatic evaluation of explanations can still be a challenge, which is a phenomenon also seen in other natural language generation fields (Celikyilmaz, Clark, & Gao, 2020).

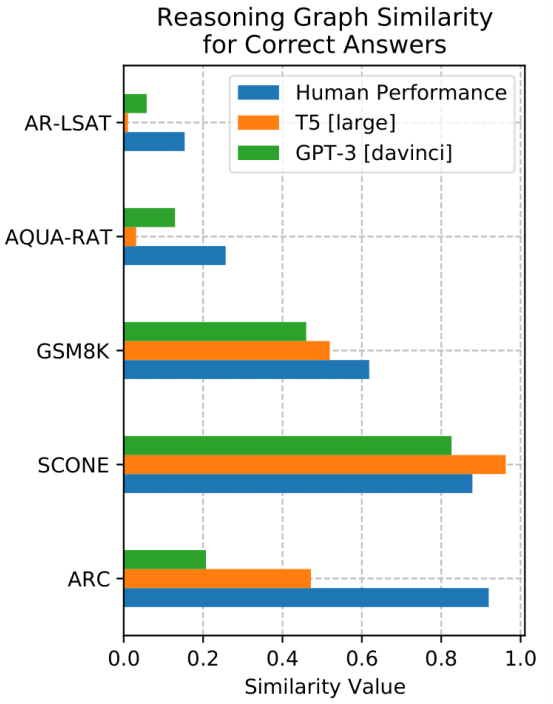

Figure 25: Histogram with Reasoning Graph Similarity of baseline models for each STREET task. Results are compared with human performance on a randomly selected subset of the test data.

7.4 Conclusion

In this Chapter we introduce STREET, a multi-step and multi-premise reasoning and explanation benchmark. We build upon existing QA datasets from different domains and add annotated explanations to their answers in the form of *reasoning graphs*. These reasoning graphs show step-by-step how premises in the question's context can be used to infer intermediate conclusions which are then used to predict the final answer. We evaluated two generative language models on the STREET tasks (namely T5 [`large`] and GPT-3 [`davinci`]) and found that these models still struggle to generate coherent reasoning graphs when compared to humans for certain STREET tasks. We hope that STREET will pave way for future work on natural language reasoning and will serve as a valuable way to systematically evaluate explanations for complex questions.

8 Conclusions

In this dissertation we study how two prominent AI architectures can perform reasoning directly over natural language input. The first AI architecture, called Companion, is a cognitive architecture that integrates a general-purpose language parser with analogical learning capabilities and is able to achieve strong performance in language tasks from a relatively small number of training examples. The second AI architecture, called Transformers, is a neural model that relies on attention mechanism to learn universal language representations and can better capture long-range dependencies between words in the input sequence. We build upon these AI architectures while proposing new algorithms, modeling strategies and methods that are used to solve a variety of language tasks including process understanding, knowledge extraction, and analytical question-answering. In the following sections we show how our experiments and findings support the claims of this thesis and discuss open questions and possible avenues for future research.

8.1 Claims Revisited

[Claim A] Discrete actions and processes in natural language can be accurately modeled using qualitative reasoning and structured action representations.

In Chapter 4 we introduce a Step Semantics, a framework that take inspiration from qualitative reasoning and provides a set of representation conventions of discreate actions and changes in processes using OpenCyc and FrameNet concepts. When paired with analogical learning capabilities of Companion, this framework can be successfully applied to model processes from the ProPara dataset, a procedural text understanding dataset that keeps track of the state of entities of interest (called participants) described in the input paragraphs. We combine methods from analogical QA learning, which is used to predict state changes on a sentence level, with a dynamic

programming algorithm, which enforces commonsense constraints on a paragraph level. We show how our analogical system can answer ProPara questions with strong performance when compared to neural network baselines.

[Claim B] Analogical learning can be successfully applied to solve complex reasoning tasks in natural language, while being explainable and data efficient.

In Chapter 4 and 5 we use the analogical learning capabilities in Companion are applied to answer questions about procedural texts and extract general world knowledge from Simple English Wikipedia articles using distant supervision. Furthermore, analogical learning is shown to have desirable properties when it comes to learning from a few training examples and producing inspectable intermediate representations which are used to make predictions. When tested on ProPara, our proposed APTU system is able to achieve 84.3% for macro-avg and 79.5% of the F1 score with just 100 training examples out of the 2,639 total training sentences, outperforming other baselines trained on full data. In the knowledge extraction domain, we show that our Analogical Knowledge Extraction (AKE) system is able to learn high precision facts even from a sparse KB such as NextKB, where most of the relations follow a long tail distribution when it comes to their frequency in facts.

[Claim C] Explainability issues of neural models can be mitigated by training such models to output a step-by-step structured explanation of their predictions.

When compared to other AI methods, artificial neural networks have strong generalization properties but are still regarded as black-box systems. In Chapters 6 and 7, we show how Transformer-based language models can be used to not only answer complex reasoning questions, but also output a human-friendly structured explanation of their output. First, we use the EntailmentBank dataset which contains explanations in the form of entailment trees to train a

generative model to output a step-by-step explanation of their answers. Our method, called Iterative Retrieval and Generation Reasoner (IRGR), performs multiple retrieval and generation steps in order to predict an entailment tree that explains a certain hypothesis (or question-answer pair). We show that IRGR is able to obtain around 300% gains in overall correctness when compared to other baselines. Second, we created a new Structured Reasoning and Explanation Benchmark called STREET that contains multi-step and multi-premise explanation for questions in various domains. Interestingly, we show that for some of the STREET tasks popular language models still lag behind humans when it comes to generating explanations to their answers.

8.2 Future Work

While we propose various methods and techniques to reason over natural language with the Companion and Transformer architectures, we believe there are still many open questions and possible research avenues that can be explored.

8.2.1 Other Integrations Between Companion and Transformer Architectures

We use both Companions and Transformer-based language models to solve various language tasks. Since both architectures have their own properties when it comes to generalization, data-efficiency and explainability, one obvious question is how to further integrate these architectures, so they mitigate each other's weaknesses. For instance, one common issue found in our error analysis was due to mistakes made by the Companion's language parser. We showed in our AKE system that it is possible to train the BERT language model on FrameNet data to predict word senses that will guide the system when making down-stream predictions. One promising next step would be to further integrate the language model to help the guide CNLU parser rank different grammatical and semantic choices to filter out unlikely parse trees and choice sets. Another direct

application of language models would be to simplify the input text or fix grammatical inaccuracies, so the parsing process is simplified.

8.2.2 Generalization of Analogically Learned Query for Procedural Text Understanding.

We show that Step Semantics could have generic applications in terms of representing state changes described in procedural texts. One interesting experiment that could be performed would be to generalize the query cases learned from the training examples in the ProPara dataset to other domains, such as understanding recipes (Kiddon, Ponnuraj, Zettlemoyer, & Choi, 2015), without any additional training data. For instance, the sentence "let the meat absorb the brine in the bowl" implies a state change (or movement event) that can be generalized from the sentence "Roots absorb water from the soil" in one of ProPara's paragraphs.

8.2.3 Improving Structured Reasoning Explanation Generation

Even though the structured explanation of both Entailment Trees and Reasoning Graphs are over natural language units (e.g., retrieved premises and intermediate conclusions), the problem of generating these types of structured explanations can still be framed as search problem similar to the symbolic reasoning systems. For instance, more recent works on Entailment Bank have shown promising results when using a reasoning controller (Hong, Zhang, Yu, & Zhang, 2022) to perform search both forwards (from premises to hypothesis) and backwards (from hypothesis to possible required intermediate conclusions), similar to meet-in-the-middle search approaches. Another possible research avenue is to leverage other networks' architectures such as Graph Neural Networks (Wu, et al., 2020) to better model these graph-like explanations.

9 References

Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., & Martineau, P. (2015). An exact graph edit distance algorithm for solving pattern recognition problems. *4th International Conference on Pattern Recognition Applications and Methods 2015.*

Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., & Martineau, P. (2018). A parallel graph edit distance algorithm. *Expert Systems with Applications, 94*, 41–57.

Allen, J. (1995). *Natural language understanding.* Benjamin-Cummings Publishing Co., Inc.

Allen, J. F., & Hayes, P. J. (1989). Moments and points in an interval-based temporal logic. *Computational Intelligence, 5*, 225–238.

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The berkeley framenet project. *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics.*

Barbella, D., & Forbus, K. (2010). Analogical dialogue acts: Supporting learning by reading analogies. *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, (pp. 96–104).

Blass, J., & Forbus, K. (2017). Analogical chaining with natural language instruction for commonsense reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, (pp. 1247–1250).

Boros, T., Dumitrescu, S. D., & Pipa, S. (2017, September). Fast and Accurate Decision Trees for Natural Language Processing Tasks. *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017* (pp. 103–110). Varna: INCOMA Ltd. doi:10.26615/978-954-452-049-6_016

Bostrom, K., Zhao, X., Chaudhuri, S., & Durrett, G. (2021, November). Flexible Generation of Natural Language Deductions. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 6266–6278). Online and Punta Cana, Dominican Republic: Association for Computational Linguistics. doi:10.18653/v1/2021.emnlp-main.506

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015, September). A large annotated corpus for learning natural language inference. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 632–642). Lisbon: Association for Computational Linguistics. doi:10.18653/v1/D15-1075

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., . . . others. (2020). Language models are few-shot learners. *Advances in neural information processing systems, 33*, 1877–1901.

Buchanan, B. G., & Smith, R. G. (1988). Fundamentals of expert systems. *Annual review of computer science, 3*, 23–58.

Camburu, O.-M., Rocktäschel, T., Lukasiewicz, T., & Blunsom, P. (2018). e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems, 31*.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E., & Mitchell, T. (2010). Toward an architecture for never-ending language learning. *Proceedings of the AAAI conference on artificial intelligence*, *24*, pp. 1306–1313.

Cartuyvels, R., Spinks, G., & Moens, M.-F. (2020, December). Autoregressive Reasoning over Chains of Facts with Transformers. *Proceedings of the 28th International Conference on Computational Linguistics* (pp. 6916–6930). Barcelona, Spain (Online): International Committee on Computational Linguistics. doi:10.18653/v1/2020.coling-main.610

Celikyilmaz, A., Clark, E., & Gao, J. (2020). Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Chang, M. (2016). *Capturing qualitative science knowledge with multimodal instructional analogies.* Ph.D. dissertation, Northwestern University.

Chen, K., & Forbus, K. (2021). Visual Relation Detection using Hybrid Analogical Learning. *Proceedings of the AAAI conference on artificial intelligence*, *35*, pp. 801–808.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Clark, P., Dalvi, B., & Tandon, N. (2018). What happened? leveraging verbnet to predict the effects of actions in procedural text. *arXiv preprint arXiv:1804.05435*.

Clark, P., Tafjord, O., & Richardson, K. (2021). Transformers as Soft Reasoners over Language. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence.* Yokohama.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., . . . others. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168.*

Crouse, M. (2021). *Question-Answering with Structural Analogy.* Ph.D. dissertation, Northwestern University.

Crouse, M., McFate, C., & Forbus, K. (2018). Learning from unannotated qa pairs to analogically disambiguate and answer questions. *Proceedings of the AAAI Conference on Artificial Intelligence, 32.*

Dalvi Mishra, B., Tandon, N., & Clark, P. (2017). Domain-Targeted, High Precision Knowledge Extraction. *Transactions of the Association for Computational Linguistics, 5*, 233–246. doi:10.1162/tacl_a_00058

Dalvi, B., Huang, L., Tandon, N., Yih, W.-t., & Clark, P. (2018, June). Tracking State Changes in Procedural Text: a Challenge Dataset and Models for Process Paragraph Comprehension. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 1595–1604). New: Association for Computational Linguistics. doi:10.18653/v1/N18-1144

Dalvi, B., Jansen, P., Tafjord, O., Xie, Z., Smith, H., Pipatanangkura, L., & Clark, P. (2021, November). Explaining Answers with Entailment Trees. *Proceedings of the 2021*

Conference on Empirical Methods in Natural Language Processing (pp. 7358–7370). Online and Punta Cana, Dominican Republic: Association for Computational Linguistics. doi:10.18653/v1/2021.emnlp-main.585

Das, R., Munkhdalai, T., Yuan, X., Trischler, A., & McCallum, A. (2019). Building Dynamic Knowledge Graphs from Text using Machine Reading Comprehension. *International Conference on Learning Representations.* Retrieved from https://openreview.net/forum?id=S1lhbnRqF7

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Minneapolis: Association for Computational Linguistics. doi:10.18653/v1/N19-1423

DeYoung, J., Jain, S., Rajani, N. F., Lehman, E., Xiong, C., Socher, R., & Wallace, B. C. (2020, July). ERASER: A Benchmark to Evaluate Rationalized NLP Models. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4443–4458). Online: Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.408

Distiawan, B., Weikum, G., Qi, J., & Zhang, R. (2019). Neural relation extraction for knowledge base enrichment. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (pp. 229–240).

Drabble, B. (1993). Excalibur: a program for planning and reasoning with processes. *Artificial Intelligence, 62*, 1–40.

Du, X., Dalvi, B., Tandon, N., Bosselut, A., Yih, W.-t., Clark, P., & Cardie, C. (2019, June). Be Consistent! Improving Procedural Text Comprehension using Label Consistency. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 2347–2356). Minneapolis: Association for Computational Linguistics. doi:10.18653/v1/N19-1244

Faloutsos, C., McCurley, K. S., & Tomkins, A. (2004). Fast Discovery of Connection Subgraphs. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 118–127). New York, NY, USA: Association for Computing Machinery. doi:10.1145/1014052.1014068

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin, 76*, 378.

Forbus, K. D. (1984). Qualitative process theory. *Artificial intelligence, 24*, 85–168.

Forbus, K. D., & De Kleer, J. (1993). *Building problem solvers* (Vol. 1). MIT press.

Forbus, K. D., & Hinrichs, T. R. (2006). Companion cognitive systems: a step toward Human-Level AI. *AI magazine, 27*, 83–83.

Forbus, K. D., Ferguson, R. W., Lovett, A., & Gentner, D. (2017). Extending SME to handle large-scale cognitive modeling. *Cognitive Science, 41*, 1152–1201.

Forbus, K. D., Gentner, D., & Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive science, 19*, 141–205.

Forbus, K. D., Hinrichs, T., De Kleer, J., & Usher, J. (2010). FIRE: Infrastructure for experience-based systems with common sense. *2010 AAAI Fall Symposium Series.*

Forbus, K. D., Riesbeck, C., Birnbaum, L., Livingston, K., Sharma, A. B., & Ureel, L. C. (2007). A Prototype System that Learns by Reading Simplified Texts. *AAAI Spring Symposium: Machine Reading*, (pp. 49–54).

Forbus, K. D., Riesbeck, C., Birnbaum, L., Livingston, K., Sharma, A., & Ureel, L. (2007). Integrating natural language, knowledge representation and reasoning, and analogical processing to learn by reading. *Proceedings of the National Conference on Artificial Intelligence, 22*, p. 1542.

Forbus, K., Chang, M., Ribeiro, D., Hinrichs, T., Crouse, M., & Witbrock, M. (2019). Step Semantics: Representation for State Changes in Natural Language. *Proceedings of the AAAI 2019 Workshop on Complex Question-Answering.*

Forbus, K., Lockwood, K., Sharma, A., & Tomai, E. (2009). Steps towards a 2nd generation learning by reading system. *AAAI Spring Symposium on Learning by Reading, Spring.*

Gao, T., Han, X., Zhu, H., Liu, Z., Li, P., Sun, M., & Zhou, J. (2019, November). FewRel 2.0: Towards More Challenging Few-Shot Relation Classification. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp.

6250–6255). Hong: Association for Computational Linguistics. doi:10.18653/v1/D19-1649

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive science, 7*, 155–170.

Gentner, D., & Forbus, K. D. (2011). Computational models of analogy. *Wiley interdisciplinary reviews: cognitive science, 2*, 266–276.

Gupta, A., & Durrett, G. (2019, June). Tracking Discrete and Continuous Entity State for Process Understanding. *Proceedings of the Third Workshop on Structured Prediction for NLP* (pp. 7–12). Minneapolis: Association for Computational Linguistics. doi:10.18653/v1/W19-1502

Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., & Smith, N. A. (2018, June). Annotation Artifacts in Natural Language Inference Data. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (pp. 107–112). New: Association for Computational Linguistics. doi:10.18653/v1/N18-2017

Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. (2020). Retrieval augmented language model pre-training. *International conference on machine learning*, (pp. 3929–3938).

Han, S., Wang, X., Bendersky, M., & Najork, M. (2020). Learning-to-Rank with BERT in TF-Ranking. *arXiv preprint arXiv:2004.08476*.

Han, X., Gao, T., Yao, Y., Ye, D., Liu, Z., & Sun, M. (2019, November). OpenNRE: An Open and Extensible Toolkit for Neural Relation Extraction. *Proceedings of the 2019*

Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations (pp. 169–174). Hong: Association for Computational Linguistics. doi:10.18653/v1/D19-3029

Han, X., Zhu, H., Yu, P., Wang, Z., Yao, Y., Liu, Z., & Sun, M. (2018, October). FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 4803–4809). Brussels: Association for Computational Linguistics. doi:10.18653/v1/D18-1514

Hanif, A., Zhang, X., & Wood, S. (2021). A survey on explainable artificial intelligence techniques and challenges. *2021 IEEE 25th international enterprise distributed object computing workshop (EDOCW)*, (pp. 81–89).

Henaff, M., Weston, J., Szlam, A., Bordes, A., & LeCun, Y. (2017). Tracking the World State with Recurrent Entity Networks. *International Conference on Learning Representations.* Retrieved from https://openreview.net/forum?id=rJTKKKqeg

Hinrichs, T. R., & Forbus, K. D. (2014). X goes first: Teaching simple games through multimodal interaction. *Advances in Cognitive Systems, 3*, 31–46.

Hogge, J. C. (1987). Compiling Plan Operators from Domains Expressed in Qualitative Process Theory. *AAAI*, (pp. 229–233).

Hong, R., Zhang, H., Yu, X., & Zhang, C. (2022, July). METGEN: A Module-Based Entailment Tree Generation Framework for Answer Explanation. *Findings of the Association for*

Computational Linguistics: NAACL 2022 (pp. 1887–1905). Seattle: Association for Computational Linguistics. doi:10.18653/v1/2022.findings-naacl.145

Inoue, N., Stenetorp, P., & Inui, K. (2020, July). R4C: A Benchmark for Evaluating RC Systems to Get the Right Answer for the Right Reason. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 6740–6750). Online: Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.602

Jain, S., & Wallace, B. C. (2019, June). Attention is not Explanation. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 3543–3556). Minneapolis: Association for Computational Linguistics. doi:10.18653/v1/N19-1357

Jansen, P., Wainwright, E., Marmorstein, S., & Morrison, C. (2018, May). WorldTree: A Corpus of Explanation Graphs for Elementary Science Questions supporting Multi-hop Inference. *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).* Miyazaki: European Language Resources Association (ELRA). Retrieved from https://aclanthology.org/L18-1433

Jhamtani, H., & Clark, P. (2020, November). Learning to Explain: Datasets and Models for Identifying Valid Reasoning Chains in Multihop Question-Answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 137–150). Online: Association for Computational Linguistics. doi:10.18653/v1/2020.emnlp-main.10

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., . . . Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys, 55*, 1–38.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., . . . Yih, W.-t. (2020, November). Dense Passage Retrieval for Open-Domain Question Answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 6769–6781). Online: Association for Computational Linguistics. doi:10.18653/v1/2020.emnlp-main.550

Khashabi, D., Chaturvedi, S., Roth, M., Upadhyay, S., & Roth, D. (2018, June). Looking Beyond the Surface: A Challenge Set for Reading Comprehension over Multiple Sentences. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 252–262). New: Association for Computational Linguistics. doi:10.18653/v1/N18-1023

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications, 82*, 3713–3744.

Kiddon, C., Ponnuraj, G. T., Zettlemoyer, L., & Choi, Y. (2015, September). Mise en Place: Unsupervised Interpretation of Instructional Recipes. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 982–992). Lisbon: Association for Computational Linguistics. doi:10.18653/v1/D15-1114

Kirk, J. R., & Laird, J. E. (2016). Learning general and efficient representations of novel games through interactive instruction. *Advances in Cognitive Systems, 4*, 5.

Ko, M., Lee, J., Kim, H., Kim, G., & Kang, J. (2020, November). Look at the First Sentence: Position Bias in Question Answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1109–1121). Online: Association for Computational Linguistics. doi:10.18653/v1/2020.emnlp-main.84

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Lao, N., Subramanya, A., Pereira, F., & Cohen, W. (2012). Reading the web with learned syntactic-semantic inference rules. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, (pp. 1017–1026).

Lenat, D. B., & Guha, R. V. (1989). *Building large knowledge-based systems; representation and inference in the Cyc project.* Addison-Wesley Longman Publishing Co., Inc.

Lenat, D., & Guha, R. V. (1990). Cyc: A midterm report. *AI magazine, 11*, 32–32.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., . . . others. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems, 33*, 9459–9474.

Lin, W., Tseng, B.-H., & Byrne, B. (2021, November). Knowledge-Aware Graph-Enhanced GPT-2 for Dialogue State Tracking. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 7871–7881). Online and Punta Cana, Dominican Republic: Association for Computational Linguistics. doi:10.18653/v1/2021.emnlp-main.620

Ling, W., Yogatama, D., Dyer, C., & Blunsom, P. (2017, July). Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 158–167). Vancouver: Association for Computational Linguistics. doi:10.18653/v1/P17-1015

Lockwood, K., & Forbus, K. (2009). Multimodal knowledge capture from text and diagrams. *Proceedings of the fifth international conference on Knowledge capture*, (pp. 65–72).

Long, R., Pasupat, P., & Liang, P. (2016, August). Simpler Context-Dependent Logical Forms via Model Projections. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1456–1465). Berlin: Association for Computational Linguistics. doi:10.18653/v1/P16-1138

Lukovnikov, D., Fischer, A., Lehmann, J., & Auer, S. (2017). Neural network-based question answering over knowledge graphs on word and character level. *Proceedings of the 26th international conference on World Wide Web*, (pp. 1211–1220).

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., . . . others. (2023). Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Mavi, V., Jangra, A., & Jatowt, A. (2022). A survey on multi-hop question answering and generation. *arXiv preprint arXiv:2204.09140*.

McCarthy, J. (1960). Programs with common sense (pp. 300-307). *RLE and MIT computation center*.

McFate, C. J., & Forbus, K. D. (2016). An Analysis of Frame Semantics of Continuous Processes. *CogSci.*

McFate, C., & Forbus, K. (2011, June). NULEX: An Open-License Broad Coverage Lexicon. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 363–367). Portland: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P11-2063

McLure, M., Friedman, S., & Forbus, K. (2015). Extending analogical generalization with near-misses. *Proceedings of the aaai conference on artificial intelligence*, *29.*

Mihaylov, T., Clark, P., Khot, T., & Sabharwal, A. (2018, October). Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 2381–2391). Brussels: Association for Computational Linguistics. doi:10.18653/v1/D18-1260

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems, 26*.

Miller, G. A. (1998). *WordNet: An electronic lexical database.* MIT press.

Mintz, M., Bills, S., Snow, R., & Jurafsky, D. (2009, August). Distant supervision for relation extraction without labeled data. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP* (pp. 1003–1011). Suntec: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P09-1113

Musen, M. A., & Van der Lei, J. (1988). Of brittleness and bottlenecks: Challenges in the creation of pattern-recognition and expert-system models. In *Machine intelligence and pattern recognition* (Vol. 7, pp. 335–352). Elsevier.

Newell, A., & Simon, H. (1956). The logic theory machine–A complex information processing system. *IRE Transactions on information theory, 2*, 61–79.

Nguyen, T.-P., Razniewski, S., & Weikum, G. (2021). Advanced semantics for commonsense knowledge extraction. *Proceedings of the Web Conference 2021*, (pp. 2636–2647).

Pan, X., Sun, K., Yu, D., Chen, J., Ji, H., Cardie, C., & Yu, D. (2019, November). Improving Question Answering with External Knowledge. *Proceedings of the 2nd Workshop on Machine Reading for Question Answering* (pp. 27–37). Hong: Association for Computational Linguistics. doi:10.18653/v1/D19-5804

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018, June). Deep Contextualized Word Representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 2227–2237). New: Association for Computational Linguistics. doi:10.18653/v1/N18-1202

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences, 63*, 1872–1897.

Qu, M., Gao, T., Xhonneux, L.-P., & Tang, J. (2020). Few-shot relation extraction via bayesian meta-learning on relation graphs. *International conference on machine learning*, (pp. 7867–7876).

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., . . . Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research, 21*, 1–67. Retrieved from http://jmlr.org/papers/v21/20-074.html

Rajani, N. F., McCann, B., Xiong, C., & Socher, R. (2019, July). Explain Yourself! Leveraging Language Models for Commonsense Reasoning. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 4932–4942). Florence: Association for Computational Linguistics. doi:10.18653/v1/P19-1487

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016, November). SQuAD: 100,000+ Questions for Machine Comprehension of Text. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 2383–2392). Austin: Association for Computational Linguistics. doi:10.18653/v1/D16-1264

Reimers, N., & Gurevych, I. (2019, November). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Hong: Association for Computational Linguistics. doi:10.18653/v1/D19-1410

Ribeiro, D. N., & Forbus, K. (2021). Combining analogy with language models for knowledge extraction. *3rd Conference on Automated Knowledge Base Construction.*

Ribeiro, D. N., Wang, S., Ma, X., Zhu, H., Dong, R., Kong, D., . . . Roth, D. (2023). STREET: A MULTI-TASK STRUCTURED REASONING AND EXPLANATION BENCHMARK.

The Eleventh International Conference on Learning Representations. Retrieved from https://openreview.net/forum?id=1C_kSW1-k0

Ribeiro, D., Hinrichs, T., Crouse, M., Forbus, K., Chang, M., & Witbrock, M. (2019). Predicting state changes in procedural text using analogical question answering. *7th Annual Conference on Advances in Cognitive Systems.*

Ribeiro, D., Wang, S., Ma, X., Dong, R., Wei, X., Zhu, H., . . . Roth, D. (2022, July). Entailment Tree Explanations via Iterative Retrieval-Generation Reasoner. *Findings of the Association for Computational Linguistics: NAACL 2022* (pp. 465–475). Seattle: Association for Computational Linguistics. doi:10.18653/v1/2022.findings-naacl.35

Rickel, J., & Porter, B. (1994). Automated modeling for answering prediction questions: Selecting the time scale and system boundary. *AAAI, 94*, pp. 1191–1198.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature, 323*, 533–536.

Sap, M., Le Bras, R., Allaway, E., Bhagavatula, C., Lourie, N., Rashkin, H., . . . Choi, Y. (2019). Atomic: An atlas of machine commonsense for if-then reasoning. *Proceedings of the AAAI conference on artificial intelligence*, *33*, pp. 3027–3035.

Schmid, H. (2013). Probabilistic part-ofispeech tagging using decision trees. *New methods in language processing*, (p. 154).

Sellam, T., Das, D., & Parikh, A. (2020, July). BLEURT: Learning Robust Metrics for Text Generation. *Proceedings of the 58th Annual Meeting of the Association for*

Computational Linguistics (pp. 7881–7892). Online: Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.704

Seo, M. J., Min, S., Farhadi, A., & Hajishirzi, H. (2017). Query-Reduction Networks for Question Answering. *ICLR (Poster).* Retrieved from https://openreview.net/forum?id=B1MRcPclx

Siler, W., & Buckley, J. J. (2005). *Fuzzy expert systems and fuzzy reasoning.* John Wiley & Sons.

Sinha, K., Sodhani, S., Dong, J., Pineau, J., & Hamilton, W. L. (2019, November). CLUTRR: A Diagnostic Benchmark for Inductive Reasoning from Text. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 4506–4515). Hong: Association for Computational Linguistics. doi:10.18653/v1/D19-1458

Song, K., Tan, X., Qin, T., Lu, J., & Liu, T.-Y. (2020). Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems, 33,* 16857–16867.

Speer, R., Chin, J., & Havasi, C. (2017). Conceptnet 5.5: An open multilingual graph of general knowledge. *Proceedings of the AAAI conference on artificial intelligence, 31.*

Strubell, E., Ganesh, A., & McCallum, A. (2020). Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence, 34,* pp. 13693–13696.

Susskind, R. (1987). *Expert systems in law*. Oxford University Press, Inc.

Tafjord, O., Dalvi, B., & Clark, P. (2021, August). ProofWriter: Generating Implications, Proofs, and Abductive Statements over Natural Language. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021* (pp. 3621–3634). Online: Association for Computational Linguistics. doi:10.18653/v1/2021.findings-acl.317

Tan, S.-S., & Na, J.-C. (2019). Positional attention-based frame identification with bert: A deep learning approach to target disambiguation and semantic frame selection. *arXiv preprint arXiv:1910.14549*.

Tandon, N., Dalvi, B., Grus, J., Yih, W.-t., Bosselut, A., & Clark, P. (2018, October). Reasoning about Actions and State Changes by Injecting Commonsense Knowledge. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 57–66). Brussels: Association for Computational Linguistics. doi:10.18653/v1/D18-1006

Tomai, E., & Forbus, K. D. (2009). EA NLU: Practical Language Understanding for Cognitive Modeling. *FLAIRS Conference.*

Tuan, Y.-L., Beygi, S., Fazel-Zarandi, M., Gao, Q., Cervone, A., & Wang, W. Y. (2022, May). Towards Large-Scale Interpretable Knowledge Graph Reasoning for Dialogue Systems. *Findings of the Association for Computational Linguistics: ACL 2022* (pp. 383–395). Dublin: Association for Computational Linguistics. doi:10.18653/v1/2022.findings-acl.33

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30*.

Veale, T., & Keane, M. T. (1997). The competence of sub-optimal theories of structure mapping on hard analogies. *IJCAI (1)*, (pp. 232–237).

Veloso, M. M., & Carbonell, J. G. (1993). Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Case-Based Learning*, 55–84.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., . . . Zhou, D. (2023). Self-Consistency Improves Chain of Thought Reasoning in Language Models. *The Eleventh International Conference on Learning Representations.* Retrieved from https://openreview.net/forum?id=1PL1NIMMrw

Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., . . . Zhou, D. (2022). Chain of Thought Prompting Elicits Reasoning in Large Language Models. In A. H. Oh, A. Agarwal, D. Belgrave, & K. Cho (Ed.), *Advances in Neural Information Processing Systems.* Retrieved from https://openreview.net/forum?id=_VjQlMeSB_J

Weston, J., Bordes, A., Chopra, S., Rush, A. M., Van Merriënboer, B., Joulin, A., & Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Weston, J., Bordes, A., Yakhnenko, O., & Usunier, N. (2013, October). Connecting Language and Knowledge Bases with Embedding Models for Relation Extraction. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1366–1371). Seattle: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D13-1136

Wilson, J. R., Chen, K., Crouse, M., Nakos, C., Ribeiro, D. N., Rabkina, I., & Forbus, K. D. (2019). Analogical question answering in a multimodal information kiosk. *Proceedings of the Seventh Annual Conference on Advances in Cognitive Systems.*

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Rush, A. (2020, October). Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 38–45). Online: Association for Computational Linguistics. doi:10.18653/v1/2020.emnlp-demos.6

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems, 32*, 4–24.

Xiong, W., Li, X., Iyer, S., Du, J., Lewis, P., Wang, W. Y., . . . Oguz, B. (2021). Answering Complex Open-Domain Questions with Multi-Hop Dense Retrieval. *International Conference on Learning Representations.* Retrieved from https://openreview.net/forum?id=EMHoBG0avc1

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., & Manning, C. D. (2018, October). HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 2369–2380). Brussels: Association for Computational Linguistics. doi:10.18653/v1/D18-1259

Yao, L., Mao, C., & Luo, Y. (2019). KG-BERT: BERT for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Yasunaga, M., Ren, H., Bosselut, A., Liang, P., & Leskovec, J. (2021, June). QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 535–546). Online: Association for Computational Linguistics. doi:10.18653/v1/2021.naacl-main.45

Zellers, R., Bisk, Y., Schwartz, R., & Choi, Y. (2018, October). SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 93–104). Brussels: Association for Computational Linguistics. doi:10.18653/v1/D18-1009

Zeng, D., Liu, K., Lai, S., Zhou, G., & Zhao, J. (2014). Relation classification via convolutional deep neural network. *Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers*, (pp. 2335–2344).

Zhang, Y., Dai, H., Kozareva, Z., Smola, A., & Song, L. (2018). Variational reasoning for question answering with knowledge graph. *Proceedings of the AAAI conference on artificial intelligence*, *32*.

Zhao, C., Xiong, C., Boyd-Graber, J., & Daumé III, H. (2021, June). Multi-Step Reasoning Over Unstructured Text with Beam Dense Retrieval. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies (pp. 4635–4641). Online: Association for Computational Linguistics. doi:10.18653/v1/2021.naacl-main.368

Zhao, W. X., Liu, J., Ren, R., & Wen, J.-R. (2022). Dense text retrieval based on pretrained language models: A survey. *arXiv preprint arXiv:2211.14876*.

Zhong, W., Wang, S., Tang, D., Xu, Z., Guo, D., Chen, Y., . . . Duan, N. (2022, July). Analytical Reasoning of Text. *Findings of the Association for Computational Linguistics: NAACL 2022* (pp. 2306–2319). Seattle: Association for Computational Linguistics. doi:10.18653/v1/2022.findings-naacl.177

Appendix

A Implementation and Experiment Details

A.1 Query Case Construction

Function: STRUCTURE-AWARE-ALIGNMENT
Input: Semantic representation of input text S, target logical expressions T.
Parameters: Preference weight λ
Output: Mapping M^* with improved correspondences between S and T.

$M_{best} \leftarrow$ SME(S, T)
$score_{best} \leftarrow sr(M_{best}) + \sum_{(a,b)\in M_{best}} \lambda * OntCon(a, b)$
while $True$ **do**
  $M_{current} \leftarrow M_{best}$
  foreach M **in** NEIGHBOR-MAPPINGS$(M_{current}, S, T)$ **do**
    $score \leftarrow sr(M) + \sum_{(a,b)\in M} \lambda * OntCon(a, b)$
    if $score > score_{best}$ **then**
      $score_{best} \leftarrow score$
      $M_{best} \leftarrow M$
    end if
  end for
  if $M_{best} = M$ **do**
    return M
  end if
end while

Figure 26: Structure-aware alignment algorithm used during the query cases construction in analogical learning.

As mentioned in Section 3.1.4, Analogical Question-Answering Training (Crouse, 2021) is used to create query cases that connect relevant parts of the semantic representation S of the input text with a desired target logical form T. The semantic representation S is a list of choice-sets, each of which contains a list of choices and their associated logical forms as shown in Figure 2. The query case construction algorithm, called STRUCTURE-AWARE-ALIGNMENT, is shown in Figure 26 and

uses the output of SME as a first-pass solution. Afterwards, it performs a local search to improve the

Function: NEIGHBOR-MAPPINGS
Input: Mapping M, Semantic representation of input text S, target logical expressions T.
Parameters: None.
Output: Union of the set of conflict-free mappings M_{one} and M_{two} of one or two neighbor substitutions, respectively.

$M_{one} \leftarrow \emptyset$
$M_{two} \leftarrow \emptyset$
foreach $edge$ **in** M **do**
  $M' \leftarrow M - edge$
  foreach (s, t) **in** $S \times T$ **do**
    if s, t **not in** $edge' \in M'$ **and not** $conflicts(s, M')$ **then**
      $M_{new} \leftarrow M' \cup \{(s, t)\}$
      $M_{one} \leftarrow M_{one} \cup \{M_{new}\}$
    end if
  end for
end for
foreach $edge_1, edge_2$ **in** $M \times M$ **do**
  $M' \leftarrow M - (edge_2, edge_2)$
  foreach $(s_1, t_1), (s_2, t_2)$ **in** $(S \times T) \times (S \times T)$ **do**
    if $(s_1, t_2, s_2, t_2$ **not in** $edge' \in M'$ **and not** $conflicts(s_1, M')$
      and not $conflicts(s_2, M')$ **and not** $conflicts(s_1, s_2))$ **then**
      $M_{new} \leftarrow M' \cup \{(s_1, t_1), (s_2, t_2)\}$
      $M_{two} \leftarrow M_{two} \cup \{M_{new}\}$
    end if
  end for
end for
return $M_{one} \cup M_{two}$

Figure 27: A sub-routine in the structure-aware alignment algorithm used to find neighboring mapping.

Connectivity following the constraints among choices. In these algorithms, the function $conflicts(a, b)$ returns $TRUE$ if and only if the logical forms (or mappings containing logical forms) a and b are conflicting. The SME output contains a mapping M with a set of correspondences between elements in the base and target logical expressions encoded as a list of pairs. Here, the function $sr(\cdot)$ takes a mapping M and counts the number of statements in the mapping. The statements from the output mapping M from STRUCTURE-AWARE-ALIGNMENT is

used to create a query case. The statements in M that come from S will be included in the query case's antecedent, and the statements in M that come from T will be included in the query case's consequent.

A.2 List of Step Semantics Associated FrameNet Frames

A more complete list of FrameNet frames associated with the five different types of steps is shown below. Note that this is not an exhaustive list since other FrameNet frames could also be associated with these step types depending on the context of the verbs and the abstraction of the entities involved. For instance, we include the frame `FN_Forming_relationships` to represent the creation of a relationship as in the sentence "*The individual befriended the little dogs by giving them treats*", even though relationship is an abstract entity.

- Creation Steps:
    - `FN_Creating, FN_Coming_to_be, FN_Cause_to_start, FN_Intentionally_create, FN_Awareness, FN_Labor_product, FN_Forming_relationships, FN_Being_born, FN_Manufacturing, FN_Cooking_creation, FN_Text_creation, FN_Knot_creation, FN_Duplication, FN_Achieving_first, FN_Forging, FN_Awareness_change_scenario, FN_Activity_start, FN_Amalgamation, FN_Assemble, FN_Cause_to_amalgamate, FN_Come_into_effect`
- Destruction Steps:
    - `FN_Destroying, FN_Cause_to_fragment, FN_Breaking_apart, Breaking_off, FN_Ceasing_to_be, FN_Dying, FN_Death, FN_Sacrificing_for, FN_Attaching,`

FN_ Activity_stop, FN_Becoming_separated,
FN_Breaking_out_captive, FN_Cause_to_end, FN_Erasing,
FN_Fire_end_scenariom FN_Losing_someone, FN_Rotting

- Property Change Steps:
  - FN_Ontogeny, FN_Undergo_change, FN_Becoming,
    FN_Improvement_or_decline, FN_Transition_to_a_state,
    FN_Change_of_phase_scenario,
    FN_Cause_change_of_consistency,
    FN_Cause_change_of_phase, FN_Cause_change_of_strength,
    FN_Adjusting, FN_ Aging, FN_ Becoming_dry,
    FN_Becoming_visible, FN_Catching_fire,
    FN_Cause_to_be_wet, FN_Cause_to_burn,
    FN_Cause_to_wake, FN_Change_of_consistency,
    FN_Change_of_phase, FN_Corroding, FN_Cure, FN_Cutting,
    FN_Damaging, FN_Go_into_shape, FN_Rejuvenation,
    FN_Reshaping
- Quantity Change Steps:
  - FN_Nuclear_process, FN_Amassing,
    FN_Cause_change_of_position_on_a_scale, FN_Aggregate,
    FN_Absorb_heat, FN_Adding_up, FN_Aging,
    FN_Cause_temperature_change,
    FN_Cause_proliferation_in_number, FN_Expansion,

FN_Fall_asleep, FN_Proliferating_in_number,

FN_Rotting, FN_Transfer, FN_Waking_up

- Subprocess / Event Steps:
  - FN_Removing, FN_ Dressing, FN_Undressing, FN_Arranging,

    FN_Arriving, FN_Attending, FN_Breathing, FN_Building,

    FN_Bringing, FN_Burying, FN_Cause_fluidic_motion,

    FN_Cause_motion, FN_Cause_to_move_in_place,

    FN_Departing, FN_Disembarking, FN_ Dodging, FN_ Dunking,

    FN_Downing, FN_Emptying, FN_Escaping, FN_Excreting,

    FN_Exporting, FN_Exercising, FN_Extradition,

    FN_Filling, FN_Fleeing, FN_Fluidic_motion,

    FN_Freeing_from_confinement, FN_Getting_up,

    FN_Ingest_substance, FN_Ingestion, FN_Installing,

    FN_Invading, FN_Kidnapping, FN_Mass_motion, FN_Motion,

    FN_Motion_directional, FN_Passing, FN_Patrolling,

    FN_Placing, FN_Procreative_sex, FN_Quitting_a_place,

    FN_Redirecting, FN_Removing_scenario, FN_Repel,

    FN_Replacing, FN_Ride_vehicle, FN_ Sending,

    FN_Separating, FN_Setting_out, FN_Smuggling,

    FN_Storing, FN_Taking, FN_Touring, FN_Travel,

    FN_Traversing, FN_Visiting,

    FN_Visiting_scenario_arrival,

```
FN_Visiting_scenario_departing,
FN_Visiting_scenario_arrival, FN_Visitor_departure
```

A.3 Common Sense Constraint Algorithm Sub Routines

The first subroutine UPDATE-TABLE simply updates the dynamic programming table D according to the possible state changes and is shown in Figure 28. The table is only updated if the table entry has not been updated or if score is higher than the existing values.

Function: UPDATE-TABLE
Input: Dynamic programming table D, step index n, participant index m, previous existence state $prev$, next existence state $next$, prediction event $event$, update score $score$.
Parameters: None
Output: Updated table D.

$score_{new} \leftarrow 0$
if $cost$ **not** ***null*** **then**
    $score_{new} \leftarrow score_{new} + cost$
end if
if $D[n][m][prev][0]$ **not** ***null*** **then**
    $score_{new} \leftarrow score_{new} + D[n][m][prev][0]$
end if
if $cost$ **not** ***null*** **then** `/* event update */`
    if ($D[n+1][m][next][0]$ is ***null***) **or** ($score_{new} > D[n+1][m][next][0]$) **then**
        $D[n+1][m][next][0] \leftarrow (score_{new}, event, prev)$
    end if
else `/* propagate prev. row update */`
    if ($D[n][m][prev][0]$ **is not** ***null***) **and** (($D[n+1][m][next][0]$ **is** ***null***) **or**
        ($score_{new} > D[n+1][m][next][0]$)) **then**
        $D[n+1][m][next][0] \leftarrow (score_{new}, event, prev)$
    end if
endif
return D

Figure 28: update table subroutine for the commonsense constraint algorithm.

After populating the dynamic programming table D, the Reconstruct-Output-Grid subroutine creates the final state change prediction table that is expected to answer ProPara Questions as shown in Figure 29.

Function: RECONSTRUCT-OUTPUT-GRID
Input: Dynamic programming table D with size ($N + 1$, M, 2).
Parameters: None.
Output: State change output grid G.

Create the output grid G with size ($N + 1$, M)
for col **from** 0 **to** $M - 1$ **do**
    $start_{st} \leftarrow 0$ **if** $D[N][col][0][0] > D[N][col][1][0]$ **else** 1
    $cur_{st} \leftarrow start_{st}$
    $prev_{event} \leftarrow \boldsymbol{null}$
    $prev_{row} \leftarrow N + 1$
    for row **from** N **to** 0 **do** `/* propagate states backwards */`
        $cur_{event} \leftarrow D[row][col][cur_{st}][1]$
        if cur_{event} **is not** $\boldsymbol{null}$ **then**
            APPLY-INERTIA(G, cur_{event}, $prev_{event}$, $row - 1$, $prev_{row} - 1$, col)
            $prev_{event} \leftarrow cur_{event}$
            $prev_{row} \leftarrow row$
        end if
    end for
    APPLY-INERTIA(G, $\boldsymbol{null}$, $prev_{event}$, 0, $prev_{row} - 1$, col)
end for
return G

Figure 29: reconstruct output grid subroutine for the commonsense constraint algorithm.

The APPLY-INERTIA assumes that the state of the participant should not change until an event happens. This function takes the output grid G and propagates the grid entries (i.e., participant states) to completely fill the grid at column col. If two adjacent events disagree with the location, then choose the location of either $prev_{event}$ or cur_{event} according to which one has the highest score.

A.4 Analogical Knowledge Extraction Article Names

Full list of Simple English Wikipedia article names used for the knowledge extraction the experiments are listed below. Some of the article names are simplified with lower case ASCII-friendly characters and spaces are replaced with the dash "-" characters.

aardvark, abstinence, acceleration, accent, accident, accordion, account, acetylene, acid, acid, acid-rain, acne, acorn, acre, action, actor, addiction, addition, address, adelaide, adhesive, adjective, adjustment, administration, administrator, adolescence, adrenal-gland, adult, adventure, adverb, advertisement, afghanistan, africa, age, agent, agreement, air, air-conditioner, air-space, aircraft, airport, alabama, alaska, albania, alberta, albinism, album, alcohol, alder, algae, algeria, alkali, alkali-metal, allergy, alligator, alloy, almond, alphabet, alpine-tundra, alternative, aluminium, ambulance, amino-acid, amoeba, ampere, amphibian, amylase, anaerobic-digestion, analysis, anaphylaxis, anatomy, ancestor, anchor, andorra, anemia, anesthetic, angel, anger, angle, angola, animal, animation, ankle, ant, antarctic-krill, antarctica, anteater, antenna, anthem, antibody, antidote, antigen, anus, anxiety, apartment, ape, aphelion, apparatus, applause, apple, apple-juice, application, apricot, april, apron, aquarium, aquifer, arc, arcade, arch, archer, archery, architecture, area, argentina, argon, argument, arithmetic, arizona, arkansas, arm, armadillo, armenia, armour, army, array, arrow, arsenic, art, artery, arthropod, artist, asexual-reproduction, ash-tree, asia, asian, aspirin, assam, asset, asteroid, asteroid-belt, asthenosphere, astronaut, astronomer, astronomy, athlete, atmosphere, atmospheric-pressure, atom, atomic-mass, atomic-theory, attack, audience, august, aunt, australia, austria, author, authority, autism, automaton, autotroph, autumn, avalanche, avocado, award, axe, axle, azerbaijan, baby, back, bacon, badminton, bag, bahrain, baker, ball, ballet, balloon, bamboo, banana, band, bandage, bandicoot, bangladesh, bank, bar, barber, bark, barley, barnacle, barometer, baseball, basement, basin, basket, basketball, bast, bat, bath, bathroom, battery,

battle, beach, beak, beaker, bean, bear, beard, beauty, beaver, bed, bedroom, bee, beef, beer, beetle, behavior, beijing, being, belarus, belgium, belief, belize, bell, belt, bench, benchmark, benin, benzene, berkshire, berry, beryllium, bhutan, bias, bicycle, biennial-plant, bile, binary-fission, binoculars, biochemistry, biodiversity, biogeography, biological-weapon, biologist, biology, biomass, biome, biosphere, biotechnology, bird, birth, birthday, biscuit, bison, bissau, bit, bivalve, black-death, black-hole, black-pepper, black-powder, blackberry, blackbird, blackboard, blacksmith, blade, bleach, blender, blindness, blonde, blood, blood-pressure, blood-transfusion, blood-vessel, blossom, blouse, blue-cheese, blue-whale, blueberry, boa, boat, body, bomb, bone, bonfire, book, boomerang, boot, boron, boss, botanist, botany, botswana, bottle, bowl, box, boy, boyfriend, brain, brake, branch, brand, brass, brass-instrument, brazil, bread, breakfast, breast, breast-cancer, breath, breed, bribery, brick, bridge, british-museum, bromine, bronze, broom, brown-bear, brown-dwarf, brunei, brush, bryophyte, bucket, budgerigar, budget, building, bulb, bulgaria, bullet, bun, buoyancy, burial, burn, burundi, bus, business, butter, buttercup, butterfly, button, c, c, cabbage, cabin, cabinet, cable-car, cactus, cadmium, cafe, caffeine, cage, cake, calcium, calculator, calendar, california, calorimeter, cambodia, camel, camera, camera-lens, cameroon, campaign, canada, canal, cancer, candle, candy, cane, canoe, canon, cantaloupe, canteen, canvas, canyon, capillary, capital, capital-accumulation, capital-city, captain, car, caracal, carbon, carbon-dioxide, carbon-monoxide, carbon-steel, card, cardboard, career, caretaker, cargo, caribbean, carnival, carnivore, carp, carpenter, carpet, carrot, cartilage, cartoon, cash, castle, cat, cataract, caterpillar, cathedral, cattle, cave, cavity, cayenne-pepper, cecum, ceiling, cell-division, cell-membrane, cell-wall, cellular-respiration, cellulose, census, centimetre, centipede, centre, centripetal-force, century, cerebellum, cerebrum, ceremony, certificate, cervix, chad, chair, chalk, chameleon, chancellor, chaos, charcoal, chart,

cheese, cheetah, chemical-bond, chemical-compound, chemical-element, chemical-equation, chemical-reaction, chemical-substance, chemist, chemistry, chemotherapy, cherry, chess, chest, chicken, child, chile, chimpanzee, chin, china, chinchilla, chipmunk, chisel, chitin, chloride, chlorine, chlorophyll, chloroplast, chocolate, chocolate-cake, choice, choir, cholera, cholesterol, chongqing, chromium, chromosome, church, cider, cigar, cigarette, cinnamon, circle, circumference, citizen, citric-acid, citrus, city, civilization, clam, clamp, clarinet, claw, clay, cleanliness, climate, clipboard, clock, cloth, clothing, cloud, clover, cluster, coal, coast, coat, cobalt, cocaine, cocoa, cocoon, cod, code, coffee, coffin, coin, cola, cold-front, college, collision, colombia, colon, colonization, colony, colorado, columbia, column, coma, comb, combat, combustion, comet, commerce, commodity, communication, community, comparison, compass, competition, complete-metamorphosis, composition, compost, computation, computer, concentration, concept, concert, concrete, condensation, condom, conduction, conductor, cone, conference, conifer, connecticut, connection, constellation, constipation, construction, construction-worker, container, continent, continuity, continuum, contract, contraction, convection, cookbook, cookie, cooperation, copper, copy, coral, coral-reef, cornea, cornwall, coronation, corporation, correlation, corrosion, cottage, cotton, council, country, county, county-seat, county-town, courage, court, cousin, cowboy, coyote, crab, cracker, craft, crater, crayon, cream, creativity, creed, crescent, crew, cricket, crime, croatia, crocodile, crop, crop-rotation, crust, crystal, crystallization, cuba, cube, culprit, culture, cumbria, cup, cupboard, currant, curry, cursor, curve, cushion, custard, custom, customer-service, cuticle, cutlery, cuttlefish, cyanobacteria, cycle, cyclone, cylinder, cynicism, cyprus, cystic-fibrosis, cytokine, cytoplasm, cytosol, dairy-product, dam, dance, dandelion, darkness, data, database, daughter, day, death, death-penalty, debate, debris, debt, decade, decapod, december, deer, defense,

definition, deflation, deforestation, degree, dehydration, deinonychus, delaware, demonstration, denial, denmark, density, deoxyribose, desert, design, dessert, detective, device, devil, devon, diagnosis, diagram, diameter, diamond, diaper, diaphragm, diarrhea, diatom, dice, dictator, dictionary, difference, digestion, dill, dimension, dimetrodon, dingo, dinner, dinosaur, direction, director, disability, disaster, disco, disease, dish, dishwasher, disk, distance, distribution, district, ditch, division, divorce, djibouti, doctor, doctrine, documentation, dodo, dog, dogma, dollar, dolphin, dome, domestic-goat, domestic-pig, domestic-sheep, domestication, dominica, donkey, door, dough, doughnut, dove, dozen, dragonfly, drain, drainage, drama, drawer, drawing, dream, dress, drink, drizzle, drought, drug, drum, drummer, duck, duke, dune, duodenum, dust, dwarf, dye, eagle, ear, earth, earth-science, earthquake, echidna, echinoderm, ecology, economics, economist, economy, ecosystem, ecuador, edge, education, eel, efficiency, egypt, ejaculation, elastic, elastic-energy, elbow, electric-charge, electric-current, electric-motor, electrical-circuit, electrical-energy, electrical-resistance, electrician, electricity, electromagnet, electromagnetic-radiation, electromagnetic-wave, electron, electron-microscope, electronics, element, elementary-particle, elephant, elevation, elevator, ellipse, elm, embryo, embryology, emerald, emotion, emperor, emperor-penguin, emphysema, empire, employee, emu, encyclopedia, endometrium, endoskeleton, enemy, energy, energy-conservation, engine, engineer, england, entertainment, entomology, envelope, environment, enzyme, eon, epididymis, epilepsy, episode, equation, equator, equinox, era, erection, eritrea, erosion, error, erythropoietin, essay, essex, estonia, estuary, ethane, ethanol, ethiopia, ethology, eucalyptus, eukaryote, eurasia, europe, european-commission, evaporation, event, event-horizon, evergreen, evidence, evolution, evolutionary-biology, example, existence, exoskeleton, experience, experiment, exploitation, explorer, explosion, extinction, eye, eyebrow, eyeglasses, eyelash, face, fact,

factory, fairy, falcon, family, farm, farmer, fashion, fast-food, fate, father, fault, fear, feather, february, feedback, fellow, felt, fence, fermentation, fern, ferret, fertilization, fertilizer, festival, fetus, fever, fibrous-protein, fiction, fife, fig, fiji, fin, fine, finger, fingernail, finland, fir, fire, fire-salamander, fireplace, firework, fish, flag, flagellum, flame, flashlight, flask, flatworm, flavor, flea, flight, flipper, floor, florida, flour, flower, fluid, fluorine, flute, flux, fly, fog, food, foot, football, footballer, footwear, force, force-field, forehead, forest, forestry, fork, formula, fossil, fossil-fuel, fowl, fox, fraction, fracture, framework, france, freedom, freeway, frequency, fresh-water, friday, friend, friendship, frog, frost, fructose, fruit, fruit-tree, fuel, fungus, fur, furniture, future, g, gabon, galaxy, galena, gallon, game, gamete, garbage, garden, gargoyle, gas, gasoline, gate, gazelle, gear, gecko, gender, gene, general-relativity, generation, genetics, genie, genus, geography, geology, gerbil, germ, german-empire, germany, ghana, ghost, giant-anteater, giant-panda, giant-salamander, giant-squid, gill, gin, ginkgo, giraffe, girl, glacier, gland, glass, glaze, glider, globe, glossary, gloucestershire, glove, glucose, glue, glycogen, glycolysis, gnome, gnu, god, gold, goldfish, golf, gonad, gondola, gonorrhea, goodbye, goodness, goose, gooseberry, gopher, gorilla, gospel, government, governor, gradient, grain, gram, grandparent, granite, grape, grapefruit, graph, graphite, grass, grasshopper, grave, gravitational-energy, gravity, greece, green-tea, greenhouse, greenhouse-effect, greenhouse-gas, grenada, grey-wolf, griffin, ground, group, guatemala, guava, guinea, guinea-pig, guitar, gull, gun, guyana, gymnastics, gymnosperm, h, habitat, haddock, hail, hair, hairdresser, haiti, hallucination, halo, halogen, ham, hamburger, hamlet, hammer, hampshire, hamster, hand, handball, hanuman, happiness, harbor, hardware, hare, harmony, harp, harvest, hat, hawaii, hawthorn, head, headache, health, healthy-diet, heart, heart-disease, heartbeat, heat, heat-capacity, heat-conduction, heat-engine, heaven, hedgehog, height, helicopter, helium, hello, helmet, hematite,

hemisphere, hemoglobin, hepatitis, herb, herbivore, herd, hereditary-disease, hero, herring, hertfordshire, heterotroph, hibernation, hierarchy, high-school, high-voltage, highland, highway, hill, hinge, hippopotamus, histogram, historian, history, hobby, hockey-stick, hoe, holiday, holland, holly, homeostasis, homework, honduras, honey, honeycomb, horizon, hormone, hornet, horror, horror-film, horse, horseshoe, hospital, hostage, hot-sauce, hot-spot, hotel, hour, house, houseplant, huckleberry, hue, human, human-evolution, human-migration, humidity, hummingbird, humpback-whale, hungary, hydrocarbon, hydrogen, hydrogen-peroxide, hydroxide, hyena, hygiene, hymn, hypertension, hypothesis, ice, ice-age, ice-cream, iceberg, iceland, icon, idaho, idea, idiot, igloo, ileum, illinois, illness, illusion, imagination, immunity, immunization, inch, incisor, income, incomplete-metamorphosis, incubation-period, india, indian, indiana, indigo, individual, indonesia, industrial-revolution, industry, inertia, infection, infectious-disease, infertility, inflammation, inflation, influenza, information, infrastructure, inhalation, inheritance, injury, ink, inorganic-compound, insect, insecticide, insectivore, insolation, institution, instrument, insulation, insulin, intelligence, interest, internal-energy, international-law, international-organization, interval, intestine, invasion, invention, inventor, iodine, ion, ionic-bond, ionic-compound, iowa, iran, iraq, ireland, iron, iron-oxide, irrigation, island, isotope, israel, issue, isthmus, italy, ivory, j, jackal, jade, jaguar, jail, jam, jamaica, janitor, january, jasmine, javan-tiger, jay, jazz, jejunum, jelly, jellyfish, jerboa, jerk, jersey, jet-engine, jewel, jewelry, job, jockey, john, joint, joke, jordan, joule, journal, journalist, judge, juice, july, june, jungle, kangaroo, kangaroo-rat, kansas, kaolinite, karate, kayak, kazakhstan, kebab, kelp, kelvin, kent, kentucky, kenya, keratin, kerosene, ketchup, kettle, key, kidney, kiln, kilogram, kilometre, kilt, kinetic-energy, kinetic-theory, king, kingdom, kiribati, kitchen, kiwi, knee, knife, knight, knob, knot, knowledge, knuckle, koala, krill, kuwait,

kyrgyzstan, l, labium, laboratory, lactose, ladder, lake, lamp, lancashire, land, landfill, landslide, language, lantern, laptop, large-intestine, larva, laser, latex, latitude, latvia, lava, law, lawn, lead, leader, leaf, leather, lebanon, lecture, leech, leek, leg, legend, leicestershire, leisure, lemon, lemonade, lemur, length, lens, leopard, lesotho, lesson, letter, lettuce, lever, liberia, librarian, library, libya, license, liechtenstein, life, ligament, light, light-pollution, lightbulb, lighthouse, lightning, lily, limestone, lincolnshire, line, linen, linguistics, link, lion, lip, lipstick, liquid, list, literacy, literature, lithium, lithosphere, lithuania, litre, liver, livestock, lizard, llama, lobster, local-government, lock, locomotion, locust, logic, logo, lord, loudspeaker, louisiana, luggage, lullaby, lumber, lung, lung-cancer, luxembourg, lymphocyte, lynx, m, ma, macau, macedonia, machine, madagascar, madness, magazine, magic, magma, magnesium, magnet, magnetic-field, magnetism, maine, majority, malaria, malawi, malay, malayan-tiger, malaysia, mali, mall, malta, mammal, mammary-gland, man, manager, mango, manitoba, manure, map, marathon, marble, margarine, marker, market, marmalade, marmot, marriage, marsh, martinique, maryland, mascot, mass, massachusetts, mast, material, mathematician, mathematics, matter, mauritania, mauritius, max, may, mayor, meal, meat, medical-emergency, medication, medicine, medicine-man, medium, medusa, meerkat, meiosis, melody, melon, member, membrane, membrane-protein, meningitis, menstruation, mental-illness, merchant, mercury, meristem, metabolism, metal, metamorphosis, meteorology, methane, method, metre, mexico, michigan, microbiology, microorganism, microscope, microwave, microwave-oven, middle-school, migrant-worker, migration, mile, military-engineer, milk, milk-powder, millipede, mind, mineral, minister, minnesota, minute, miracle, mirror, missile, mississippi, missouri, mitosis, mitten, mixture, moat, modem, moldova, mole, molecule, mollusc, momentum, monaco, monarch, monastery, monday, money, mongolia, monitor, monk, monkey,

monosaccharide, monotreme, monster, montana, montenegro, month, monument, moose, mop, morality, morning, morocco, mosque, mosquito, moss, moth, mother, motion-sensor, motorway, mountain, mountain-climber, mouse, mouth, movement, movie, mozambique, mud, mud, mule, multicellular-organism, multiplication, municipality, murder, muscle, muse, museum, mushroom, music, musical-instrument, musician, musk-deer, mussel, mustache, mustard-plant, mutation, mycelium, nail, namibia, nancy, nasal-cavity, nation, national-park, natural-disaster, natural-environment, natural-gas, natural-resource, nature, nausea, navigation, nebraska, nectar, needle, neighbour, neon, nepal, nerve, nest, net, netball, netherlands, network, neuron, neurotransmitter, neutron, nevada, news, newspaper, newt, newton, nickel, nicotine, niger, nigeria, nightmare, nitrogen, noble-gas, noise, nonmetal, noodle, noon, norfolk, norm, northern-hemisphere, northern-territory, northumberland, norway, nose, nostril, notebook, noun, novel, november, nuclear-energy, nuclear-fission, nuclear-fusion, nuclear-power, nuclear-reaction, nuclear-reactor, nuclear-weapon, nucleotide, number, nun, nunavut, nurse, nutrient, nylon, o, oak, oar, oat, oaxaca, obesity, observation, observatory, october, octopus, odor, office, officer, official, offspring, ohio, oil, okapi, oklahoma, olive-oil, olive-tree, oman, omelette, omnivore, onion, ontario, opinion, orange, orangutan, orbit, orca, orchestra, orchestra-pit, ore, oregon, organelle, organic-chemistry, organic-compound, organism, organization, oriole, orphan, osprey, ostrich, oval, ovary, oven, ovum, owl, ownership, oxbow-lake, oxidation, oxide, oxygen, oyster, p, pain, painting, pakistan, palace, palestine, palm-tree, panama, pancake, pancreas, pangolin, panther, papaya, paper, paperback, parachute, paragraph, paraguay, paralysis, parameter, parasitism, parent, park, parliament, parrot, parsley, particle, pascal, passport, past, pasta, pasture, patch, patent, pathogen, pattern, pea, peace, peach, peafowl, peanut, pear, pearl, pelican, pelvis, pen, pencil, penguin, penicillin, penis, pennsylvania, penny, pepper, pepsin,

percent, perception, perch, percussion-instrument, perfume, perihelion, periodic-table, peru, pesticide, pet, petroleum, pharmacist, pharynx, phenomenon, phenotype, philippines, philosopher, phloem, phosphate, phosphorus, photographer, photography, photon, photosynthesis, phrase, phylum, physical-law, physical-property, physiology, pi, pianist, piano, picture, pie, piece, pigment, pillar, pillow, pilot, pin, pine, pineapple, pint, pipe-smoke, piston, pituitary-gland, pizza, placenta, plain, planet, plankton, plant, plant-stem, plaque, plastic, plate, platelet, platinum, platypus, playground, plow, plum, plunger, pneumonia, pocket, pocket-gopher, pod, poet, poetry, poison, poland, polar-bear, police, police-officer, politician, pollination, pollutant, pollution, polyp, polysaccharide, pomegranate, pond, poodle, population, population-density, porcelain, pore, pork, porpoise, portugal, position, possession, poster, pot, potassium, potato, potential, potential-energy, poverty, powder, power, power-station, power-structure, prairie, prayer, precipitation, predator, prediction, pregnancy, presentation, president, pressure, prevention, price, pride, priest, primate, prime-minister, prince, princess, principal, prison, prisoner, prize, probability, problem, producer, profession, professor, profit, project-management, prokaryote, pronghorn, propeller, property, proposal, protein, protest, protist, proton, protractor, province, psychoanalysis, puberty, public-school, pudding, puddle, pulley, pulmonary-artery, pulmonary-hypertension, pump, pumpkin, pupa, pylon, pyramid, python, q, qatar, quake, quality, quantity, quart, quarter, quartz, quebec, queen, question, queue, quiz, r, rabbit, rabies, racket, radar, radian, radiation, radio, radio-telescope, radio-wave, radioactive-decay, radius, radon, raid, raid, railway-station, rain, rainbow, rainbow-trout, rainforest, rainstorm, raisin, ranch, raspberry, rat, ratio, raven, ravioli, raw-material, ray, razor, reality, rear, rear-window, receipt, recorder, recreation, rectangle, rectum, red-panda, reduction, reed, referee, reference, reflection, refraction, refrigerator, region, reindeer, relative-humidity, religion,

renewable-energy, renewable-resource, reproduction, reptile, republic, resin, resistor, respiration, restaurant, retina, reunion, revolution, reward, rhinoceros, rhubarb, rhythm, rib-cage, ribose, ribosome, rice, rifle, ring, riot, river, road, robot, rock, rocket, rodent, rodeo, roe, role, roller-coaster, roman, roman-emperor, romania, roof, room-temperature, root, rope, rose, rose-hip, rotation, rowan, royal-family, rubber, rug, rule, ruler, rumble, russia, rust, rutland, rwanda, s, sail, sailor, saint, sake, salad, salamander, salinity, saliva, salivary-gland, salmon, salmonberry, salt, salt-water, salvador, samoa, sand, sandstone, sandwich, sarcasm, sardine, saskatchewan, satellite, sauce, sausage, savanna, scar, scarcity, schizophrenia, science, scientist, scissors, scone, score, scorpion, scotland, screw, scrotum, sea, sea-anemone, sea-cucumber, sea-level, sea-urchin, seafood, seal, search-engine, season, seaweed, sebaceous-gland, second, second-person, secretary, secretion, seed, segment, seizure, semen, seminal-vesicle, senate, senegal, sensation, sense, sentence, september, septum, sequence, serbia, series, servant, server, service, set, sewer, sex, sex-organ, sexual-reproduction, shade, shadow, shampoo, shape, shark, shaving, shelf, shellfish, shelter, shield, shield-volcano, ship, shire, shirt, shock, shoe, shooter, shop, shore, shoulder, shovel, shower, shrew, shrimp, shrub, siberian-tiger, sibling, sickle, side, sight, sign, silicon, silk, similarity, simulation, singapore, singer, sir, siren, site, size, skeleton, skin, skink, skirt, skull, skunk, sky, skyscraper, slang, sleeve, slide, slope, sloth-bear, slovakia, slovenia, small-intestine, smile, smoke, smoothie, snack, snail, snake, snow, snow-leopard, snowball, snowflake, snowman, soap, soap-bubble, social-contract, social-worker, society, sock, sodium, sodium-hydroxide, soft-drink, softball, software, soil, soil-moisture, solar-eclipse, solar-energy, solar-radiation, soldier, solstice, solubility, solution, solvent, somalia, somersault, somerset, son, song, soul, sound, soup, southern-hemisphere, southwest, soy-milk, soy-sauce, soybean, space-shuttle, spacecraft, spade, spaghetti, spain, spear, special-relativity, spectrum,

speech, speed, sperm, sphere, spice, spider, spinach, spinal-cord, spiny-lobster, spirit, spleen, sponge, spoon, spore, sport, spree, spring, spruce, squid, squirrel, stainless-steel, stalactite, stalk, standard, standard-deviation, stapler, star, starch, starvation, state, statement, statue, steak, steam, steam-engine, steel, steppe, stethoscope, stick, stimulus, sting, stingray, stirrup, stoat, stock, stomach, stone, stone-fruit, stopwatch, storage, stork, storm, strategy, strawberry, stream, street, strength, string, structure, student, studio, stuff, subduction, sublimation, submarine, substance, suburb, succulent-plant, sudan, sugar, sugar-beet, suggestion, suicide, sulfur, sultan, sumatran-tiger, summary, summer, sun, sunburn, sunflower, sunlight, sunscreen, sunset, superman, supermarket, supernova, surface, surface-area, surgery, suriname, survivor, sussex, sustainable-development, swamp, swan, swaziland, sweden, switzerland, sword, syllable, symbiosis, symbol, symmetry, symptom, syphilis, syria, syrup, system, table, table-salt, tadpole, tail, tajikistan, talent, tank, tanzania, tapeworm, tapir, tarsier, taste, tattoo, tax, taxi, taxon, taxonomy, tea, teacher, team, teapot, teaspoon, technique, technology, teenager, telephone, telescope, television, temperature, template, temple, tendon, tennessee, tennis, tent, tentacle, test, test-tube, testosterone, tetanus, texas, textbook, texture, thailand, theatre, theorem, theory, thermal-energy, thermometer, thorax, thought, thousand, threat, throat, throne, thumb, thunder, thunderstorm, thursday, tibet, tidal-force, tide, tiger, tile, tillage, timbre, time, time-zone, tin, tire, titanium, toaster, tobacco, toddler, toe, togo, toilet, tolerance, tomato, tomorrow, tonga, tongue, tonne, tooth, toothache, toothbrush, tornado, torque, toucan, touch, tourism, towel, tower, town, toy, trademark, trader, trail, tram, transaction, transition-metal, translation, transparency, transpiration, transport, tray, treaty, tree, trench, trial, triangle, tribe, trinity, trolley, trombone, tropical-cyclone, tropical-rainforest, trout, truck, trumpet, tsunami, tuatara, tuba, tuberculosis, tuesday, tuna, tundra, tungsten, tunisia, tunnel, turbine, turkey, turkmenistan,

turtle, tuvalu, twin, typewriter, tyrannosaurus, uganda, ukraine, ulcer, umbilical-cord, umbrella, uncle, underwear, unemployment, ungulate, unicellular-organism, unicorn, uniform, unit, universe, university, uranium, urban-area, urethra, urination, urine, uruguay, utah, uterus, uzbekistan, vacation, vacuum, vacuum-tube, vagina, vaginal-secretion, valley, value, vampire, vampire-bat, vampire-squid, van, vanilla, vanuatu, vapor, vascular-plant, vascular-tissue, vase, vector, vegetable, vegetable-oil, vegetation, vehicle, vein, velocity, venezuela, verb, vermont, vertebrate, vessel, vibration, vice-president, victim, video, video-game, vienna, vietnam, vietnam-war, village, villain, vinegar, violence, violet, violin, virginia, virus, viscacha, visible-light, vitamin, vocabulary, voice, volcano, vole, volleyball, volt, voltage, volume, vulture, vulva, waiter, wall, wallet, war, war-crime, wart, washington, wasp, wasp, waste, waste-heat, water, water-vapor, water-wheel, waterfall, watermelon, waterway, watt, wavelength, wax, weapon, weasel, weather, weather-station, wedge, wednesday, weed, week, weekend, weight, west, wetland, whale, whale-shark, wheat, wheel, wheelbarrow, wheelchair, whip, whisker, whistle, white-dwarf, whorl, width, wife, wild-turkey, wildebeest, wind, windmill, window, wine, wing, winter, winter-storm, wire, wisconsin, witch, witness, wolverine, woman, wombat, wonder-woman, wood, woodland, woodpecker, woodwind-instrument, wool, word, word-order, worker, world, worm, worship, wrist, wristwatch, writer, wyoming, xylem, xylem-vessel, year, yeast, yemen, yes, yoga, yoghurt, yorkshire, yukon, z, zambia, zebra, zebra-mussel, zero, zimbabwe, zinc, zoo, zygote

A.5 Knowledge Extraction Hyperparameters and Training Settings

Here are the training details and hyperparameters mentioned in submodule for AKE as well as baseline models.

Word Sense Disambiguation: the disambiguation module uses the Hugging Face's distribution of BERT[13] [`base-uncased`] as the base model to predict the word senses associated with FrameNet frames. It is fine-tuned by optimizing the cross-entropy loss using Adam optimizer, with a learning rate of 10^{-4} and trained for 10 epochs with a batch size of 32 examples.

Construction of Query Cases and Fact Prediction: the hyper parameters by the SEMANTIC-SELECTION algorithm in Figure 12 were set to $\lambda_1 = 1, \lambda_2 = 0.2, \lambda_3 = 5, \lambda_4 = 5, \varepsilon_1 = 0.3$. The hyper parameters used in Equation (14) were set to $\lambda_5 = 1, \lambda_6 = 1, \lambda_7 = 1, \lambda_8 = 1, \varepsilon_2 = 2.5$. These values were chosen according to the importance and magnitude of each score while considering the results of trial runs.

Fact Scoring: the fact scoring module uses the Hugging Face's distribution of BERT [`base-uncased`] to score predicted facts as a post processing step. The model is fine tuned to optimize the cross-entropy loss using the AdamW optimizer, with a learning rate of $5 * 10^{-5}$ and is trained for 3 epochs with a batch size of 16 examples.

Text-to-text T5 Training Details: the baseline uses the Hugging Face's distribution of T5[14] [`small`] on a set of 3,962 sentence-fact pairs. The model was trained auto-regressively using the AdamW optimizer, with a learning rate of $3 * 10^{-4}$ for 2 epochs.

Relation Extraction Training Details: Both Relation Extraction baselines (with CNN and BERT encoders) use the OpenNRE[15] implementation. They are trained on a dataset of 28,800 training examples and 3,000 validation sentence-fact pairs, where 50% of which were negative examples. Sentences that reference two entities that are not connected by a fact in the KB were used as negative examples. All facts with more than two examples were discarded. The model with CNN

[13] Available at https://huggingface.co/docs/transformers/model_doc/bert
[14] Available at https://huggingface.co/docs/transformers/model_doc/t5
[15] Available at https://github.com/thunlp/OpenNRE

encoder was trained with a learning rate of $1 * 10^{-1}$, for 500 epochs with a batch of 160 examples. The BERT based model was trained with a learning rate of $2 * 10^{-5}$, for 10 epochs with a batch size of 64 examples. The remaining hyperparameter were kept as the OpenNRE defaults.

A.6 Iterative Reasoner Hyperparameters and Training Settings

The $IRGR - generator$ uses the Hugging Face's distribution of the encoder-decoder T5[16] [`small`] model. The model weights with best *Overall: All-Correct* metric on the validation set were selected. The module was trained auto-regressively using the Adam optimizer and with the following hyper parameters: learning rate: $3 * 10^{-5}$, train epochs: 15, training batch size: 4, validation batch size: 4, maximum number of input tokens: 512, maximum number of output tokens: 256, warm-up steps: 0, weight decay: 0.

The $IRGR - retriever$ module uses the version `all-mpnet-base-v2` from the Sentence-Transformers library. The model is fine-tuned to optimize the cosine similarity loss using the AdamW optimizer with the following hyper parameters: learning rate: $5 * 10^{-5}$, epochs: 10, training batch size: 32, validation batch size: 32, loss function: cosine similarity loss, warm-up steps: 0, weight decay: 0.

A.7 Reasoning Graph Generation Hyperparameters and Training Settings

Full supervision: The T5 [`large`] model uses the Hugging Face's distribution[17] model (770 million parameters) is trained on each task separately. The training is done using a machine with four NVIDIA Tesla V100-SXM2 with 16GB of VRAM each. We select the model weights

[16] Available at https://huggingface.co/docs/transformers/model_doc/t5
[17] Available at https://huggingface.co/docs/transformers/model_doc/t5

(checkpoint) with highest answer accuracy on the development set. The model is fine-tuned auto regressively and uses the AdamW as optimizer. The training runs for up to 30 epochs with training batch size of 2 data examples. The learning rate starts at zero and is gradually increased to its maximum value of $3 * 10^{-5}$. After 1000 steps, the learning rate is decreased following a cosine function scheduler. The weight decay 10^{-3}. During the generation step we use beam search with a beam size of 5.

Few-shot prompting: we run GPT-3 [`davinci`] by accessing OpenAI's API[18]. The API provides access to a few model variants and for our experiments we use the largest advertised model, namely `text-davinci-002` (175B parameters). When generating the reasoning graphs, we select up to 5 examples (depending on the tasks and models, fewer prompt examples might be provided due to the encoder token size limit) as few-shot prompts for the model. We use greedy decoding and use the remaining default hyperparameters, not setting any maximum or minimum output size.

[18] Available at: https://platform.openai.com/docs/models/gpt-3

B Structured Reasoning and Explanation Benchmark

B.1 Data Examples

Below are data point textual encoding examples for each of the five tasks in STREET taken from the development set. The SCONE task contains three different sub-tasks (namely Alchemy, Scene, and Tangrams), and we show examples for each of them separately since their format change among different subtasks.

ARC:

```
$question$ = (1) the sun rising / setting occurs once per day (2) the sun setting is a kind of event (3) the sun rising is a kind of event (4) Which event occurs on a daily cycle? (5) A) The Sun rises and sets. (6) B) Earth tilts on its axis. (7) C) Earth revolves around the Sun. (8) D) The Moon revolves around Earth.

$proof$ = (1) & (2) & (3) -> (9): the sun rising and setting is the event that occurs once per day; (9) -> (10): The answer is A);
```

SCONE (Alchemy):

```
$question$ = (1) first beaker has 0 chemicals (2) second beaker has 1 green chemical (3) third beaker has 1 purple chemical (4) fourth beaker has 1 orange chemical (5) fifth beaker has 1 green chemical (6) sixth beaker has 1 red chemical (7) seventh beaker
```

has 1 yellow chemical (8) throw out the orange chemical (9) then, add the leftmost beaker of green chemical to the yellow chemical (10) mix it (11) then, add the remaining green chemical to it (12) mix that too

$proof$ = (4) & (8) -> (13): fourth beaker has 0 chemicals; (2) & (7) & (9) -> (14): seventh beaker has 1 yellow and 1 green chemical; (2) & (9) -> (15): second beaker has 0 chemicals; (14) & (10) -> (16): seventh beaker has 2 brown chemicals; (16) & (11) & (5) -> (17): seventh beaker has 2 brown and 1 green chemicals; (11) & (5) -> (18): fifth beaker has 0 chemicals; (17) & (12) -> (19): seventh beaker has 3 brown chemicals;

SCONE (Scene):

$question$ = (1) position 1 has no person (2) position 2 has no person (3) position 3 has no person (4) position 4 has no person (5) position 5 has person in red shirt and yellow hat (6) position 6 has no person (7) position 7 has no person (8) position 8 has no person (9) position 9 has no person (10) position 10 has no person (11) a man in a yellow shirt appears on the right of the man in a red shirt and yellow hat (12) a second man in a yellow shirt appears on the left end (13) he leaves (14) the man in the red shirt and yellow hat moves one space to the left (15) a man in a red shirt appears on his right

$proof$ = (11) & (6) -> (16): position 6 has person in yellow shirt and no hat; (1) & (12) -> (17): position 1 has person in yellow shirt and no hat; (17) & (13) -> (18): position 1 has no person; (14) & (4) & (5) -> (19): position 4 has person in red shirt and yellow hat; (14) & (5) -> (20): position 5 has no person; (20) & (15) -> (21): position 5 has person in red shirt and no hat;

SCONE (Tangrams):

$question$ = (1) position 1 has figure A (2) position 2 has figure D (3) position 3 has figure E (4) position 4 has figure C (5) position 5 has figure B (6) swap the 1st and 5th figure (7) swap the 1st and 3rd figure (8) swap them back (9) delete the 5th figure (10) add it back

$proof$ = (1) & (6) -> (11): position 1 has figure B; (5) & (6) -> (12): position 5 has figure A; (11) & (7) -> (13): position 1 has figure E; (3) & (7) -> (14): position 3 has figure B; (13) & (8) -> (15): position 1 has figure B; (14) & (8) -> (16): position 3 has figure E; (12) & (9) -> (17): position 5 has no figure; (17) & (10) -> (18): position 5 has figure A;

GSM8K:

$question$ = (1) Adam and Tom are brothers. (2) Adam is 8 years old and (3) Tom is 12 years old. (4) In how many years will their combined age be 44 years old?

$proof$ = (2) & (3) -> (5): At present, the two brothers have a combined age of 8 + 12 = 20 years old.; (5) -> (6): Therefore, 1 year means an increase in the sum of their ages by 1 * 2 = 2 years.; (4) & (5) -> (7): Adam and Tom need a total of 44 - 20 = 24 more years to be 44 years old together.; (6) & (7) -> (8): So both brothers will be 44 years old together after 24 / 2 = 12 years.; (4) & (8) -> (9): The answer is 12;

AQUA-RAT:

$question$ = (1) Three birds are flying at a fast rate of 900 kilometers per hour. (2) What is their speed in miles per minute? (3) [1km = 0.6 miles] (4) A)32400 (5) B)6000 (6) C)600 (7) D)60000 (8) E)10

$proof$ = (0) -> (9): To calculate the equivalent of miles in a kilometer; (3) -> (10): 0.6 kilometers = 1 mile; (10) & (1) -> (11): 900 kilometers = (0.6)*900 = 540 miles; (0) -> (12): In 1 hour there are 60 minutes; (11) & (12) & (2) -> (13): Speed in miles/minutes = 60 * 540 = 32400; (13) & (2) & (4) -> (14): Correct answer - A; (14) -> (15): The answer is A);

AR-LSAT:

$question$ = (1) Four boys - (2) Fred, Juan, Marc, and Paul - (3) and three girls - (4) Nita, Rachel, and Trisha - (5) will be assigned to a row of five adjacent lockers, (6) numbered consecutively 1 through 5, (7) arranged along a straight wall. (8) The following conditions govern the assignment of lockers to the seven children: (9) Each locker must be assigned to either one or two children, (10) and each child must be assigned to exactly one locker. (11) Each shared locker must be assigned to one girl and one boy. (12) Juan must share a locker, (13) but Rachel cannot share a locker. (14) Nita's locker cannot be adjacent to Trisha's locker. (15) Fred must be assigned to locker 3. (16) Which one of the following is a complete and (17) accurate list of the children who must be among those assigned to shared lockers? (18) A) Fred, Juan (19) B) Juan, Paul (20) C) Juan, Marc, Paul (21) D) Juan, Marc, Trisha (22) E) Juan, Nita, Trisha

$proof$ = (1) & (3) & (5) -> (23): Four boys and three girls will be assigned to five adjacent lockers; (10) & (11) & (9) -> (24): Each locker can be assigned to max two children, one girl and one boy, and one child must be assigned to exactly one locker; (12) & (14) -> (25): Kids who can share lockers : Juan, Nita,

```
Trisha; (13) -> (26): Kids not sharing lockers : Rachel; (0) ->
(27): Answer is 22; (27) -> (28): The answer is E);
```

B.2 Textual Logical Units Extraction

In Figure 30 we show the TLU extraction pseudo-code. This simple script is used to extract TLUs from the components of the questions, context and others. The algorithm shown uses Python's module "re" style of regular expression and matching to determine boundaries between TLUs. The pattern values used are "`(.)|(,)|(!)|(?)|(and)|(then)`".

Function: EXTRACT-TEXTUAL-LOGICAL-UNITS
Input: Text of component $text$.
Parameters: Boundary patterns $patterns$.
Output: List of textual logical units $tlus$.

$matches \leftarrow text.regex\_match(patterns)$
$tlus \leftarrow \emptyset$
$lastPos \leftarrow 0$
for $match \in matches$ **do**
  $tlu \leftarrow text.substring(lastPos,\ match.end)$
  if $math.text \in \{$",",“and”,“$then$”$\}$ **then**
    if $tokens(tlu).size \geq 5$ **then**
      $tlus.push(tlu)$
      $lastPos \leftarrow match.end$
    end if
  end if
end for
return $tlus$

Figure 30: textual logical units extraction algorithm.

B.3 Further Annotation Details

Annotation Instructions

All the expert annotators were given instructions with the description of the annotation task, some expected examples and a list of guidelines on how to properly label the data. We held multiple meetings to elucidate any further questions about the annotation effort. The overall annotation instructions are summarized below:

- **Completeness:** The premises (directed edges) should contain all the information needed to ensure that the conclusion nodes are entailed given the premise.
- **Relevance:** Edges connecting nodes should ensure the entailment is correct, and no further irrelevant edges should be included.
- **Purposefulness:** The nodes containing the question and the answer should always be included in the final reasoning graph.
- **Granularity:** While writing the step-by-step rationales (in the case of AR-LSAT), the entailments should be fine-grained, encoding a single inference or logical step.

Annotation User Interface

In Figure 31 we show the user interface used to annotate STREET data with an example from the GSM8K task. The user interface consists of a web-based form with fields containing the question, context, answer choice, and answer. During annotation, each form consisted of a single entailment step in the reasoning graph and the annotators had to select which premises were associated with the current entailment step. For AR-LSAT annotators were also asked to write the step rationale, or conclusion from the entailment step. In Figure 31 the fields on the left-hand side contain the TLUs from the question and context while the fields on the right-hand side contain fields to select the premises for the current reasoning step.

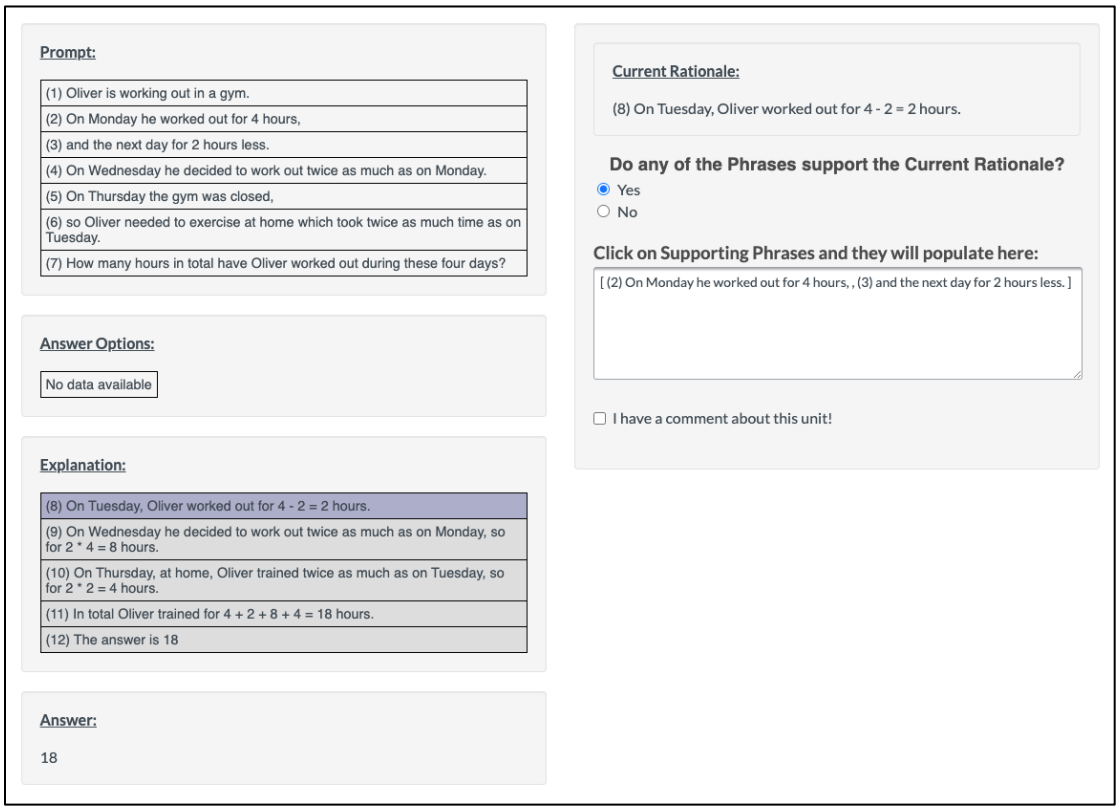

Figure 31: Annotation user interface designed to author the reasoning graphs.

B.4 Evaluation Metrics

The textual similarity function $\sigma(a, b)$ is a binary function that maps the text of two nodes (a and b) to a `TRUE` or `FALSE` value. We use different textual similarity functions for each task in STREET. When a reasoning step contains no antecedents (for instance, when a reasoning step conclusion is used as a supporting fact such as "*one hour has 60 minutes*", then that node might not have antecedents). The similarity used for each task are described below:

SCONE: The nodes in scone follow a well-defined textual structure, therefore the node similarity returns `TRUE` if and only if $a = b$.

GSM8K and AQUA-RAT: We parse the node text and extract all the mathematical values inside the node. For instance, "*Natalia sold 48 / 2 = 24 clips in May*" would be converted to the set $\{48, 2, 24\}$. Then the similarity function returns `TRUE` if the sets extracted from a and b are equal.

ARC and AR-LSAT: Since the conclusions in the reasoning graph can be any arbitrary text use the BLEURT (Sellam, Das, & Parikh, 2020) as the text similarity function. BLEURT is a trained metric that uses the BERT language model and was shown to have better correlation to human judgment than standard metrics like BLEU and ROUGE. We define that BLEURT > 0.25 as the threshold that decides if two texts a and b are similar enough.