PhD Thesis

Koustuv Sinha

Acknowledgements

Abstract

Abstract in French

Contributions to Original Knowledge

Contributions of Authors

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Central Theme of the thesis : Understanding systematicity in pre-trained language models through semantic and syntactic generalization.

In this thesis I discuss my work on understanding systematicity in pre-trained language models.

Chapter 2

Background

2.1 Early methods for text representation

2.2 Neural Inductive bias of text representation

2.2.1 Feed Forward Neural Networks

2.2.2 Recurrent Neural Networks

2.2.3 Transformer Models

Large Language Models (LLMs) are the state-of-the-art in language models, which are based on Transformers.

2.3 Pre-training and the advent of Large Language Models

Success of pre-training and scale

2.4 Systematicity and Generalization

2.4.1 Definitions

1. Productivity

2. Word Order Sensitivity

2.4.2 Tasks

Chapter 3

Understanding semantic generalization through productivity

Natural Language Understanding (NLU) systems have been extremely successful at reading comprehension tasks, such as question answering (QA) and natural language inference (NLI). An array of existing datasets are available for these tasks. This includes datasets that test a system's ability to extract factual answers from text [1, 2, 3, 4, 5], as well as datasets that emphasize commonsense inference, such as entailment between sentences [6, 7].

However, there are growing concerns regarding the ability of NLU systems—and neural networks more generally—to generalize in a systematic and robust way [8, 9, 10]. For instance, recent work has highlighted the brittleness of NLU systems to adversarial examples [11], as well as the fact that NLU models tend to exploit statistical artifacts in datasets, rather than exhibiting true reasoning and generalization capabilities [12, 13]. These findings have also dovetailed with the recent dominance of large pre-trained language models, such as BERT, on NLU benchmarks [14, 15], which suggest that the primary difficulty in these datasets is incorporating the statistics of the natural language, rather than reasoning.

An important challenge is thus to develop NLU benchmarks that can precisely test a model's capability for robust and systematic generalization. Ideally, we want language understanding systems that can not only answer questions and draw inferences from text, but that can also do so in a systematic, logical, and robust way. While such reasoning capabilities are certainly required for many existing NLU tasks, most datasets combine several challenges of language understanding into one, such as coreference/entity resolution, incorporating world knowledge, and semantic parsing—making it difficult to isolate and diagnose a model's capabilities for systematic generalization and robustness.

Inspired by the classic AI challenge of inductive logic programming [16], in this chapter I discuss my work on developing semi-synthetic benchmark designed to explicitly test an NLU model's ability for systematic and robust logical generalization [17]. Our benchmark suite—termed CLUTRR (Compositional Language Understanding and Text-based Relational Reasoning)—contains a large set of semi-synthetic stories involving hypothetical families. Given a story, the goal is to infer the relationship between two family members, whose relationship is not explicitly mentioned. To solve this task, a learning agent must extract the relationships mentioned in the text, induce the logical rules governing the kinship relationships (e.g., the transitivity of the sibling relation), and use these rules to infer the relationship between a given pair of entities. Crucially, the CLUTRR benchmark allows us to test a learning agent's ability for *systematic generalization* by testing on stories that contain unseen combinations of logical rules. CLUTRR also allows us to precisely test for the various forms of *model robustness* by adding different kinds of superfluous *noise facts* to the stories.

3.1 Technical Background

3.1.1 Notations and Terminology

Following standard practice in formal semantics, we use the term *atom* to refer to a *predicate* symbol and a list of terms, such as $[\texttt{grandfatherOf}, X, Y]$, where the predicate `grandfatherOf` denotes the *relation* between the two *variables*, X and Y. We restrict the predicates to have an arity of 2, i.e., binary predicates. A logical *rule* in this setting is of the form $\mathcal{H} \vdash \mathcal{B}$, where $\mathcal{B}$ is the *body* of the rule, i.e., a conjunction of two *atoms* ($[\alpha_1, \alpha_2]$) and $\mathcal{H}$ is the *head*, i.e., a single atom (α) that can be viewed as the goal or query. For instance, given a knowledge base (KB) R that contains the single rule

$$[\texttt{grandfatherOf}, X, Y] \vdash [[\texttt{fatherOf}, X, Z], [\texttt{fatherOf}, Z, Y]], \tag{3.1}$$

the query $[\texttt{grandfatherOf}, X, Y]$ evaluates to true if and only if the body

$$\mathcal{B} = [[\texttt{fatherOf}, X, Z], [\texttt{fatherOf}, Z, Y]] \tag{3.2}$$

is also true in a given world. A rule is called a *grounded* rule if all atoms in the rule are themselves *grounded*, i.e., all variables are replaced with *constants* or entities in a world. A *fact* is a grounded binary predicate. A *clause* is a conjunction of two or more atoms ($\mathcal{C} = (\mathcal{H}_\mathcal{C} \vdash \mathcal{B}_\mathcal{C} = ([\alpha_1, ..., \alpha_n]))$) which can be built using a set of rules.

3.2 Overview and construction of CLUTRR

The core idea behind the CLUTRR benchmark suite is the following: Given a natural language story describing a set of kinship relations, the goal is to infer the relationship between two entities, whose relationship is *not* explicitly stated in the story. To generate these stories, we first design a knowledge base (KB) with rules specifying how kinship relations resolve, and we use the following steps to create semi-synthetic stories based

on this knowledge base:

Step 1. **Generate** a random kinship graph that satisfies the rules in our KB.

Step 2. **Sample a target fact** (i.e., relation) to predict from the kinship graph.

Step 3. **Apply backward chaining** to sample a set of facts that can prove the target relation (and optionally sample a set of "distracting" or "irrelevant" noise facts).

Step 4. **Convert the sampled facts into a natural language story** through pre-specified text templates and crowd-sourced paraphrasing.

The short stories in CLUTRR are essentially narrativized renderings of a set of logical facts. In the following sections, we describe how we sample the logical facts that make up a story by generating random kinship graphs and using backward chaining to produce logical reasoning chains.

3.2.1 Graph generation

To generate a kinship graph (say, G) underlying a particular story, we first sample a set of gendered[1] entities and kinship relations using a stochastic generation process. This generation process contains a number of tunable parameters—such as the maximum number of children at each node, the probability of an entity being married to another entity, etc.—and is designed to produce a valid, but possibly incomplete "backbone graph". For instance, this backbone graph generation process will specify "parent"/"child" relations between entities but does not add "grandparent" relations. After this initial generation process, we recursively apply the logical rules in R to the backbone graph to produce a final graph G that contains the full set of kinship relations between all the entities. [2]

[1]Kinship and gender roles are oversimplified in our data (compared to the real world) to maintain tractability.

[2]In the context of our data generation process, we distinguish between the knowledge base, R, which contains a finite number of predicates and rules specifying how kinship relations in a family resolve, and a particular kinship graph G, which contains a grounded set of atoms specifying the particular kinship

In the CLUTRR Benchmark, the following kinship relations are used: *son, father, husband, brother, grandson, grandfather, son-in-law, father-in-law, brother-in-law, uncle, nephew, daughter, mother, wife, sister, granddaughter, grandmother, daughter-in-law, mother-in-law, sister-in-law, aunt, niece*.

$$
\begin{aligned}
[\texttt{grand}, X, Y] &\vdash [[\texttt{child}, X, Z], [\texttt{child}, Z, Y]], \\
[\texttt{grand}, X, Y] &\vdash [[\texttt{SO}, X, Z], [\texttt{grand}, Z, Y]], \\
[\texttt{grand}, X, Y] &\vdash [[\texttt{grand}, X, Z], \\
&\qquad [\texttt{sibling}, Z, Y]], \\
[\texttt{inv-grand}, X, Y] &\vdash [[\texttt{inv-child}, X, Z], \\
&\qquad [\texttt{inv-child}, Z, Y]], \\
[\texttt{inv-grand}, X, Y] &\vdash [[\texttt{sibling}, X, Z], \\
&\qquad [\texttt{inv-grand}, Z, Y]], \\
[\texttt{child}, X, Y] &\vdash [[\texttt{child}, X, Z], \\
&\qquad [\texttt{sibling}, Z, Y]], \\
[\texttt{child}, X, Y] &\vdash [[\texttt{SO}, X, Z], \\
&\qquad [\texttt{child}, Z, Y]], \\
[\texttt{inv-child}, X, Y] &\vdash [[\texttt{sibling}, X, Z], \\
&\qquad [\texttt{inv-child}, Z, Y]], \\
[\texttt{inv-child}, X, Y] &\vdash [[\texttt{child}, X, Z], \\
&\qquad [\texttt{inv-grand}, Z, Y]], \\
[\texttt{sibling}, X, Y] &\vdash [[\texttt{child}, X, Z], \\
&\qquad [\texttt{inv-un}, Z, Y]], \\
[\texttt{sibling}, X, Y] &\vdash [[\texttt{inv-child}, X, Z], \\
&\qquad [\texttt{child}, Z, Y]]
\end{aligned}
$$

relations that underlie a single story. In other words, R contains the logical rules that govern all the generated stories in CLUTRR, while G contains the grounded facts that underlie a specific story.

$$
\begin{aligned}
[\texttt{sibling}, X, Y] \vdash [&[\texttt{sibling}, X, Z],\\
&[\texttt{sibling}, Z, Y]],\\
[\texttt{in-law}, X, Y] \vdash [&[\texttt{child}, X, Z],\\
&[\texttt{SO}, Z, Y]],\\
[\texttt{inv-in-law}, X, Y] \vdash [&[\texttt{SO}, X, Z],\\
&[\texttt{inv-child}, Z, Y]],\\
[\texttt{un}, X, Y] \vdash [&[\texttt{sibling}, X, Z],\\
&[\texttt{child}, Z, Y]],\\
[\texttt{inv-un}, X, Y] \vdash [&[\texttt{inv-child}, X, Z],\\
&[\texttt{sibling}, Z, Y]],
\end{aligned}
$$

We used a small, tractable, and logically sound KB of rules as mentioned above. We carefully select this set of deterministic rules to avoid ambiguity in the resolution. We use gender-neutral predicates and resolve the gender of the predicate in the head $\mathcal{H}$ of a clause $\mathcal{C}$ by deducing the gender of the second constant. We have two types of predicates, *vertical* predicates (parent-child relations) and *horizontal* predicates (sibling or significant other). We denote all the vertical predicates by its *child-to-parent* relation and append the prefix `inv-` to the predicates for the corresponding *parent-to-child* relation. For example, `grandfatherOf` is denoted by the gender-neutral predicate $[\texttt{inv-grand}, X, Y]$, where the gender is determined by the gender of Y.

3.2.2 Backward chaining

The resulting graph G provides the *background knowledge* for a specific story, as each edge in this graph can be treated as a grounded predicate (i.e., fact) between two entities. From this graph G, we sample the facts that make up the story, as well as the target fact that we seek to predict: First, we (uniformly) sample a target relation $\mathcal{H}_\mathcal{C}$, which is the fact that we want to predict from the story. Then, from this target relation

$\mathcal{H}_\mathcal{C}$, we run a simple variation of the backward chaining [18] algorithm for k iterations starting from $\mathcal{H}_\mathcal{C}$, where at each iteration we uniformly sample a subgoal to resolve and then uniformly sample a KB rule that resolves this subgoal. Crucially, unlike traditional backward chaining, we do not stop the algorithm when a proof is obtained; instead, we run for a fixed number of iterations k in order to sample a set of k facts $\mathcal{B}_\mathcal{C}$ that imply the target relation $\mathcal{H}_\mathcal{C}$.

3.2.3 Adding natural language

So far, we have described the process of generating a conjunctive logical clause $\mathcal{C} = (\mathcal{H}_\mathcal{C} \vdash \mathcal{B}_\mathcal{C})$, where $\mathcal{H}_\mathcal{C} = [\alpha^*]$ is the target fact (i.e., relation) we seek to predict and $\mathcal{B}_\mathcal{C} = [\alpha_1, ..., \alpha_k]$ is the set of supporting facts that imply the target relation. We now describe how we convert this logical representation to natural language through crowdsourcing.

Paraphrasing using Amazon Mechanical Turk

The basic idea behind our approach is that we show Amazon Mechanical Turk (AMT) crowd-workers the set of facts $\mathcal{B}_\mathcal{C}$ corresponding to a story and ask the workers to paraphrase these facts into a narrative. Since workers are given a set of facts $\mathcal{B}_\mathcal{C}$ to work from, they are able to combine and split multiple facts across separate sentences and construct diverse narratives (Figure 3.3).

We use ParlAI [19] Mturk interface to collect paraphrases from the users. Specifically, given a set of facts, we ask the users to paraphrase the facts into a story. The users (*turkers*) are free to construct any story they like as long as they mention all the entities and all the relations among them. We also provide the head $\mathcal{H}$ of the clause as an *inferred* relation and specifically instruct the users to *not* mention it in the paraphrased story. In order to evaluate the paraphrased stories, we ask the turkers to peer review a story paraphrased by a different turker. Since there are two tasks - paraphrasing a

story and rating a story - we choose to pay 0.5$ for each annotation. A sample task description in our MTurk interface is as follows:

> In this task, you will need to write a short, simple story based on a few facts. **It is crucial that the story mentions each of the given facts at least once.** The story does not need to be complicated! It just needs to be grammatical and mention the required facts.
>
> After writing the story, you will be asked to evaluate the quality of a generated story (based on a different set of facts). **It is crucial that you check whether the generated story mentions each of the required facts.**
>
> *Example of good and bad stories: Good Example*
>
> **Facts to Mention**
>
> - John is the father of Sylvia.
> - Sylvia has a brother Patrick.
>
> **Implied Fact**: John is the father of Patrick.
>
> **Written story**
>
> John is the proud father of the lovely Sylvia. Sylvia has a love-hate relationship with her brother Patrick.
>
> *Bad Example*
>
> **Facts to Mention**
>
> - Vincent is the son of Tim.
> - Martha is the wife of Tim.
>
> **Implied Fact** : Martha is Vincent's mother.
>
> **Written story**
>
> Vincent is married at Tim and his mother is Martha.

The reason the above story is bad:

- This story is bad because it is nonsense / ungrammatical.
- This story is bad because it does not mention the proper facts.
- This story is bad because it reveals the implied fact.

A sample of the AMT interface is shown in Figure 3.1. To ensure that the turkers are providing high-quality annotations without revealing the inferred fact, we also launch another task to ask the turkers to rate three annotations to be either good or bad which are provided by a set of *different* turkers. We pay 0.2$ for each HIT consisting of three reviews. This helped to remove logical and grammatical inconsistencies to a large extent. Based on the reviews, 79% of the collected paraphrases passed the peer-review sanity check where all the reviewers agree on the quality. This subset of the placeholders is used in the benchmark. A sample of programmatically generated dataset for clause length of $k = 2$ to $k = 6$ is provided in the tables 3.3 to 3.7.

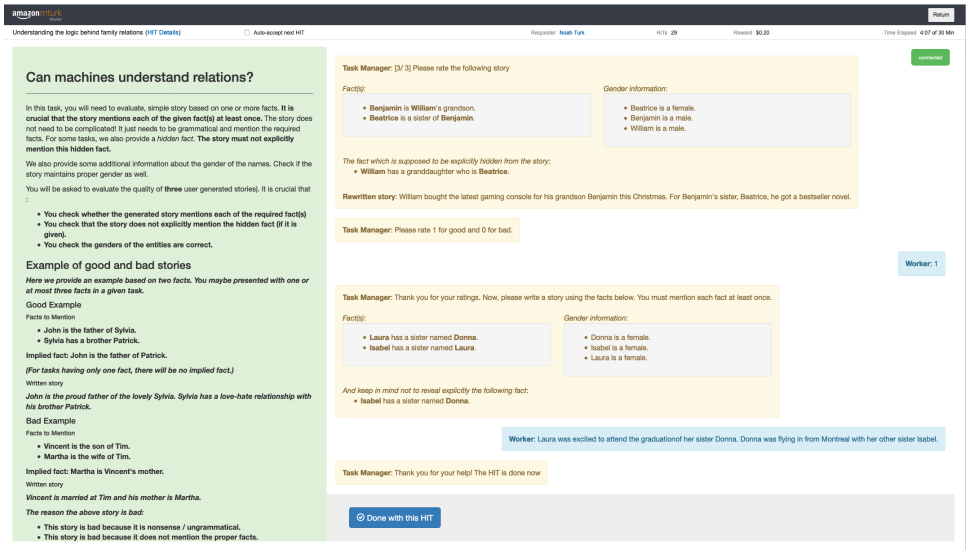

Figure 3.1 Amazon Mechanical Turker Interface built using ParlAI which was used to collect data as well as peer reviews.

Reusability and composition

One challenge for data collection via AMT is that the number of possible stories generated by CLUTRR grows combinatorially as the number of supporting facts increases, i.e., as $k = |\mathcal{B}_\mathcal{C}|$ grows. This combinatorial explosion for large k—combined with the difficulty of maintaining the quality of the crowd-sourced paraphrasing for long stories—makes it infeasible to obtain a large number of paraphrased examples for $k > 3$. To circumvent this issue and increase the flexibility of our benchmark, we reuse and compose AMT paraphrases to generate longer stories. In particular, we collected paraphrases for stories containing $k = 1, 2, 3$ supporting facts and then replaced the entities from these collected stories with placeholders in order to re-use them to generate longer semi-synthetic stories. An example of a story generated by stitching together two shorter paraphrases is provided below:

> [Frank] went to the park with his father, [Brett]. [Frank] called his brother [Boyd] on the phone. He wanted to go out for some beers. [Boyd] went to the baseball game with his son [Jim].
>
> Q: What is [Brett] and [Jim]'s relationship?

Thus, instead of simply collecting paraphrases for a fixed number of stories, we instead obtain a diverse collection of natural language templates that can be programmatically recombined to generate stories with various properties.

3.2.4 AMT Template statistics

At the time of submission, we have collected 6,016 unique paraphrases with an average of 19 paraphrases for every possible logical clause of length $k = 1, 2, 3$. Table 3.1 contains summary statistics of the collected paraphrases. Overall, we found high linguistic diversity in the collected paraphrases. For instance, the average Jaccard overlap in unigrams between pairs paraphrases corresponding to the same logical clause was only 0.201 and only 0.0385 for bigrams.

Number of Paraphrases			# clauses
	$k = 1$	1,868	20
	$k = 2$	1,890	58
	$k = 3$	2,258	236
	Total	6,016	
Unique Word Count		3,797	
Jaccard Word Overlap	Unigrams	0.201	
	Bigrams	0.0385	

Table 3.1 Statistics of the AMT paraphrases. Jaccard word overlap is calculated within the templates of each individual clause of length k.

3.2.5 Human performance

Relation Length	Human Performance		Reported Difficulty
	Time Limited	Unlimited Time	
2	0.848	1	1.488 +- 1.25
3	0.773	1	2.41 +- 1.33
4	0.477	1	3.81 +- 1.46
5	0.424	1	3.78 +- 0.96
6	0.406	1	4.46 +- 0.87

Table 3.2 Human performance accuracies on CLUTRR dataset. Humans are provided the Clean-Generalization version of the dataset, and we test on two scenarios: when a human is given limited time to solve the task, and when a human is given unlimited time to solve the task. Regardless of time, our evaluators provide a score of difficulty of individual puzzles.

To get a sense of the data quality and difficulty involved in CLUTRR, we asked human annotators to solve the task for random examples of length $k = 2, 3, ..., 6$. (Table 3.2) We perform the evaluation in two scenarios: first a time-limited scenario where we ask AMT Turkers to solve the puzzle in a fixed time. Turkers were provided a maximum time of 30 mins, but they solved the puzzles in an average of 1 minute 23 seconds. Secondly, we use another set of expert evaluators who are given ample time to solve the tasks. Not surprisingly, if a human being is given ample time (experts took an average of 6 minutes per puzzle) and a pen and a paper to aid in the reasoning, they get all the relations correct. However, if an evaluator is short of time, they might miss

important details on the relations and perform poorly. Thus, our tasks require *active attention*.

We found that time-constrained AMT annotators performed well (i.e., $> 70\%$) accuracy for $k \leq 3$ but struggled with examples involving longer stories, achieving 40-50% accuracy for $k > 3$. However, trained annotators with unlimited time were able to solve 100% of the examples, highlighting the fact that this task requires attention and involved reasoning, even for humans.

3.2.6 Representing the question and entities

The AMT paraphrasing approach described above allows us to convert the set of supporting facts $\mathcal{B}_\mathcal{C}$ to a natural language story, which can be used to predict the target relation/query $\mathcal{H}_\mathcal{C}$. However, instead of converting the target query, $\mathcal{H}_\mathcal{C} = [\alpha^*]$, to a natural language question, we instead opt to represent the target query as a K-way classification task, where the two entities in the target relation are provided as input and the goal is to classify the relation that holds between these two entities. This representation avoids the pitfall of revealing information about the answer in the question [13].

When generating stories, entity names are randomly drawn from a set of 300 common gendered English names. Thus, depending on each run, the entities are never the same. This ensures that the entity names are simply placeholders and uncorrelated from the task.

3.3 Experimental Setups

The modular nature of CLUTRR provides rich diagnostic capabilities for evaluating the robustness and generalization abilities of neural language understanding systems. We highlight some key diagnostic capabilities available via different variations of CLUTRR

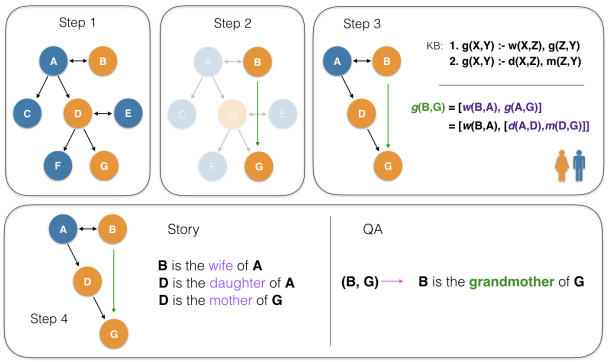

Figure 3.2 Data generation pipeline. Step 1: generate a kinship graph. Step 2: sample a target fact. Step 3: Use backward chaining to sample a set of facts. Step 4: Convert sampled facts to a natural language story.

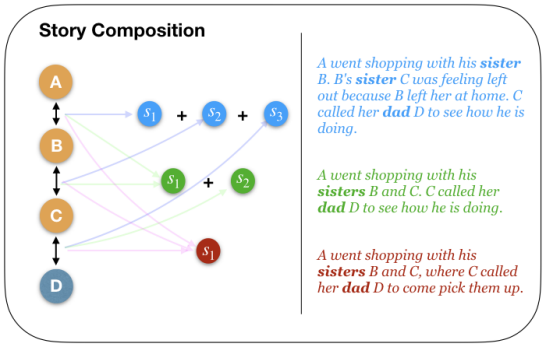

Figure 3.3 Illustration of how a set of facts can split and combined in various ways across sentences.

below. These diagnostic variations correspond to the concrete datasets that we generated in this work, and we describe the results on these datasets in section 3.5.

3.3.1 Systematic generalization on Productivity

Most prominently, CLUTRR allows us to explicitly evaluate a model's ability for systematic generalization, especially productivity. In particular, we rely on the following hold-out procedures to test systematic generalization:

- During training, we hold out a subset of the collected paraphrases, and we only use

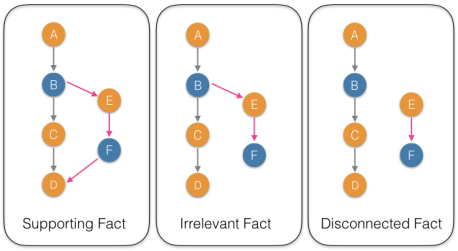

Figure 3.4 Noise generation procedures of CLUTRR.

this held-out subset of paraphrases when generating the test set. Thus, to succeed on CLUTRR, an NLU system must exhibit *linguistic generalization* and be robust to linguistic variation at test time.

- We also hold out a subset of the logical clauses during training (for clauses of length $k > 2$).[3] In other words, during training, the model sees all logical rules but does not see all *combinations* of these logical rules. Thus, in addition to linguistic generalization, success on this task also requires *logical generalization*.
- Lastly, as a more extreme form of both logical and linguistic generalization, we consider the setting where the models are trained on stories generated from clauses of length $\leq k$ and evaluated on stories generated from larger clauses of length $> k$. Thus, we explicitly test the ability for models to generalize on examples that require more steps of reasoning that any example they encountered during training.

3.3.2 Robust Reasoning

In addition to evaluating systematic generalization, the modular setup of CLUTRR also allows us to diagnose model robustness by adding *noise facts* to the generated narratives. Due to the controlled semi-synthetic nature of CLUTRR, we are able to

[3]One should not holdout clauses from length $k = 2$ in order to allow models to learn the compositionality of all possible binary predicates.

provide a precise taxonomy of the kinds of noise facts that can be added (Figure 3.4). In order to structure this taxonomy, it is important to recall that any set of supporting facts $\mathcal{B}_\mathcal{C}$ generated by CLUTRR can be interpreted as a path, $p_\mathcal{C}$, in the corresponding kinship graph G (Figure 3.2). Based on this interpretation, we view adding noise facts from the perspective of sampling three different types of noise paths, p_n, from the kinship graph G:

- *Irrelevant facts*: We add a path p_n, which has exactly one shared end-point with p_c. In this way, this is a *distractor* path, which contains facts that are connected to one of the entities in the target relation, $\mathcal{H}_\mathcal{C}$, but do not provide any information that could be used to help answer the query.
- *Supporting facts*: We add a path p_n, whose two end-points are on the path $p_\mathcal{C}$. The facts on this path p_n are noise because they are not needed to answer the query, but they are supporting facts because they can, in principle, be used to construct alternative (longer) reasoning paths that connect the two target entities.
- *Disconnected facts*: We add paths which neither originate nor end in any entity on p_c. These disconnected facts involve entities and relations that are completely unrelated to the target query.

3.3.3 Generated Datasets

For all experiments, we generated datasets with 10-15k training examples. In many experiments, we report training and testing results on stories with different clause lengths k. (For brevity, we use the phrase "clause length" throughout this section to refer to the value $k = |\mathcal{B}_\mathcal{C}|$, i.e., the number of steps of reasoning that are required to predict the target query.) In all cases, the training set contains 5000 train stories per k value, and, during testing, all experiments use 100 test stories per k value. All experiments were run 10 times with different randomly generated stories, and means and standard errors over these 10 runs are reported. As discussed above, during training

we holdout 20% of the paraphrases, as well as 10% of the possible logical clauses.

Table 3.3 Snapshot of puzzles in the dataset for k=2

Puzzle	Question	Gender	Answer
Charles's son *Christopher* entered rehab for the ninth time at the age of thirty. *Randolph* had a nephew called *Christopher* who had n't seen for a number of years.	Randolph is the _____ of Charles	Charles:male, Christopher:male, Randolph:male	brother
Randolph and his sister *Sharon* went to the park. *Arthur* went to the baseball. game with his son *Randolph*	Sharon is the _____ of Arthur	Arthur:male, Randolph:male, Sharon:female	daughter
Frank went to the park with his father, *Brett*. *Frank* called his brother *Boyd* on the phone. He wanted to go out for some beers.	Brett is the _____ of Boyd	Boyd:male, Frank:male, Brett:male	father

Table 3.4 Snapshot of puzzles in the dataset for k=3

Puzzle	Question	Gender	Answer
Roger was playing baseball with his sons *Sam* and *Leon*. *Sam* had to take a break though because he needed to call his sister *Robin*.	Leon is the _____ of Robin	Robin:female, Sam:male, Roger:male, Leon:male	brother
Elvira and her daughter *Nancy* went shopping together last Monday and they bought new shoes for *Elvira*'s kids. *Pedro* and his sister *Allison* went to the fair. *Pedro*'s mother, *Nancy*, was out with friends for the day.	Elvira is the _____ of Allison	Allison:female, Pedro:male, Nancy:female, Elvira:female	grandmother
Roger met up with his sister *Nancy* and her daughter *Cynthia* at the mall to go shopping together. *Cynthia*'s brother *Pedro* was going to be the star in the new show.	Pedro is the _____ of Roger	Roger:male, Nancy:female, Cynthia:female, Pedro:male	nephew

3.4 Evaluated Models

Our primary baselines are neural language understanding models that take unstructured text as input. We consider bidirectional LSTMs [20, 21] (with and without attention), as well as models that aim to incorporate inductive biases towards relational reasoning: Relation Networks (RN) [22], Relational Recurrent Networks (RMC) [23] and Compositional Memory Attention Network (MAC) [24]. We also use the large

Table 3.5 Snapshot of puzzles in the dataset for k=4

Puzzle	Question	Gender	Answer
Celina has been visiting her sister, *Fran* all week. *Fran* is also the daughter of *Bethany*. *Ronald* loves visiting his aunt *Bethany* over the weekends. *Samuel*'s son *Ronald* entered rehab for the ninth time at the age of thirty.	Celina is the _____ of Samuel	Samuel:male, Ronald:male, Bethany:female, Fran:female, Celina:female	niece
Celina adores her daughter *Bethany*. *Bethany* loves her very much, too. *Jackie* called her mother *Bethany* to let her know she will be back home soon. *Thomas* was helping his daughter *Fran* with her homework at home. Afterwards, *Fran* and her sister *Jackie* played Xbox together.	Celina is the _____ of Thomas	Thomas:male, Fran:female, Jackie:female, Bethany:female, Celina:female	daughter
Raquel is *Samuel* 'daughter and they go shopping at least twice a week together. *Kenneth* and her mom, *Theresa*, had a big fight. *Theresa*'s son, *Ronald*, refused to get involved. *Ronald* was having an argument with her sister, *Raquel*.	Samuel is the _____ of Kenneth	Kenneth:male, Theresa:female, Ronald:male, Raquel:female, Samuel:male	father

pre-trained language model, BERT [14], as well as a modified version of BERT having a trainable LSTM encoder on top of the pretrained BERT embeddings. All of these models (except BERT) were re-implemented in PyTorch 1.0 [25] and adapted to work with the CLUTRR benchmark.

Since the underlying relations in the stories generated by CLUTRR inherently form a graph, we also experiment with a Graph Attention Network (GAT) [26]. Rather than taking the textual stories as input, the GAT baseline receives a structured graph representation of the facts that underlie the story.

Entity and query representations. We use the various baseline models to encode the natural language story (or graph) into a fixed-dimensional embedding. With the exception of the BERT models, we do not use pre-trained word embeddings and learn the word embeddings from scratch using end-to-end backpropagation. An important

Table 3.6 Snapshot of puzzles in the dataset for k=5

Puzzle	Question	Gender	Answer
Steven's son is *Bradford*. *Bradford* and his father always go fishing together on Sundays and have a great time together. *Diane* is taking her brother *Brad* out for a late dinner. *Kristin*, *Brad*'s mother, is home with a cold. *Diane*'s father *Elmer*, and his brother *Steven*, all got into the rental car to start the long cross-country roadtrip they had been planning.	Bradford is the _____ of Kristin	Kristin:female, Brad:male, Diane:female, Elmer:male, Steven:male, Bradford:male	nephew
Elmer went on a roadtrip with his youngest child, *Brad*. *Lena* and her sister *Diane* are going to a restaurant for lunch. *Lena*'s brother *Brad* is going to meet them there with his father *Elmer* *Brad* ca n't stand his unfriendly aunt *Lizzie*.	Lizzie is the _____ of Diane	Diane:female, Lena:female, Brad:male, Elmer:male, Lizzie:female	aunt
Ira took his niece *April* fishing Saturday. They caught a couple small fish. *Ronald* was enjoying spending time with his parents, *Damion* and *Claudine*. *Damion*'s other son, *Dennis*, wanted to come visit too. *Dennis* often goes out for lunch with his sister, *April*.	Ira is the _____ of Claudine	Claudine:female, Ronald:male, Damion:male, Dennis:male, April:female, Ira:male	brother

Table 3.7 Snapshot of puzzles in the dataset for k=6

Puzzle	Question	Gender	Answer
Mario wanted to get a good gift for his sister, *Marianne*. *Jean* and her sister *Darlene* were going to a party held by *Jean*'s mom, *Marianne*. *Darlene* invited her brother *Roy* to come, too, but he was too busy. *Teri* and her father, *Mario*, had an argument over the weekend. However, they made up by Monday. *Agnes* wants to make a special meal for her daughter *Teri*'s birthday.	Roy is the _____ of Agnes	Agnes:female, Teri:female, Mario:male, Marianne:female, Jean:female, Darlene:female, Roy:male	nephew
Robert's aunt, *Marianne*, asked *Robert* to mow the lawn for her. *Robert* said he could n't because he had a bad back. *William*'s parents, *Brian* and *Marianne*, threw him a surprise party for his birthday. *Brian*'s daughter *Jean* made a mental note to be out of town for her birthday! *Agnes*'s biggest accomplishment is raising her son *Robert*. *Jean* is looking for a good gift for her sister *Darlene*.	Darlene is the _____ of Agnes	Agnes:female, Robert:male, Marianne:female, William:male, Brian:male, Jean:female, Darlene:female	niece
Sharon and her brother *Mario* went shopping. *Teri*, *Mario*'s daughter, came too. *Agnes*, *Annie*'s mother, is unhappy with *Robert*. She feels her son is cruel to *Annie*'s sister *Teri*, and she wants *Robert* to be nicer. *Robert*'s sister, *Nicole*, participated in the dance contest.	Nicole is the _____ of Sharon	Sharon:female, Mario:male, Teri:female, Annie:female, Agnes:female, Robert:male, Nicole:female	niece

note, however, is that we perform Cloze-style anonymization [27] of the entities (i.e., names) in the stories, where each entity name is replaced by a *@entity-k* placeholder, which is randomly sampled from a small, fixed pool of placeholder tokens. The embeddings for these placeholders are randomly initialized and fixed during training.

To make a prediction about a target query given a story, we concatenate the embedding of the story (generated by the baseline model) with the embeddings of the two target entities and we feed this concatenated embedding to a 2-layer feed-forward neural network with a softmax prediction layer.

3.4.1 Hyperparameters

For all models, the common hyperparameters used are: Embedding dimension: 100 (except BERT based models), Optimizer: Adam, Learning rate: 0.001, Number of epochs: 100, Number of runs: 10. Specific model-based hyperparameters are given as follows:

- **Bidirectional LSTM**: LSTM hidden dimension: 100, # layers: 2, Classifier MLP hidden dimension: 200
- **Relation Networks**: f_{θ_1} : 256, f_{θ_2}: 64, g_θ : 64
- **MAC**: # Iterations: 6, `shareQuestion`: True, Dropout - Memory, Read and Write: 0.2
- **Relational Recurrent Networks**: Memory slots: 2, Head size: 192, Number of heads: 4, Number of blocks : 1, forget bias : 1, input bias: 0, gate style: unit, key size: 64, # Attention layers: 3, Dropout: 0
- **BERT**: Layers : 12, Fixed pretrained embeddings from `bert-base-uncased` using Pytorch HuggingFace BERT repository [4], Word dimension: 768, appended

[4] https://github.com/huggingface/pytorch-pretrained-BERT

with a two-layer MLP for final prediction.

- **BERT-LSTM**: Same parameters as above, with a two-layer unidirectional LSTM encoder on top of BERT word embeddings.
- **GAT**: Node dimension: 100, Message dimension: 100, Edge dimension: 20, number of rounds: 3

3.5 Results

We evaluate several NLU systems on the proposed CLUTRR benchmark to surface the relative strengths and shortcomings of these models in the context of inductive reasoning and combinatorial generalization.[5] We aim to answer the following key questions:

(**Q1**) How do state-of-the-art NLU models compare in terms of systematic generalization? Can these models generalize to stories with unseen combinations of logical rules?

(**Q2**) How does the performance of neural language understanding models compare to a graph neural network that has full access to graph structure underlying the stories?

(**Q3**) How robust are these models to the addition of noise facts to a given story?

3.5.1 Systematic Generalization with Productivity

We begin by using CLUTRR to evaluate the ability of the baseline models to perform systematic generalization (**Q1**). In this setting, we consider two training regimes: in the first regime, we train all models with clauses of length $k = 2, 3$, and in the second regime, we train with clauses of length $k = 2, 3, 4$. We then test the generalization of these models on test clauses of length $k = 2, ..., 10$.

[5] Code to reproduce all the results in this section are available at https://github.com/facebookresearch/clutrr/.

Figure 3.5 illustrates the performance of different models on this generalization task. We observe that the GAT model is able to perform near-perfectly on the held-out logical clauses of length $k = 3$, with the BERT-LSTM being the top-performer among the text-based models but still significantly below the GAT. Not surprisingly, the performance of all models degrades monotonically as we increase the length of the test clauses, which highlights the challenge of "zero-shot" systematic generalization [9, 28]. However, as expected, all models improve on their generalization performance when trained on $k = 2, 3, 4$ rather than just $k = 2, 3$ (Figure 3.5, right). The GAT, in particular, achieves the biggest gain by this expanded training.

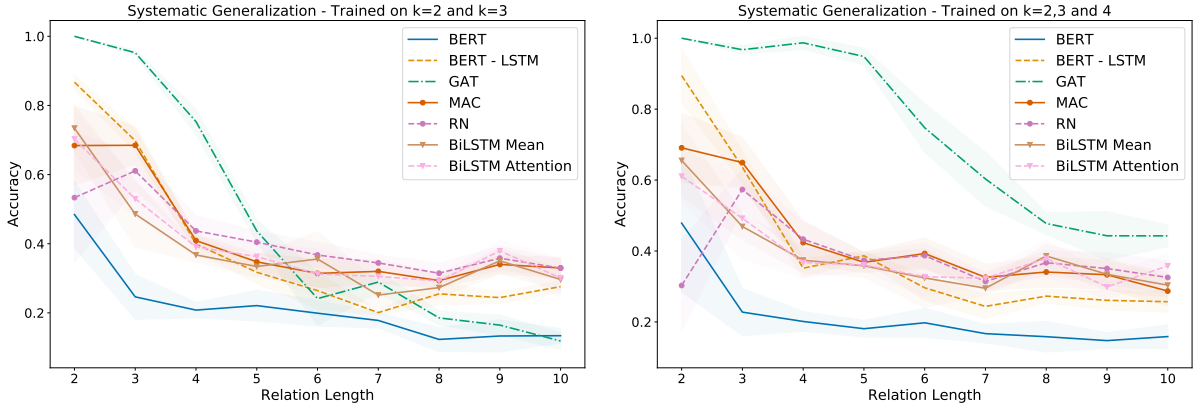

Figure 3.5 Systematic generalization performance of different models when trained on clauses of length $k = 2, 3$ (Left) and $k = 2, 3, 4$ (Right).

3.5.2 The benefit of structure

The empirical results on systematic generalization also provide insight into how the text-based NLU systems compare against the graph-based GAT model that has full access to the logical graph structure underlying the stories (**Q2**). Indeed, the relatively strong performance of the GAT model (Figure 3.5) suggests that the language-based models fail to learn a robust mapping from the natural language narratives to the underlying logical facts.

Models		Unstructured models (no graph)						Structured model (with graph)
Training	Testing	BiLSTM - Attention	BiLSTM - Mean	RN	MAC	BERT	BERT-LSTM	GAT
Clean	Clean	0.58 ±0.05	0.53 ±0.05	0.49 ±0.06	0.63 ±0.08	0.37 ±0.06	0.67 ±0.03	**1.0** ±0.0
	Supporting	**0.76** ±0.02	0.64 ±0.22	0.58 ±0.06	0.71 ±0.07	0.28 ±0.1	0.66 ±0.06	0.24 ±0.2
	Irrelevant	0.7 ±0.15	**0.76** ±0.02	0.59 ±0.06	0.69 ±0.05	0.24 ±0.08	0.55 ±0.03	0.51 ±0.15
	Disconnected	0.49 ±0.05	0.45 ±0.05	0.5 ±0.06	0.59 ±0.05	0.24 ±0.08	0.5 ±0.06	**0.8** ±0.17
Supporting	Supporting	0.67 ±0.06	0.66 ±0.07	0.68 ±0.05	0.65 ±0.04	0.32 ±0.09	0.57 ±0.04	**0.98** ±0.01
Irrelevant	Irrelevant	0.51 ±0.06	0.52 ±0.06	0.5 ±0.04	0.56 ±0.04	0.25 ±0.06	0.53 ±0.06	**0.93** ±0.01
Disconnected	Disconnected	0.57 ±0.07	0.57 ±0.06	0.45 ±0.11	0.4 ±0.1	0.17 ±0.05	0.47 ±0.06	**0.96** ±0.01
Average		**0.61** ±0.08	0.59 ±0.08	0.54 ±0.07	**0.61** ±0.06	0.30 ±0.07	0.56 ±0.05	**0.77** ±0.09

Table 3.8 Testing the robustness of the various models when training and testing on stories containing various types of noise facts. The types of noise facts (supporting, irrelevant, and disconnected) are defined in Section .

To further confirm this trend, we ran experiments with modified train and test splits for the text-based models, where the same set of natural language paraphrases were used to construct the narratives in both the train and test splits. In this simplified setting, the text-based models must still learn to reason about held-out logical patterns, but the difficulty of parsing the natural language is essentially removed, as the same natural language paraphrases are used during testing and training. We found that the text-based models were competitive with the GAT model in this simplified setting, confirming that the poor performance of the text-based models on the main task is driven by the difficulty of parsing the unseen natural language narratives.

3.5.3 Robust Reasoning

Finally, we use CLUTRR to systematically evaluate how various baseline neural language understanding systems cope with noise (**Q3**). In all the experiments we provide a combination of $k = 2$ and $k = 3$ length clauses in training and testing, with noise facts being added to the train and/or test set depending on the setting (Table 3.8). We use the different types of noise facts defined in Section 3.3.2..

Overall, we find that the GAT baseline outperforms the unstructured text-based

models across most testing scenarios (Table 3.8), which showcases the benefit of a structured feature space for robust reasoning. When training on clean data and testing on noisy data, we observe two interesting trends that highlight the benefits and shortcomings of the various model classes:

1. All the text-based models excluding BERT actually perform better when testing on examples that have *supporting* or *irrelevant* facts added. This suggests that these models actually benefit from having more content related to the entities in the story. Even though this content is not strictly useful or needed for the reasoning task, it may provide some linguistic cues (e.g., about entity genders) that the models exploit. In contrast, the BERT-based models do not benefit from the inclusion of this extra content, which is perhaps due to the fact that they are already built upon a strong language model (e.g., that already adequately captures entity genders.)
2. The GAT model performs poorly when *supporting* facts are added but has no performance drop when *disconnected* facts are added. This suggests that the GAT model is sensitive to changes that introduce cycles in the underlying graph structure but is robust to the addition of noise that is disconnected from the target entities.

Moreover, when we trained on noisy examples, we found that only the GAT model was able to consistently improve its performance (Table 3.8). Again, this highlights the performance gap between the unstructured text-based models and the GAT.

3.6 Related Work

To design the CLUTRR dataset, we draw inspiration from the classic work on inductive logic programming (ILP), a long line of reading comprehension benchmarks in NLP, as well as work combining language and knowledge graphs.

3.6.1 Reading comprehension benchmarks

Many datasets have been proposed to test the reading comprehension ability of NLP systems. This includes the SQuAD [1], NewsQA [3], and MCTest [29] benchmarks that focus on factual questions; the SNLI [6] and MultiNLI [7] benchmarks for sentence understanding; and the bABI tasks [30], to name a few. Our primary contribution to this line of work is the development of a carefully designed *diagnostic* benchmark to evaluate model robustness and systematic generalization in the context of NLU.

3.6.2 Systematic generalization

A growing body of literature has demonstrated that NLU models tend to exploit statistical artifacts in datasets and lack true generalization capabilities [11, 12, 13, 9]. These critical examinations have dovetailed with similar studies on visual question answering [31, 8, 10]. CLUTRR, contributes to this growing area by introducing a principled and flexible benchmark to evaluate systematic generalization in the context of language understanding—with our notion of systematic generalization being grounded in classic work on inductive logic programming (ILP) [16].

3.6.3 Question-answering with knowledge graphs

Our work is also related to the domain of question answering and reasoning in knowledge graphs [32, 33, 34, 35, 36, 37, 38], where either the model is provided with a knowledge graph to perform inference over or where the model must infer a knowledge graph from the text itself. However, unlike previous benchmarks in this domain—which are generally *transductive* and focus on leveraging and extracting knowledge graphs as a source of background knowledge about a fixed set of entities—CLUTRR requires *inductive logical reasoning*, where every example requires reasoning over a new set of previously unseen entities.

3.7 Discussion

In this paper we introduced the CLUTRR benchmark suite to test the systematic generalization and inductive reasoning capababilities of NLU systems. We demonstrated the diagnostic capabilities of CLUTRR and found that existing NLU systems exhibit relatively poor robustness and systematic generalization capabilities—especially when compared to a graph neural network that works directly with symbolic input. These results highlight the gap that remains between machine reasoning models that work with unstructured text and models that are given access to more structured input.

3.8 Follow-up findings in the community

Chapter 4

Quantifying syntactic generalization using word order

Paper [**?**]

4.1 Technical Background

4.2 Word Order in Natural Language Inference

4.2.1 Probe Construction

4.3 Experiments & Results

4.4 Related Work

4.5 Discussion

4.6 Follow-up findings in the community

Chapter 5

Probing syntax understanding through distributional hypothesis

Paper: [**?**]

5.1 Technical Background

5.2 Dataset construction and pre-training

5.3 Experiments

5.3.1 Downstream reasoning tasks

5.3.2 Evaluating the effectiveness of probing syntax

5.4 Related Work

5.5 Discussion

5.6 Follow-up findings in the community

Chapter 6

Measuring systematic generalization by exploiting absolute positions

6.1 Technical Background

6.2 Systematic understanding of absolute position embeddings

6.3 Related Work

6.4 Experiments

6.5 Discussion

Chapter 7

Conclusion

7.1 Summary

7.2 Limitations

7.3 Future Work

Bibliography

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[2] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

[3] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. NewsQA: A machine comprehension dataset. *arXiv preprint*, pages 1–12, 2016.

[4] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of NAACL-HLT*, pages 839–849, 2016.

[5] Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572, 2016.

[6] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, 2015.

[7] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1112–1122, 2018.

[8] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*, 2019.

[9] Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2879–2888, 2018.

[10] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE, 2017.

[11] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, 2017.

[12] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A Smith. Annotation artifacts in natural language inference

data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, 2018.

[13] Divyansh Kaushik and Zachary C Lipton. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, 2018.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

[16] J R Quinlan. Learning logical definitions from relations. *Mach. Learn.*, 5(3):239–266, August 1990.

[17] Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China, November 2019. Association for Computational Linguistics.

[18] Herve Gallaire and Jack Minker. *Logic and Data Bases*. Perseus Publishing, 1978.

[19] Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. Parlai: A dialog research software platform.

In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 79–84, 2017.

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

[21] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[22] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[23] Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. Relational recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 7310–7321, 2018.

[24] Drew Arad Hudson and Christopher D. Manning. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*, 2018.

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro

Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[27] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[28] Shagun Sodhani, Sarath Chandar, and Yoshua. Bengio. On Training Recurrent Neural Networks for Lifelong Learning. *arXiv e-prints*, November 2018.

[29] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, 2013.

[30] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-Complete question answering: A set of prerequisite toy tasks. 2015.

[31] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*, 2016.

[32] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*, 2018.

[33] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1980–1990, 2018.

[34] Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2026–2037. Curran Associates, Inc., 2018.

[35] Z. Wang, L. Li, D. D. Zeng, and Y. Chen. Attention-based multi-hop reasoning for knowledge graph. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 211–213, Nov 2018.

[36] Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573, 2017.

[37] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302, 2018.

[38] Dimitri Kartsaklis, Mohammad Taher Pilehvar, and Nigel Collier. Mapping text to knowledge graph entities using multi-sense lstms. *arXiv preprint arXiv:1808.07724*, 2018.

Glossary

Transformers A class of models first derived by Vaswani et al. 2017. 2

Acronyms

LLMs Large Language Models. 2

NLU Natural Language Understanding. 4, 5, 24

Chapter 8

Appendix

8.1 Org mode auto save

Run the following snippet to auto save and compile in org mode.

```
(defun kdm/org-save-and-export ()
(interactive)
(if (and (eq major-mode 'org-mode)
    (ido-local-file-exists-p (concat (file-name-sans-extension (buffer-name
  (org-latex-export-to-latex)))

(add-hook 'after-save-hook 'kdm/org-save-and-export)
```

8.2 Remove "parts" from report

```
(add-to-list 'org-latex-classes
             '("report-noparts"
               "\\documentclass[11pt]{report}"
               ("\\chapter{%s}" . "\\chapter*{%s}")
```

```
                ("\\section{%s}" . "\\section*{%s}")
                ("\\subsection{%s}" . "\\subsection*{%s}")
                ("\\subsubsection{%s}" . "\\subsubsection*{%s}")))
```

8.3 Add newpage before a heading

```
(defun org/get-headline-string-element  (headline backend info)
  (let ((prop-point (next-property-change 0 headline)))
    (if prop-point (plist-get (text-properties-at prop-point headline) :pa

(defun org/ensure-latex-clearpage (headline backend info)
  (when (org-export-derived-backend-p backend 'latex)
    (let ((elmnt (org/get-headline-string-element headline backend info)))
      (when (member "newpage" (org-element-property :tags elmnt))
        (concat "\\clearpage\n" headline)))))

(add-to-list 'org-export-filter-headline-functions
             'org/ensure-latex-clearpage)
```

8.4 Glossary and Acronym build using Latexmk

Add the following snippet in the file "~/.latexmkrc": (Source: `https://tex.stackexchange.com/a/44316`)

```
add_cus_dep('glo', 'gls', 0, 'run_makeglossaries');
add_cus_dep('acn', 'acr', 0, 'run_makeglossaries');
```

```
sub run_makeglossaries {
    my ($base_name, $path) = fileparse( $_[0] ); #handle -outdir param by s
    pushd $path; # ... cd-ing into folder first, then running makeglossarie

    if ( $silent ) {
        system "makeglossaries -q '$base_name'"; #unix
        # system "makeglossaries", "-q", "$base_name"; #windows
    }
    else {
        system "makeglossaries '$base_name'"; #unix
        # system "makeglossaries", "$base_name"; #windows
    };

    popd; # ... and cd-ing back again
}

push @generated_exts, 'glo', 'gls', 'glg';
push @generated_exts, 'acn', 'acr', 'alg';
$clean_ext .= ' %R.ist %R.xdy';
```

8.5 Citation style buffer local

```
(set (make-local-variable 'bibtex-completion-format-citation-functions)
  '((org-mode      . my/bibtex-completion-format-citation-org-default-cite
```

8.6 Org latex compiler options

```
(setq org-latex-pdf-process (list "latexmk -f -pdf -%latex -interaction=nor
```

Original value

```
(setq org-latex-pdf-process (list "latexmk -f -pdf %f"))
```

Let us try Fast compile `https://gist.github.com/yig/ba124dfbc8f63762f222`.

```
(setq org-latex-pdf-process (list "latexmk-fast %f"))
```

- Doesn't seem to work from Emacs.
- I need to change the save function to only export in tex. Then, have a separate process run latexmk.
- Using the python package `when-changed` to watch the thesis.tex file for change.
- Usage:

```
when-changed thesis.tex latexmk -f -pdf -interaction=nonstopmode -output-di
```

- The pdf does not update. It seems to but not always? No it does. For some reason, compilation takes ages.
- Works with `when-changed`!