The Thesis Committee for Zayne Sprague
certifies that this is the approved version of the following thesis:

Additive Deductive Embeddings for Planning of Natural Language Proofs

SUPERVISING COMMITTEE:

Greg Durrett, Supervisor

Swarat Chaudhuri

Additive Deductive Embeddings for Planning of Natural Language Proofs

by

Zayne Sprague

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

The University of Texas at Austin

May 2023

Abstract

Additive Deductive Embeddings for Planning of Natural Language Proofs

Zayne Sprague, M.S.Comp.Sc.
The University of Texas at Austin, 2023

SUPERVISOR: Greg Durrett

Constructing a logical argument to support a claim entails selecting relevant evidence from a collection of facts and formulating a sequence of reasoning steps. Current natural language systems designed for claim validation employ large language models to reason and plan deductions. In this paper, we investigate whether embedding spaces can adequately represent natural language deductions for use in proof-generating systems. We introduce a novel method for planning natural language proof generation called *Additive Deduction*, which leverages simple arithmetic operations performed exclusively in an embedding space. We explore multiple sources of off-the-shelf dense embeddings in addition to sparse embeddings from BM25. We devise three experiments to demonstrate the effectiveness of embedding models for natural language planning. The first experiment consists of two intrinsic evaluations of Additive Deduction using two off-the-shelf sentence encoders. The second incorporates an embedding-based heuristic into planning for natural language proof generation datasets, namely EntailmentBank and Everyday Norms: Why Not. Lastly, we create a dataset that benchmarks various reasoning categories and common reasoning failures. Our findings suggest that while standard embedding methods frequently embed conclusions near the sums of their premises, they lack the ability to fully express certain categories of reasoning, hurting their proof generation performance.

Table of Contents

List of Tables . . . 7
List of Figures . . . 8
Chapter 1: Introduction . . . 9
Chapter 2: Problem Description and Motivation . . . 12
2.1 Problem Setup . . . 12
2.2 Additive Reasoning . . . 13
Chapter 3: Proof Generation . . . 14
3.1 Heuristic . . . 14
3.1.1 Additive Deduction Heuristic . . . 14
3.1.2 SCSearch Heuristic . . . 15
3.1.3 Caching . . . 15
3.2 Deductive Step Model . . . 16
3.3 Reasoning Validation . . . 17
3.4 Entailment Scores . . . 17
Chapter 4: Heuristics and Datasets . . . 18
4.1 Heuristics . . . 18
4.1.1 BM25 . . . 18
4.1.2 SCSearch . . . 18
4.1.3 SimCSE . . . 19
4.1.4 GPT3 . . . 19
4.2 Datasets . . . 19
4.2.1 EntailmentBank . . . 19
4.2.2 Everyday Norms: Why Not (ENWN) . . . 19
4.2.3 Single-Step Reasoning Contrast dataset . . . 20
Chapter 5: Experiments . . . 22
5.1 Intrinsic Evaluation . . . 22
5.1.1 Comparing Deduction Embedding Representations . . . 22
5.1.2 Ranking Gold Steps . . . 23
5.1.3 Results . . . 24
5.2 Generating Proofs . . . 25
5.2.1 Results . . . 26
5.3 Single-Step Reasoning Contrast Dataset . . . 27
5.3.1 Results . . . 28

Chapter 6: Discussion 31
Chapter 7: Related work 32
Chapter 8: Conclusion 34
Works Cited 35

List of Tables

3.1 Examples taken from the Single-Step Reasoning Contrast (SSRC) dataset. The Category and Perturbation column shows which reasoning category is used in the deduction as well as what type of perturbation is applied to the premise. The perturbed premise is then used to create invalid premise pairs (where one premise could be a gold premise, but the other is perturbed) such that when the two are combined, their deduction does not lead to the conclusion. There are ten examples per reasoning category, and each example has multiple perturbed premises for each of the four perturbation types. 16

5.1 We look at the average cosine similarity score of different summed premise pairs with their textual deduction embedding. We see large gaps between a Random set of premises and Partial set (meaning one is a gold premise required for the deduction) with their Gold set of premises, aligning with expectation. The Model columns shows that there no loss in representing deductions if the deduction is the gold annotation or from the deduction step model. 23

5.2 Comparison against different heuristics on the Mean Reciprocal Rank of selecting gold premises conditioned on their immediate deduction and the goal of the tree. GPT3 outperforms BM25, indicating that there are more complex reasoning steps required than just lexical overlap. However, SCSearch still outperforms all methods by as much as 0.26. 23

5.3 Generated proofs per heuristic on the two datasets. BM25 has high performance on both of these datasets, indicating that textual overlap is enough to plan reasoning steps for nearly 50% of the examples. SimCSE (AD) and GPT3 (AD) underperforms BM25 on ENWN, this could mean that these methods are not as sensitive to lexical overlap as BM25 is. SCSearch still outperforms every baseline by as much as 38% showing that a lot of reasoning is unaccounted for in the other methods. 26

5.4 Results of each heuristic on SSRC broken down by the reasoning category and averaged over the individual perturbation types of each category. We separate SCSearch and SimCSE (AD) from the BM25 and GPT3 (AD) heuristics as the BM25 and GPT3 (AD) have not been trained on any natural language inference data (with the possibility that GPT3 may have seen some incidental examples of inferences in its pretraining), making them close to zero-shot on this task. SCSearch and SimCSE (AD) have both been fine-tuned on reasoning datasets (EntailmentBank and NLI respectively). 30

5.5 Results of each heuristic on SSRC broken down by the perturbation type and averaged over the individual reasoning categories. We again separate SCSearch and SimCSE (AD) from the BM25 and GPT3 (AD) heuristics. 30

List of Figures

1.1 Overview of our approach. Initial facts and claims are given as input to an embedding model, like GPT3. We want the summed embeddings of facts to be close in the embedding space to statements that can be deductively inferred from those premises. Given a target claim to prove, this constitutes a planning heuristic for selecting facts in a natural language proof generation system. 10

4.1 Distribution of cosine similarities for examples in EntailmentBank T2 and ENWN. We expect that Random and Partial should not overlap with Gold, Model, or G. to S. because that would indicate random summed premise embeddings are close to the embedded deduction indicating an invalid embedding space. We see that the distributions align well with expectations with little overlap between Rand and a small overlap with Partial. 21

5.1 Overall scores of each heuristic on the SSRC dataset. GPT3 (AD) outperforms SCSearch slightly on this benchmark, differing from the intrinsic evaluations and end-to-end experiments. This implies that the types of reasoning involved in these datasets may be skewed towards a specific category. 27

5.2 Comparison plot of the heuristic methods versus the SCSearch heuristic. If a point is above the green line, then that method outperformed SCSearch. Circles indicate reasoning categories and X-marks indicate perturbation types. BM25 underperforms all other methods, showing that the dataset is not sensitive to lexical overlap. 29

Chapter 1: Introduction

Humans can integrate diverse pieces of information to form coherent and persuasive arguments. By employing logical reasoning techniques such as deduction or induction, we assemble structured rationales that effectively support a claim. The ways facts can be combined through reasoning are numerous, including many different modes of deduction like syllogism or modus tollens. This process can be automated with natural language processing, using systems to generate natural language proofs that use evidence to derive a claim through a structured argument. Large language models (LLMs) like GPT4 (OpenAI, 2023) have exhibited impressive performance in reasoning tasks. However, these models can still make unsound inferences.

One reason for these errors is that models may fail to plan reasoning effectively. LLMs do not have explicit planning capabilities: they generate conclusions in a way that conflates lexical choice and decisions of what to generate. However, a separate line of work (Creswell et al., 2023; Sprague et al., 2022; Bostrom et al., 2022, 2021) shows how to decouple these stages. What is missing is a method of doing this planning that is scalable; past work frequently involves early-fusion invocation of pre-trained LMs (Xiong et al., 2021) and does not scale to thousands of premises.

This work explores the feasibility of planning the reasoning process directly in a vector space. We introduce *Additive Deductive Embeddings*, a property of an embedding space that should enable this planning. Each piece of evidence is embedded into a fixed-size vector, and we want these vectors to have the property that the combined embeddings of two facts are close to embeddings of statements that are entailed from those two facts via deduction. Ideally, if we are repeatedly applying deductions to reach a goal, facts that bring us closer to that goal will be useful in the natural language proof process, enabling us to use this vector space as a heuristic during search. Planning is performed by summing fact embeddings and maximizing

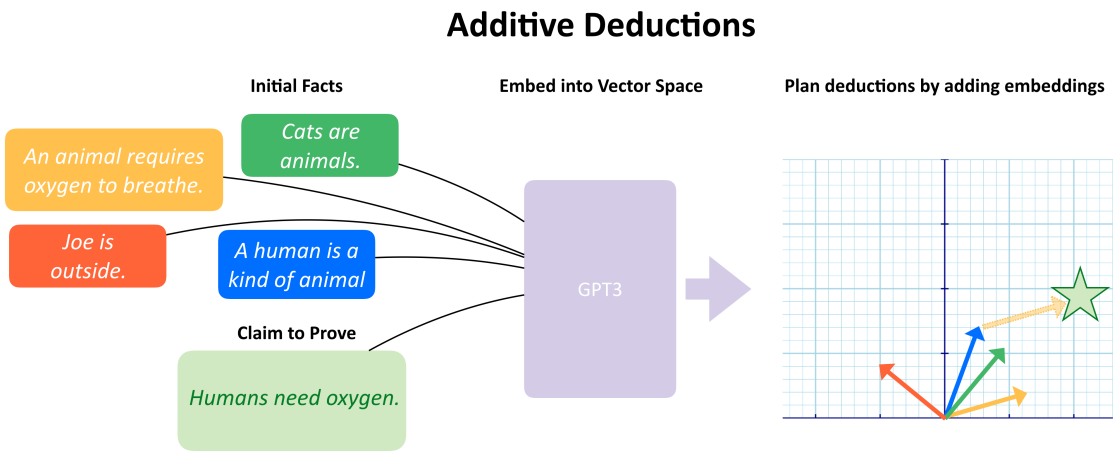

Figure 1.1: Overview of our approach. Initial facts and claims are given as input to an embedding model, like GPT3. We want the summed embeddings of facts to be close in the embedding space to statements that can be deductively inferred from those premises. Given a target claim to prove, this constitutes a planning heuristic for selecting facts in a natural language proof generation system.

the dot product with the encoding of a claim, facilitating rapid retrieval through efficient search algorithms. Our approach is outlined in Figure 1.1.

We explore several questions about Additive Deductive Embeddings. We first conduct intrinsic evaluations on existing embedding methods to see whether they exhibit the additive deductive property. These evaluations look at the distributions of added embeddings to ensure that they are close to the embedding of a deduction.

The second experiment focuses on proof generation using embedding-based heuristics. We use the Entailment Bank (Dalvi et al., 2021) and Everyday Norms: Why Not (Sprague et al., 2022) datasets, which consist of entailment tree examples. To evaluate our approach, we remove intermediate deductions, providing only the leaves (initial facts) and the claim to the system. We then use the step-by-step system outlined in Bostrom et al. (2022) and modify it by employing a new heuristic based on Additive Deductive Embeddings to facilitate planning, determining which facts on the deductive fringe should be combined to reconstruct the entailment tree. This experiment is meant to determine how well these vector-based methods perform

in realistic settings compared to leading state-of-the-art planning systems.

Lastly, we introduce and evaluate on a new dataset, the Single-Step Reasoning Contrast (SSRC) dataset, featuring 14 distinct reasoning categories such as Syllogisms, Analogies, and Modus Ponens. Each category contains ten single-step deduction examples and four different perturbation types that can be applied to either premise where the reasoning example would be invalid if used. These four perturbation types test the embeddings' ability to correctly rank true deductions above invalid deductions. We use MRR to measure the performance of each model on the SSRC dataset. This experiment is meant to provide a nuanced measure of performance across multiple reasoning categories and failure cases which can help determine areas that are lacking for both vector-based and early-fusion pre-trained LM planning.

Our main contributions are threefold: (1) We propose a novel method for planning reasoning steps over a collection of facts purely based on vector arithmetic. (2) We show that Additive Deductive Embeddings have promise, but do not fully meet the properties required for planning in natural language deduction scenarios; (3) We present a new dataset meant to help diagnose and identify areas where deduction planning methods are underperforming across a range of different reasoning categories.

Chapter 2: Problem Description and Motivation

2.1 Problem Setup

Our study explores the process of proving a goal statement (or claim) g by generating an entailment tree T, given a set of general-purpose facts $X = x_1, \ ... \ x_n$ and a collection of instance-specific facts $F = f_1, \ ... \ f_m$. Instance-specific facts typically pertain to the context or background of a particular scenario, while general-purpose facts can be applied more broadly. An example can be seen in Figure 1.1, where F consists of one node, "Joe is outside." and all other facts belong to the set of general-purpose facts, X. T is a binary-branching tree with its leaves being members of X and its non-leaf nodes (which we also call *intermediates*) being new statements generated via deductive reasoning. The root of T must logically entail g.

There are three task settings that vary in difficulty based on the size of the general-purpose fact set X. The first setting, denoted as Task 1 (T1) (Dalvi et al., 2021), provides only the general-purpose facts relevant to the construction of the gold entailment tree, making it the easiest setting as it eliminates the need to sift through irrelevant facts. The second setting, denoted as Task 2 (T2) (Dalvi et al., 2021), includes both the relevant facts and distractor facts that appear lexically pertinent but are ultimately unnecessary for constructing the tree T. Lastly, the third setting, denoted as Task 3 (T3) (Dalvi et al., 2021), encompasses all facts from a large corpus like Wikipedia as the general-purpose fact set X, rendering it the most challenging but also the most realistic setting. In all these settings, the task involves iteratively building the entailment tree through deductions until the original goal g is entailed. Our experiments will focus on the T2 setting.

We follow past work on these tasks (Bostrom et al., 2022; Sprague et al., 2022) where the intermediate nodes of the entailment tree are generated from a pre-trained language model. Specifically, given two premise statements p_a and p_b, we assume

access to a model $P(d_{ab} \mid p_a, p_b)$ that places a distribution over valid deductions d given the two sentences. If the two sentences do not combine to yield any meaningful new conclusions, the behavior of this system is not well-defined.

2.2 Additive Reasoning

Our method focuses on evaluating sentence encoders E that embed sentences in $\mathbb{R}^d$. Our method tests if common encoders, like GPT3, have an additive deductive property useful for planning deductions. Specifically, the encoder E, when provided with two premises p_a, p_b, and their deduction d_{ab} (generated in natural language), produces encodings e_a, e_b, and e_d, respectively, such that the cosine similarity of the added embeddings for the two premises is near 1 for the deductive embedding: $\cos(e_a + e_b, e_d) \approx 1.0$, where $\cos(a, b)$ is the cosine similarity of a and b. Our ultimate goal is to encode all premises $X = p_1, \ ... \ p_n$ into the embedding space and compare sums of their embeddings against the embedding of the goal g. This is the Additive Deductive Embeddings property, which enables us to use a heuristic based on "additive reasoning" in a search loop.

If the cosine similarity between premises and their deductions holds in the vector space, it would indicate that the vectors can model complex logical relationships in their encoding, which can be expressed through simple operations. With perfect encodings, we could efficiently plan the construction of entailment trees without generating the actual deductions in natural language. As a result, the construction and planning of logical arguments could be reduced to a simpler search in a non-natural language space. We explore this question of perfect representations in our first experiment.

Chapter 3: Proof Generation

We begin with our evidence collection X. A heuristic is used to rank the usefulness of deductions derived from a set of premises in proving the claim. A deductive step model then selects the highest-scoring step and generates a set of deductions.[1] These deductions are validated and added back to the evidence collection X for the heuristic to rank. This process is repeated until the $maxSteps$ limit is reached.

3.1 Heuristic

Our search algorithm maintains a deductive fringe comprising potential steps that can be taken based on the original premises and current deductions. A heuristic is employed to rank which steps (premise pairs) are most likely to prove the claim. This fringe must be updated each time a deduction is performed, as the generations from that step are paired with the original premises and other intermediate deductions in the fringe.

3.1.1 Additive Deduction Heuristic

We utilize the Additive Deductive Embeddings to construct a heuristic for prioritizing the fringe. We rely on embeddings of the premises p and the goal g using our embedding method. For each step, we encode any new deductions from the previous step and then sum all the pairs to create deductive representations $e_{d'}$ for hypothetical deduced pairs d'; we denote this as a representation d' to differentiate from an embedding of an actual deduction e_d. We then compute the cosine similarity of each $e_{d'}$ with e_g (the goal embedding), which is used as a score to select the next

[1] To thoroughly explore the space of all plausible deductions, we sample multiple, k, generations ($k = 5$ in all our experiments).

step $S_i = \underset{d'}{\text{argmax}}\ M(e_{d'}, e_g)$. Finally, S_i is provided to the deductive step model to generate new deductions that will be scored in subsequent steps.

3.1.2 SCSearch Heuristic

Past work like SCSearch (Bostrom et al., 2022) has used heuristics with a substantially different structure. These heuristics use language models like DeBERTa to score premise pairs conditioned on a claim. Specifically, these models are of the form $w^\top E(p_1, p_2, g)$; they encode p_1, p_2, and g jointly with an encoder model. A linear layer is then used to predict a logit value used for ranking. These models are trained as binary classifiers on EntailmentBank by selecting positive examples of premise pairs that eventually lead to g and negative examples of unrelated premise pairs. This allows the language model to determine if the immediate deduction would be beneficial towards deducing the claim that it is conditioning on. It also allows the language model to see the claim and premise pairs in context and model interactions between them. Because these methods can perform non-linear transformations and are using Transformers to score the premise pair, these models are strictly more expressive than the Additive Deduction heuristic.

3.1.3 Caching

Certain heuristics, such as the one we construct using Additive Deduction, can cache the encodings of the initial evidence pool X. This offers significant time savings in completing the first step of a search procedure (where a non-cached method would need to set up and rank the pairs for the initial set). However, any subsequent deductions will need to be encoded since they cannot be precomputed and cached. We also found the time savings to be relatively limited in the $T1$ and $T2$ settings, since n is relatively small, so we do not expand on this capability further.

Category and Perturbation	Premises	Conclusion	Perturbed Premises
Categorical Syllogism, Negation	All cats are animals. Whiskers is a cat.	Whisksers is an Animal.	Some cats are not animals. Whiskers is not a cat.
Causal Reasoning, Irrelevant Fact	High levels of stress cause anxiety. Linda has been under a lot of stress lately.	Linda may develop anxiety.	Anxiety disorders can also manifest as physical symptoms. ...
Comparative Reasoning, Incorrect Quantifier	John is stronger than Mary. Mary is stronger than Sue.	John is stronger than Sue.	Some women are stronger than Sue.
Temporal Reasoning, False Premise	The store is open for 12 hours. The store opens at 9 AM.	The store closes at 9 PM.	The store is open for 10 hours.

Table 3.1: Examples taken from the Single-Step Reasoning Contrast (SSRC) dataset. The Category and Perturbation column shows which reasoning category is used in the deduction as well as what type of perturbation is applied to the premise. The perturbed premise is then used to create invalid premise pairs (where one premise could be a gold premise, but the other is perturbed) such that when the two are combined, their deduction does not lead to the conclusion. There are ten examples per reasoning category, and each example has multiple perturbed premises for each of the four perturbation types.

3.2 Deductive Step Model

The deductive step model is trained using the EntailmentBank dataset following Bostrom et al. (2022). We transform the annotated entailment trees into individual steps $T_i = (x_1, x_2 \rightarrow c)$ and fine-tune a pre-trained language model to generate the deduction given a set of premises. We do not use data from (Bostrom

et al., 2021).

3.3 Reasoning Validation

To ensure that the search space generates well-reasoned deductions, we implement a set of validators that examine both the types of steps being taken and the generations produced by the step models following Sprague et al. (2022). Firstly, we employ a Consanguinity Thresholder validation algorithm to ensure that the search procedure does not permit steps to consist of the same premise or premises that result in immediate deductions. For instance, if p_a and p_b create the deduction d_{ab}, we disallow a new step to be (p_a, d_{ab}). This approach effectively promotes diversity in the types of steps being taken. We also enforce that no generation from a step model is an exact duplicate of one of the inputs.

Furthermore, to avoid identifying high-ranking pairs of premises that result in illogical deductions due to hallucination, we devise a new validation method to ensure consistency. The Deduction Agreement validator compares the embedding of the added premises $e_{d'}$ with the embedding of the generated deduction e_d. If the cosine similarity falls below a threshold t_{da}, the step is filtered out. A running average of all $M(e_{d'}, e_d)$ for previous deductions is maintained. If a branch in the entailment tree generates too many deductions not accurately represented by the Additive Deductive Embeddings, it will be filtered out.

3.4 Entailment Scores

We employ a DeBERTa model, fine-tuned on the MNLI and WaNLI tasks, to assess the entailment of each generated natural language deduction. If a deduction achieves a score above a predefined threshold, t_g, it is considered to have recovered the goal g. Once a deduction has successfully recovered the goal, we can trace back the steps used to create that specific deduction, resulting in a minimal proof tree that contains only the essential steps required to prove the goal.

Chapter 4: Heuristics and Datasets

To measure the performance of our heuristic method in proving claims, we explore a variety of benchmarks and metrics to test its ability to match deduction embeddings with summed premise embeddings, to solve proofs, and to reason across a variety of different types of deductions.

4.1 Heuristics

We consider four heuristics: BM25, a sparse retrieval method; the original heuristic from previous work, SCSearch, which employs an early-fusion premise ranker model; a pre-trained sentence encoder, SimCSE as a candidate source for Additive Deductive Embeddings; and finally, GPT3 Ada embeddings as another candidate for Additive Deductive Embeddings.

4.1.1 BM25

BM25 (Robertson et al., 1995) matches items in an index with a query via sparse vector representations, capturing lexical overlap but not deeper semantic similarity. We index all premises and deductions, then query based on the goal.

4.1.2 SCSearch

SCSearch (Bostrom et al., 2022) uses a fine-tuned DeBERTa Large model to rank premises below an intermediate deduction higher than premises in unrelated trees or steps.

4.1.3 SimCSE

SimCSE (Gao et al., 2021) is an encoder that produces sentence embeddings optimized using a contrastive objective.[1] We test for Additive Deduction Embeddings created by this encoder.

4.1.4 GPT3

We use OpenAI's embedding endpoint to create sentence embeddings using the Ada model (Brown et al., 2020). We test for Additive Deduction Embeddings created by this encoder as well.

4.2 Datasets

We use three datasets in our experiments on the heuristics.

4.2.1 EntailmentBank

This dataset comprises annotated entailment trees for textbook-based science facts (Dalvi et al., 2021). We used this dataset for training the majority of our models in a T1 setting. We evaluate the models on the test slice of entailment trees for the T2 task setting.

4.2.2 Everyday Norms: Why Not (ENWN)

ENWN (Sprague et al., 2022) contains annotated entailment trees for common scenarios that individuals encounter in everyday life. ENWN aims to combine common social rules deductively to determine whether a person should perform a

[1]Although we do not fine-tune this method, the training for SimCSE was performed on natural language inference (NLI) examples from MNLI and SNLI datasets. While this is not the same as EntailmentBank data, it is not quite the same as other embedding-based or language model-based methods, in that it has been fine-tuned on a related task. We still place it in the zero-shot bucket for the purposes of this work.

particular action (usually something they should not do). ENWN currently does not have a T2 or T3 setting.

4.2.3 Single-Step Reasoning Contrast dataset

Both EntailmentBank and ENWN test a subset of logical inference types, but do not necessarily have broad coverage. For example, EntailmentBank has very few examples involving negation, despite this being a very important phenomenon to model in practice. We, therefore, want to test whether our embedding methods can handle a wider range of cases.

We construct a new dataset that examines the most common forms of logical reasoning via synthesized examples. We consider fourteen categories, including Analogy, Categorical Syllogism, Causal reasoning, Classification, Comparison, Composition, Division, Modus Ponens, Modus Tollens, Definition, Temporal Logic, Propositional Logic, Quantificational Logic, and Spatial Relationship. For each category, we use GPT 3.5 to generate ten examples of deductions given two premises using the corresponding reasoning category. For all ten examples, we perturb the premises in four ways, making them either *negated*, a *false statement*, an *irrelevant fact*, or with an *incorrect quantifier*. Examples from the dataset are shown in Table 3.1.

Given these perturbed premises, we now generate premise sets. Each deduction can have 2 negated premises (one for each original premise), 2 false premises, up to 30 irrelevant facts, and 2 premises with incorrect quantifiers. We can then use these perturbed premises to create premise pairs that are incorrect (they do not yield the deduction). We create a set of m premise pairs, with only one representing a valid deduction. This creates 496 irrelevant fact premise pairs, 10 negation pairs, 10 incorrect quantifier pairs, and 10 false premise pairs, where in each of those sets only 1 is the gold premise pair. Some examples may have fewer pairs if a generated perturbed premise was duplicated. Our goal is for heuristics to rank the correct premises for the deduction higher than the distractors. We can evaluate this with

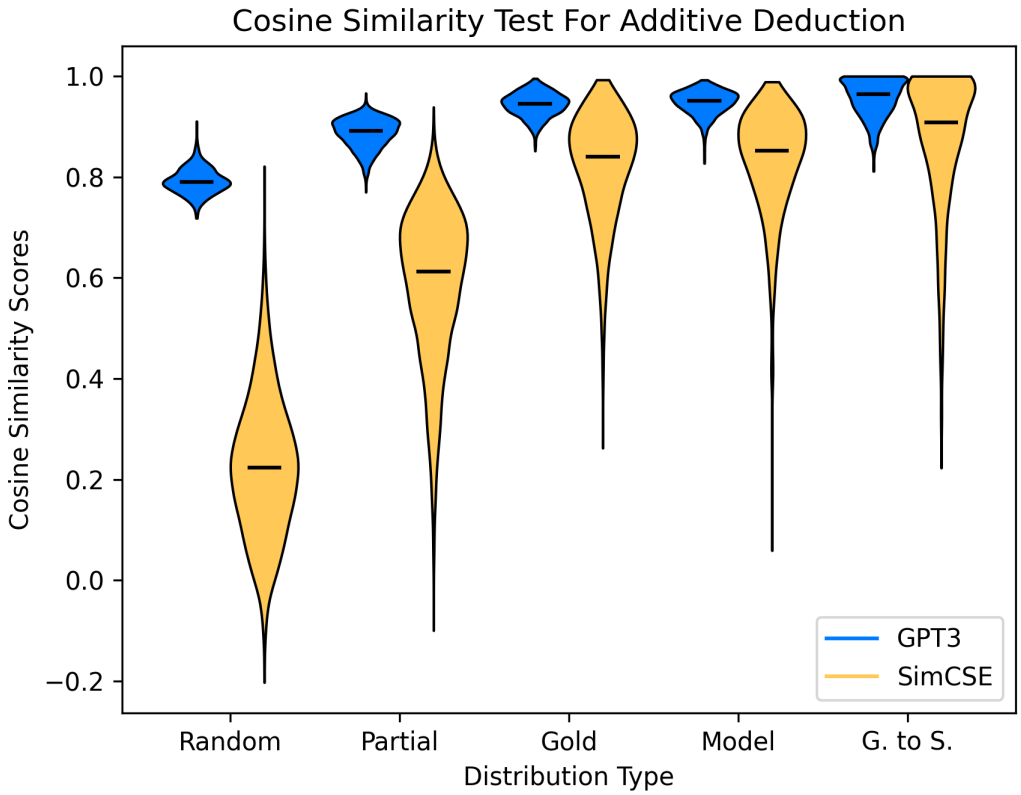

Figure 4.1: Distribution of cosine similarities for examples in EntailmentBank T2 and ENWN. We expect that Random and Partial should not overlap with Gold, Model, or G. to S. because that would indicate random summed premise embeddings are close to the embedded deduction indicating an invalid embedding space. We see that the distributions align well with expectations with little overlap between Rand and a small overlap with Partial.

mean reciprocal rank (MRR) of the correct premise pair in the set.

Chapter 5: Experiments

5.1 Intrinsic Evaluation

We performed two intrinsic evaluations measuring the distributions of cosine similarity scores when using Additive Deductive Embeddings on random, partially random, and gold premises conditioned on a deduction. The second intrinsic evaluation isolates the planning step for proof generation and evaluates how well each heuristic can rank a gold premise pair given their deduction in a set of unrelated premise pairs.

5.1.1 Comparing Deduction Embedding Representations

In our first intrinsic evaluation, we measure the cosine similarity distributions using Additive Deduction Embeddings of premise pairs and a deduction in three settings. The first setting uses a deduction d_{ab} and measures the cosine similarity of its embedding $E(d_{ab})$ with a random premise pair $P_r = \{p_x, p_y\}$ where p_x and p_y are drawn randomly from the set of premises, $U(P)$. The next setting looks at partially random premise pairs, $P_p = \{p_a, p_y\}$ where p_a is one of the gold premises $P_g = \{p_a, p_b\}$ that yield the deduction d_{ab}. Finally, we measure the distribution of scores for the gold premise pair P_g and the deduction that follows from those premises d_{ab}. These three settings correspond to Random, Partial, Gold respectively in Figure 4.1.

Additionally, we also compared the gold premise pair $P_g = \{p_a, p_b\}$ with the deductive step models generated deduction $S_d(p_a, p_b) = d'_{ab}$ and measured its cosine similarity $cos(E(p_a)+E(p_b), E(d'_{ab}))$. Finally, we measured the cosine similarity scores of the annotated deductions and the generated deductions $cos(E(d_{ab}), E(d'_{ab}))$; this is a sort of sanity check to see if our Additive Deductive Embedding property holds. If the deductive step model generated inconsistent deductions from the annotations, the planning module might suffer from not being able to model the deductive distributions

generated by the step model S_d, adding an additional layer of complexity. These settings correspond to Model and G. to S. respectively in Figure 4.1, all settings have their averages reported in Table 5.1 as well.

	Entailment Bank				ENWN			
Heuristic	**Rand**	**Partial**	**Gold**	**Model**	**Rand**	**Partial**	**Gold**	**Model**
SimCSE	0.25	0.62	0.85	0.85	0.14	0.48	0.72	0.76
GPT3	0.79	0.88	0.93	0.94	0.79	0.89	0.95	0.95

Table 5.1: We look at the average cosine similarity score of different summed premise pairs with their textual deduction embedding. We see large gaps between a Random set of premises and Partial set (meaning one is a gold premise required for the deduction) with their Gold set of premises, aligning with expectation. The Model columns shows that there no loss in representing deductions if the deduction is the gold annotation or from the deduction step model.

	Entailment Bank		ENWN
Heuristic	**Deductive**	**Goal**	**Deductive**
BM25	0.47	0.21	0.50
SCSearch	0.78	0.39	0.82
SimCSE (AD)	0.46	0.20	0.59
GPT3 (AD)	0.54	0.24	0.56

Table 5.2: Comparison against different heuristics on the Mean Reciprocal Rank of selecting gold premises conditioned on their immediate deduction and the goal of the tree. GPT3 outperforms BM25, indicating that there are more complex reasoning steps required than just lexical overlap. However, SCSearch still outperforms all methods by as much as 0.26.

5.1.2 Ranking Gold Steps

The second intrinsic evaluation measures the rankings of premise pairs, P_{pairs}, conditioned on a deduction embedding, $E(d_{ab})$, where one pair is the gold premise pair $P_g = \{p_a, p_b\}$ which yield the deduction. All other pairs are either random $P_r = \{p_x, p_y\}$, where p_x and p_y are sampled uniformly from the set of premises $U(P)$, or are partially random $P_p = \{p_a, p_y\}$. The full list of premise pairs is the union of all

these sets $P_{\text{pairs}} = P_g \cup P_p \cup P_r$. We shuffle P_{pairs}, and then we calculate scores for each pair according to how each heuristic scores premise pairs, scores = $\{\text{heuristic}(P_s, d_{ab}) \mid P_s \in P_{\text{pairs}}\}$. For the heuristics using Additive Deduction Embeddings (AD), the scores are cosine similarities, scores = $\{cos(E(p_n) + E(p_m), E(d_{ab})) \mid \{p_n, p_m\} \in P_{\text{pairs}}\}$. Finally, we sort scores and find the rank of the gold premise pair.

We calculate the mean reciprocal rank (MRR) using the ranks of the gold premise pairs across all examples in the EntailmentBank T2 and Everyday Norms: Why Not datasets. We also repeat this process for EntailmentBank T2 where we condition on the claim g instead of the immediate deduction d_{ab}. Because the claim g is often a product of multiple deductions in the premise set P, we expect the MRR scores to be far lower than the scores on the immediate deductions d_{ab}. ENWN does not have a T2 setting, so we do not show the claim-conditioned scores because every premise would be related to the claim g making nearly all pairs valid. These are shown in Table 5.2, a number closer to 1.0 indicates that the gold premise pair was consistently ranked higher than partial and random premise pairs.

5.1.3 Results

Table 5.1 show that the cosine similarity distributions between random and partial premise pairs are lower than the gold for both GPT3 and SimCSE aligning with expectations for Additive Deductive Embeddings. We see this result in Figure 4.1 as well, where the random and partial pair distributions has low overlap with the distribution of gold pairs. There is more overlap between partial and gold than random and gold pairs, which is also expected since one of the premises in the partial pair is used in the natural language deduction. We also see high agreement between the gold premise pair and the generated deduction, indicating that deductions generated by the step model are similar to the annotated deductions which is important for proof generation where annotated intermediate deductions are unavailable.

Table 5.2 shows the BM25 MRR scores as being quite competitive with the Ad-

ditive Deduction Embedding methods, SimcSE and GPT3, all of which are within 0.1 of each other. BM25s high performance indicates that the datasets EntailmentBank T2 and ENWN have many examples where the lexical overlap is enough to determine the gold premise pair P_g. GPT3 does outperform the BM25 baseline, however, and in nearly every case, the SimCSE heuristic does as well (except for ENWN). SCSearch still outperforms all leading methods. There is a significant drop across all methods between ranking premise pairs with the immediate deduction and the goal. Although this was expected, the drop is quite significant and is worth exploring further in future work on how it could be mitigated.

5.2 Generating Proofs

We test the Additive Deduction Embedding heuristics on two proof generation datasets. These datasets are used to measure the end-to-end performance using Additive Deductions as a planning heuristic in search. The first is EntailmentBank T2 (Dalvi et al., 2021) and the other is ENWN (Sprague et al., 2022). Each example in these datasets contains a set of premises, P, and a claim g that we are trying to prove given P. To prove g, the system has to produce a series of deductions by combining two premises from the set P, then combining intermediate deductions and the premises in P until the claim is proven. Whether it is proven is determined via an entailment model scoring g above a certain threshold from some generated conclusion being above a certain threshold. Planning heuristics must determine which premise-premise or premise-deduction pairs are most likely to help in proving the claim.

In these datasets, the number of premises n is fairly small; $n < 30$ for most examples. There are usually only 3 to 5 deductions involved to produce the annotated entailment tree. For each dataset, we allow for a total of 10 steps ($maxStep$), and for each step, we allow for 5 generations to be sampled (k). We report the percentage of proofs that entailed the goal, g, as well as the average number of steps to prove the

claim across all planning heuristics in Table 5.3.

	Entailment Bank T2		ENWN	
	Completed	**Steps**	**Completed**	**Steps**
BM25	43%	2.2	48%	5.1
SCSearch	61%	3.4	86%	9.8
SimCSE (AD)	44%	2.8	46%	2.7
GPT3 (AD)	49%	2.2	41%	2.2

Table 5.3: Generated proofs per heuristic on the two datasets. BM25 has high performance on both of these datasets, indicating that textual overlap is enough to plan reasoning steps for nearly 50% of the examples. SimCSE (AD) and GPT3 (AD) underperforms BM25 on ENWN, this could mean that these methods are not as sensitive to lexical overlap as BM25 is. SCSearch still outperforms every baseline by as much as 38% showing that a lot of reasoning is unaccounted for in the other methods.

5.2.1 Results

GPT3 (AD) and SimCSE (AD) are able to produce slightly more proofs than BM25 on the EntailmentBank T2 dataset, but fail to outperform BM25 on ENWN. This result highlights how important it is for these heuristic planners to handle lexical overlap and plan deductions around how much the premises are lexically similar to it. However, SCSearch is able to generate more proofs than the other methods across both datasets by as much as 36%. This finding is also supported by the MRR results of the second intrinsic evaluation, shown in Table 5.2, where we saw SCSearch significantly outperforming other heuristics. SCSearch, appears to be capable of finding lexically similar premises for planning deductions, but is also finding deeper relations between premise pairs and the claim g when planning. Investigating what examples SCSearch can solve that the other methods cannot could indicate what these vector-based methods are not able to represent.

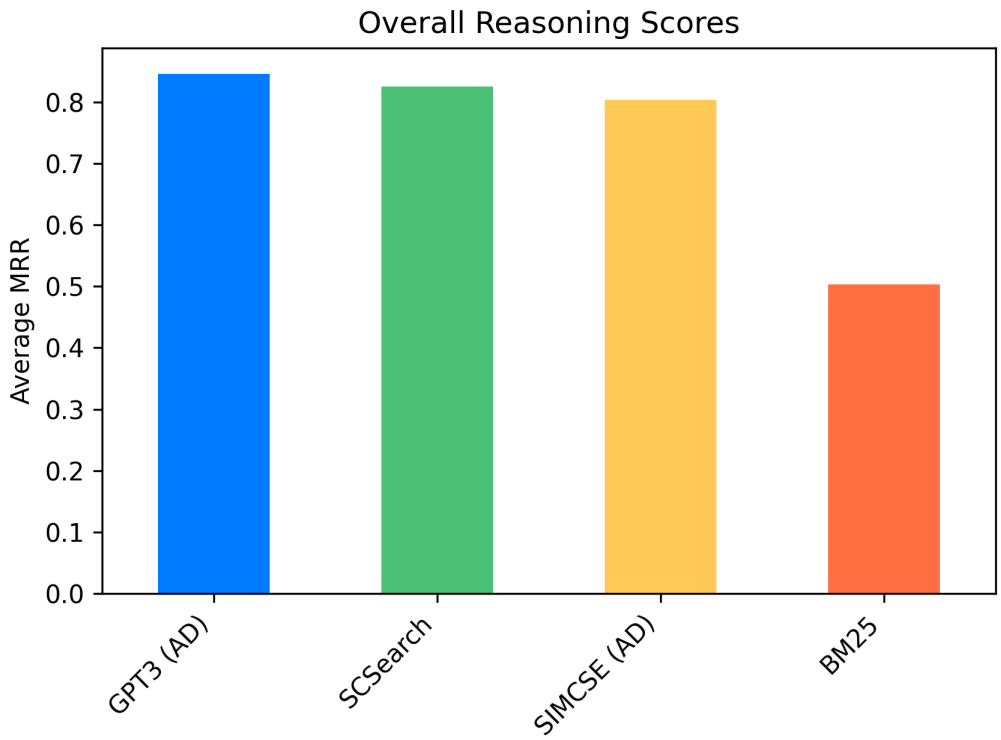

Figure 5.1: Overall scores of each heuristic on the SSRC dataset. GPT3 (AD) outperforms SCSearch slightly on this benchmark, differing from the intrinsic evaluations and end-to-end experiments. This implies that the types of reasoning involved in these datasets may be skewed towards a specific category.

5.3 Single-Step Reasoning Contrast Dataset

To best understand where the vector-based methods are lacking in performance and pinpoint where improvements can be made, we create the Single-Step Reasoning Contrast (SSRC) dataset and test each method across a variety of types of reasoning and common failure cases.

5.3.1 Results

We report the average MRR for each heuristic across all fourteen reasoning categories in Table 5.4 and the MRR across all perturbation types in Table 5.5. Table 5.4 shows that GPT3 (AD) can outperform both BM25 and SimCSE (AD) consistently across nearly every reasoning category and all perturbation types. Furthermore, we see that GPT3 (AD) is capable of beating or matching SCSearch on half of the reasoning categories and perturbation types. We find this result slightly surprising as it deviates from the findings from both the intrinsic evaluations and proof generation results. However, this might be related to GPT3 (AD) performing worse on the reasoning categories in SSRC that EntailmentBank T2 and Everyday Norms: Why Not dataset are sensitive to, meaning they have a high number of examples that can be solved using just lexical overlap. We have left classifying reasoning categories and perturbation types of these datasets as future work. Another possibility is that both datasets may have steps that use multiple categories of reasoning, whereas our dataset isolates and focuses on one category at a time. Distinguishing between a failure to handle complex reasoning categories or if EntailmentBank T2 and ENWN are skewed in certain reasoning categories is left for future work. However, BM25 does extremely well on the intrinsic evaluations and proof generation datasets and only performs on par with other methods in the Irrelevant Fact perturbation category. This isolated performance is more evidence that the reasoning in EntailmentBank T2 and ENWN is skewed toward lexical overlap.

We show average performance on the SSRC dataset in Figure 5.1. This result shows GPT3 (AD) outperforming SCSearch slightly overall, but because reasoning can be nuanced we think that the specific categories of reasoning and the specific perturbations are more telling of the performance. Furthermore, determining the reasoning categories in specific datasets would be needed to determine which categories are more important. We also compare SCSearch with all other heuristic methods, as SCSearch was the best in generating proofs, in Figure 5.2. This figure shows reason-

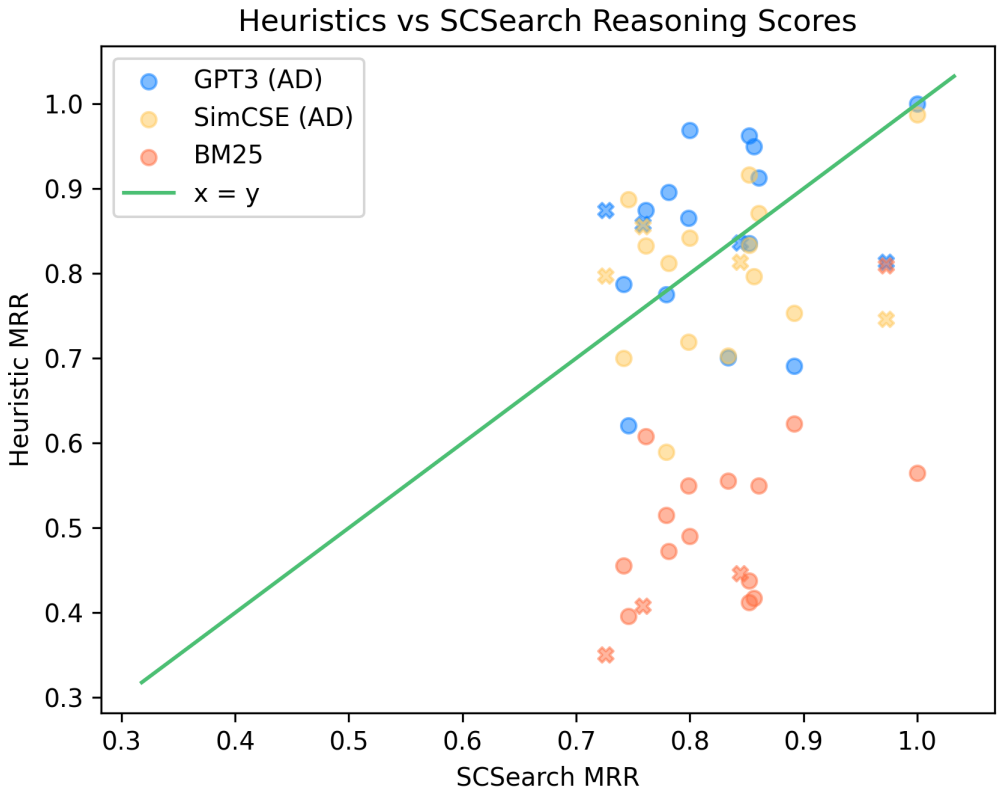

Figure 5.2: Comparison plot of the heuristic methods versus the SCSearch heuristic. If a point is above the green line, then that method outperformed SCSearch. Circles indicate reasoning categories and X-marks indicate perturbation types. BM25 underperforms all other methods, showing that the dataset is not sensitive to lexical overlap.

ing category scores as circles and perturbation type scores as X-marks. The green line X = Y indicates where SCSearch scored for that category or perturbation type: if a datapoint is above this line, it outperforms SCSearch. The figure highlights the information in Table 5.4 showing how BM25 consistently underperforms all methods, SimCSE does slightly better than BM25, and GPT3 (AD) is comparable to SCSearch.

	SCSearch	SimCSE (AD)	BM25	GPT3 (AD)
Analogy	0.86	0.80	0.42	**0.95**
Categorical syllogism	0.80	0.72	0.55	**0.87**
Causal Reasoning	**0.78**	0.59	0.52	**0.78**
Classification	0.86	0.87	0.55	**0.91**
Comparative Reasoning	0.85	0.92	0.44	**0.96**
Composition	0.75	**0.89**	0.40	0.62
Definition	0.80	0.84	0.49	**0.97**
Divisions	**0.85**	0.83	0.41	0.84
Modus Ponens	**1.0**	0.99	0.56	**1.0**
Modus Tollens	**0.83**	0.70	0.56	0.7
Propositional Logic	**0.89**	0.75	0.62	0.69
Quantification Logic	0.76	0.83	0.61	**0.87**
Spatial Reasoning	0.78	0.81	0.47	**0.90**
Temporal Reasoning	0.74	0.70	0.46	**0.79**

Table 5.4: Results of each heuristic on SSRC broken down by the reasoning category and averaged over the individual perturbation types of each category. We separate SCSearch and SimCSE (AD) from the BM25 and GPT3 (AD) heuristics as the BM25 and GPT3 (AD) have not been trained on any natural language inference data (with the possibility that GPT3 may have seen some incidental examples of inferences in its pretraining), making them close to zero-shot on this task. SCSearch and SimCSE (AD) have both been fine-tuned on reasoning datasets (EntailmentBank and NLI respectively).

	SCSearch	SimCSE (AD)	BM25	GPT3 (AD)
False Premise	**0.84**	0.81	0.45	0.81
Irrelevant Fact	**0.97**	0.75	0.81	0.87
Incorrect Quantification	0.76	**0.86**	0.41	**0.86**
Negated	0.73	0.80	0.35	**0.87**

Table 5.5: Results of each heuristic on SSRC broken down by the perturbation type and averaged over the individual reasoning categories. We again separate SCSearch and SimCSE (AD) from the BM25 and GPT3 (AD) heuristics.

Chapter 6: Discussion

Vector-based methods are not sufficient to capture all information for planning deductions. Our study revealed that vector-based methods, while promising in their simplicity, fall short in planning reasoning steps when compared to early-fusion premise rankers like SCSearch. This suggests that more complex, structured approaches may be necessary to effectively plan reasoning steps in step-by-step systems. Furthermore, we found that a majority of the success of these methods can be attributed to being sensitive to lexical overlap despite being able to represent more complex reasoning.

Datasets with more diverse reasoning types are needed. Not all claims can be proven by finding lexically similar evidence. However, a large percentage of the examples in the datasets like EntailmentBank used for proving claims can be solved by following that simple strategy. This can lead to ineffectual methods that do not generalize to more complex reasoning problems as we've shown in SSRC with BM25. To build better heuristics, we need better datasets that encompass various types of reasoning.

Understanding skew in our current datasets can improve task performance. A promising direction for future work is understanding the skew in existing datasets which can be facilitated by classifying the reasoning categories present in them. This classification will allow researchers to identify potential biases, gaps, and areas for improvement in the datasets, ultimately contributing to the development of more effective and comprehensive reasoning systems.

Chapter 7: Related work

Our work follows from models and methods done in the Question Answering domain where models are required to generate an answer or select evidence that lead to the answer through "multi-hop" reasoning (Chen et al., 2019; Min et al., 2019; Nishida et al., 2019). Although these end-to-end methods can be used in proof generation, understanding the underlying reasoning of the decisions being made is impactful for understanding the affordances of the model (Hase and Bansal, 2020; Bansal et al., 2021).

Step-by-step methods have been looked at for proof generation, detangling planning and reasoning into separate subsystems that work together as a whole when proving a claim (Creswell et al., 2023; Bostrom et al., 2022; Dalvi et al., 2021; Ribeiro et al., 2022). Our work extends from this literature, focusing on exploring alternative methods for natural language deduction planning entirely in embedding space by tapping into the property of additive deductions.

We also follow work being done in retrieval, which focuses on finding evidence from a large corpus that would help answer a query. State-of-the-art retrieval methods involve encoding the corpus into vector indexes that can be used to calculate the cosine similarity of an encoded query (Xiong et al., 2021; Karpukhin et al., 2020; Khattab and Zaharia, 2020). Sparse encoders, like BM25, have also been used to help reduce the search space for relevant passages (Valentino et al., 2022). However, none of the methods tap into the additive deduction property in their embedding spaces and instead encode the query to find relevant passages and then re-encode the query with the appended passages to find additional relevant passages. We consider this to be similar to early-fusion premise rankers in the proof generation task.

Another line of relevant work deals with understanding reasoning errors from language models, like the detection of logical fallacies in text (Jin et al., 2022), toxicity

and biases in reasoning (Zhou et al., 2023), and how priors can lead to common human-like errors (Lin et al., 2022). We further this line of work with our new Single-Step Reasoning Contrast dataset which classifies examples by the type of reasoning being done as well as the type of perturbed premises that are being favored by the model rather than categorizing our examples by textual topics.

Chapter 8: Conclusion

In this study we have looked at the Additive Deduction property in different embedding models as a method for planning in proof generation. While Additive Deduction can outperform sparse methods, it still falls short when compared to early-fusion premise rankers like SCSearch. To further investigate why early fusion premise rankers outperform other methods, we introduced the Single-Step Reasoning Contrast dataset, which is intended to score a models ability to reason about natural language deductions using various categories of logical reasoning among common reasoning errors. We've found that methods using the Additive Deduction property can encompass different reasoning categories, similar to early-fusion premise rankers, but are less sensitive to lexical overlap which hinders performance on proof generation datasets. We believe that future work should explore the combination of categories into more intricate examples, the classification of reasoning categories in current datasets, and the assessment of each category's impact on downstream datasets. In doing so, we believe more robust datasets can be created that test a variety of different types of reasoning and thus create more generalized proof generation systems.

Works Cited

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. Does the Whole Exceed Its Parts? The Effect of AI Explanations on Complementary Team Performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380966. URL https://doi.org/10.1145/3411764.3445717.

Kaj Bostrom, Xinyu Zhao, Swarat Chaudhuri, and Greg Durrett. Flexible generation of natural language deductions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6266–6278, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.506. URL https://aclanthology.org/2021.emnlp-main.506.

Kaj Bostrom, Zayne Sprague, Swarat Chaudhuri, and Greg Durrett. Natural language deduction through search over statement compositions. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4871–4883, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-emnlp.358.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle,

M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Jifan Chen, Shih-ting Lin, and Greg Durrett. Multi-hop question answering via reasoning chains. *arXiv*, abs/1910.02610, 2019. URL https://arxiv.org/abs/1910.02610.

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3Pf3Wg6o-A4.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.585. URL https://aclanthology.org/2021.emnlp-main.585.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.552. URL https://aclanthology.org/2021.emnlp-main.552.

Peter Hase and Mohit Bansal. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the 58th Annual*

Meeting of the Association for Computational Linguistics, pages 5540–5552, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.491. URL https://aclanthology.org/2020.acl-main.491.

Zhijing Jin, Abhinav Lalwani, Tejas Vaidhya, Xiaoyu Shen, Yiwen Ding, Zhiheng Lyu, Mrinmaya Sachan, Rada Mihalcea, and Bernhard Schoelkopf. Logical fallacy detection. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7180–7198, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-emnlp.532.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/

2022.acl-long.229.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1613. URL https://aclanthology.org/P19-1613.

Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. Answering while summarizing: Multi-task learning for multi-hop QA with evidence extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2335–2345, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1225. URL https://aclanthology.org/P19-1225.

OpenAI. Gpt-4 technical report, 2023.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Henghui Zhu, Rui Dong, Xinchi Chen, Zhu Peng, Zhiheng Huang, Andrew Arnold, and Dan Roth. Entailment tree explanations via iterative retrieval-generation reasoner. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online and Seattle, USA, July 2022. Association for Computational Linguistics. URL https://arxiv.org/abs/2205.09224.

Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. Gaithersburg, MD: NIST, January 1995. URL https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/.

Zayne Sprague, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Natural language deduction with incomplete information. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8230–8258, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.564.

Marco Valentino, Mokanarangan Thayaparan, Deborah Ferreira, and André Freitas. Hybrid autoregressive inference for scalable multi-hop explanation regeneration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11403–11411, 2022.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=EMHoBG0avc1.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=92gvk82DE-.