Math-DB: A Discourse Framework for Mathematical Word Problems to Enhance LLM Reasoning

Mustafa Erolcan Er
Department of Cognitive Science
Middle East Technical University (METU)
Ankara, Türkiye
erolcan@metu.edu.tr

Abstract

Large Language Models have demonstrated significant progress in solving mathematical word problems through techniques like Chain-of-Thought (CoT) prompting. However, recent research indicates that these models often rely on statistical regularities and surface-level patterns rather than true logical reasoning, leading to performance drops when faced with minor problem perturbations or irrelevant information. In this study, we introduce Math Discourse Bank (Math-DB), a novel discourse framework and annotated dataset designed to enhance LLM reasoning. Inspired by the Penn Discourse TreeBank (PDTB) and mathematics education research, Math-DB defines a hierarchy of discourse senses designed for quantitative reasoning, including categories such as Change, Combine, Compare, and Equalize. We applied this framework to the GSM-Symbolic dataset of 12,500 problems, yielding 47,815 sense-labeled discourse relations over 11,414 successfully-aligned instances (91.3% pipeline yield). Our experiments demonstrate that incorporating Math-DB annotations into CoT prompts consistently improves LLM performance across various difficulty levels.

1 Introduction

Large Language Models have made significant improvement in solving mathematical word problems, especially with techniques like CoT prompting (Wei et al., 2022). In particular, prompting LLMs to generate intermediate reasoning steps has enabled remarkable performance on benchmarks such as GSM8K (Cobbe et al., 2021). Despite these advances, a growing body of evidence indicates that current LLMs still fall short of true logical reasoning. By *true logical reasoning*, we mean the ability to derive a correct solution from the underlying semantic structure of a problem in a way that is invariant to surface-level changes such as renaming entities, substituting numerical values, or inserting irrelevant context. Empirically, this is distinguished from template- or pattern-based behavior by testing whether model accuracy remains stable under such controlled perturbations. Current LLMs instead often detect statistical regularities and template patterns present in the training data. This can lead to memorized behavior, where small changes in a problem's presentation can cause drastic drops in LLM's performance (Mirzadeh et al., 2024).

Mirzadeh et al. (2024) generated diverse math problems from symbolic templates to test whether models truly understand the reasoning or just the surface form. They found that even minor perturbations, such as modifying numerical values or proper nouns, noticeably degrade LLM performance. Furthermore, inserting irrelevant information (distractor clauses that do not affect the solution) can confuse models into drastic errors. This suggests that models do not effectively reason about math word problems but rather memorize the answers already found in training data. Similar findings by Shi et al. (2023) were obtained on GSM-IC, a variant of GSM8K in which each problem is augmented with a sentence containing irrelevant but topically related information; the addition of such distractor context caused dramatic performance decreases.

One promising direction is to provide semantic guidance for the LLM's reasoning. Humans solving word problems rely on the narrative relations between quantities (e.g. identifying that one quantity increases another, or that two parts combine into a whole) rather than just crunching numbers in order. Discourse frameworks like PDTB (Prasad et al., 2017) capture how clauses connect logically (causal, temporal, contrastive), but general-purpose relations such as *Contingency.Cause* do not directly map to the operations in math problems. Mathematics education research, by contrast, has long categorized arithmetic word problems into schema types based on the semantic relations between quantities (Daroczy et al., 2015; Riley et al., 1984). We

Proceedings of the 20th Linguistic Annotation Workshop (LAW XX), pages 75–94

adopt and adapt these categories in Math-DB (Section 3), hypothesizing that labeling the semantic relations between clauses can guide the model to focus on what each step means rather than relying on positional heuristics or keywords alone.

In this paper, we present Math-DB, a novel discourse framework and annotated dataset for mathematical word problems. Math-DB defines a PDTB-style hierarchy of discourse senses customized for quantitative reasoning. Each pair of adjacent clauses in a math problem is assigned a relation sense that describes its math-relevant semantic connection. A sample annotated problem can be seen in Figure 1 below.

We applied Math-DB to the full GSM-Symbolic benchmark of Mirzadeh et al. (2024), comprising 5,000 main problems, 5,000 GSM-P1 problems, and 2,500 GSM-P2 problems, for a total of 12,500 source instances. GSM-P1 problems are generated by adding one additional clause to each original problem, while GSM-P2 problems add two such clauses. Each added clause introduces one extra reasoning step, forcing the model to perform additional calculations beyond the standard baseline question. Our semi-automatic annotation pipeline successfully aligned 11,414 of these 12,500 instances (4,621 main, 4,535 P1, and 2,258 P2; 91.3% overall yield), producing 47,815 sense-labeled discourse relations. The remaining 1,086 instances were rejected at the alignment stage and are documented in the released diagnostic logs (see §4). To create Math-DB, we built a semi-automatic pipeline: a fine-tuned BERT-based parser identifies discourse connectives and segments each problem into PDTB-style arguments (Arg-1, Arg-2), after which six trained undergraduate annotators from a mathematics department assign the relation sense to each link.

Finally, we show how Math-DB can enhance Chain-of-Thought prompts: alongside the problem text, the model receives discourse annotations that flag the operation each clause introduces (e.g. aggregating parts vs. subtracting a removed amount). Our results (Section 5.2) show that this guidance makes the model more robust to the added-complexity perturbations in GSM-P1 and GSM-P2.

In summary, our contributions are:

- A domain-specific discourse relation hierarchy for math word problems, grounded in known semantic problem types (Change, Combine, Compare, Equalize, etc.), presented in Section 3.

- An annotated corpus consisting of 47,815 Math-DB discourse relations over 11,414 successfully-aligned instances from the 12,500-problem GSM-Symbolic benchmark, together with diagnostic logs covering the 1,086 unaligned instances, described in Section 4. This new corpus will be released publicly, providing a resource to train and evaluate models that can understand and predict mathematical discourse structure.

- A discourse-augmented prompting experiment that elaborates classical CoT prompts with Math-DB annotations for LLMs, presented in Section 5.

2 Related Work

2.1 Discourse Frameworks

Clause connectivity is a fundamental focus in NLP, exemplified by the Penn Discourse Treebank (PDTB) (Prasad et al., 2017). The PDTB framework utilizes a lexically grounded, three-level hierarchy to map logical and pragmatic links between abstract objects (Miltsakaki et al., 2008; Webber et al., 2019). The hierarchy proceeds from broad classes at Level-1 (e.g., *Temporal*, *Contingency*, *Comparison*, *Expansion*) to finer types at Level-2 (e.g., *Temporal.Asynchronous*) and to specific subtypes at Level-3 (e.g., *Temporal.Asynchronous.Precedence*, as in "She finished her homework *before* dinner"). Math-DB adapts this paradigm to the mathematical domain, redefining top-level categories to align with cognitive situation schemas: Change, Combine, Compare, and Equalize. These schemas are drawn from a long tradition in mathematics education research showing that elementary arithmetic word problems cluster into a small number of recurring semantic structures, and that the structure of a problem (rather than its surface wording or arithmetic difficulty) is a primary determinant of how children represent and solve it (Carpenter and Moser, 1979; Riley et al., 1984; Fuson, 2012; Greeno, 1978; Nesher, 2020; Vergnaud, 2020). We adopt these schemas as Math-DB's Level-1 categories because they provide a cognitively grounded inventory that maps directly onto the arithmetic operations needed to solve a problem.

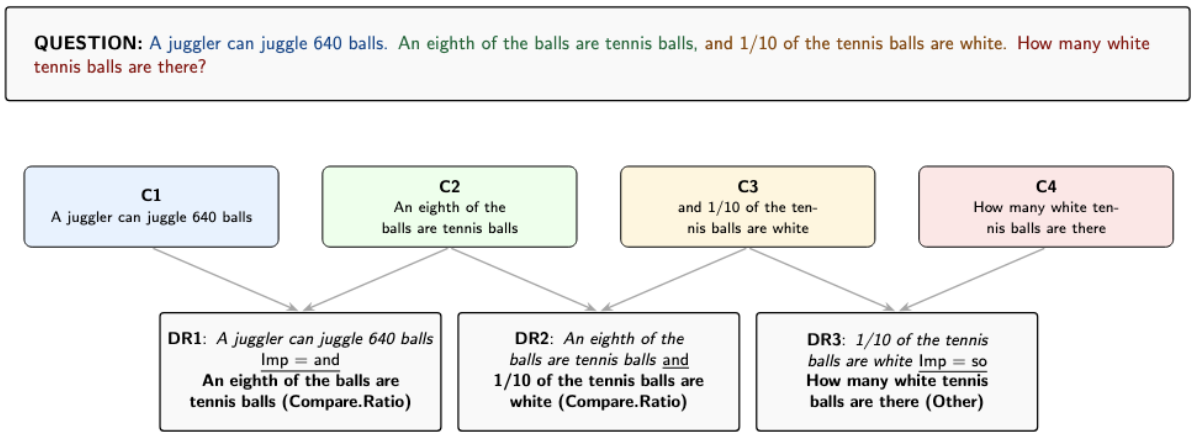

Figure 1: A schematic annotation example from Math-DB. **C** denotes a clause (component) extracted from the problem text; **DR** denotes a discourse relation between two adjacent clauses, labeled with a Math-DB Level-1.Level-2 sense; **Imp** indicates that the relation is implicit (no overt discourse connective is present in the text).

2.2 Math Word Problem Solving

Solving math word problems has been a long-standing challenge in AI, characterized by distinct paradigm shifts over the years. Early studies were predominantly rule-based, relying on manually crafted logic and pattern matching to map natural language directly to mathematical expressions (Bobrow, 1964; Fletcher, 1985). Subsequently, the field shifted toward machine learning-based approaches, which treated problem texts as templates to be matched to known equation forms through statistical mapping rather than hand-coded rules (Hosseini et al., 2014; Kushman et al., 2014; Roy and Roth, 2018; Shi et al., 2015).

2.3 LLM Reasoning and Prompting Strategies

LLMs with Chain-of-Thought (CoT) prompting offer a new way to tackle word problems, shifting from specialized pre-trained architectures (Liang et al., 2021; Shen et al., 2021) to in-context learning. CoT prompting enabled models like GPT-3/PaLM to reach state-of-the-art on GSM8K (Wei et al., 2022; Kojima et al., 2022), spurring strategies such as Least-to-Most (Zhou et al., 2022), self-consistency (Wang et al., 2022), and Tree-of-Thoughts (Yao et al., 2023).

However, LLM reasoning remains limited, particularly regarding input sensitivity. Chen et al. (2024) found that reordering premises without changing content causes significant accuracy swings. This implies LLMs lack internal mechanisms to organize information independently, relying instead on the provided narrative flow.

2.4 Discourse Structure for Reasoning

Recently, discourse structure has been explored as a catalyst for reasoning. Sharma et al. (2025) showed that providing general discourse context boosts GSM8K performance. Our work differs in three ways. First, while Sharma et al. (2025) use the tree-based Rhetorical Structure Theory (RST) (Mann and Thompson, 1987), Math-DB adopts the lexically-grounded PDTB paradigm (Prasad et al., 2017) for more granular analysis. Second, while Sharma et al. (2025) use general relations, Math-DB introduces a domain-specific hierarchy (e.g., Change, Combine, Compare) engineered for quantitative reasoning. Finally, unlike their synthetic annotations from a 70B model (Dubey et al., 2024), Math-DB is human-annotated and expert-verified.

In summary, literature highlights the need for structure in reliable reasoning. Our work builds on discourse analysis and LLM prompting to develop models that produce grounded, generalizable, and trustworthy reasoning.

3 Math-DB Framework

Math-DB provides discourse relation senses for math word problems in a two-level hierarchy (Level-1 categories, each with Level-2 subtypes). Each relation describes how Arg-2 relates to Arg-1 in a way that informs quantitative reasoning; labeling each adjacent clause pair yields a structured recipe of the reasoning steps.

The five Level-1 categories (Change, Combine, Compare, Equalize, and Other) were directly adopted from the canonical schema typology established in the mathematics education literature on

additive and subtractive word problems (Riley et al., 1984; Carpenter and Moser, 1979; Fuson, 2012). The Level-2 subtypes were derived through iterative analysis of GSM8K-style problems during a pilot annotation phase: annotators labeled a shared set of 50 problems, and recurring patterns of ambiguity and distinct semantic operations within each top-level category (e.g., additive vs. multiplicative composition under Combine) were promoted to formal subtypes. The resulting two-level hierarchy was then frozen prior to the main annotation effort.

3.1 Level-1 Relation Categories

We define five top-level relation categories in the Math-DB, which correspond to high-level situation schemas common in math narratives:

Change: Relations where Arg-2 represents an update to a quantity mentioned in Arg-1, usually through some event over time. These are the classic "change" situations (increase or decrease of some amount). For example, "Alice had 10 apples. Then she bought 5 more.", the second sentence changes Alice's apple count from the first sentence. Change relations involve the same quantity evolving, as opposed to combining different entities or comparing separate entities.

Combine: Relations where Arg-2, together with Arg-1, contributes to forming a whole. This often corresponds to part-whole addition or to the components of a multiplication scenario. Combine captures the idea of composition.

Compare: Relations where Arg-2 introduces a comparison between two quantities (often one from Arg-1 and another from Arg-2) via either a difference or a ratio. Unlike Combine, a comparison does not merge the quantities into a single whole, but relates them to each other (how much more/less? how many times larger/smaller?).

Equalize: Relations where Arg-2 is about making two quantities equal (or examining what it takes to make them equal). This often appears in problems asking "how many more… to have the same …" or scaling one quantity to match another. Equalize situations imply a goal of matching values, either by adding/removing a difference or by scaling via a ratio.

Other: A catch-all category for relations that do not directly contribute to the quantitative solution by introducing a new operation. We use Other in two principal cases. First, when Arg-2 provides background or contextual detail about Arg-1 that sets the scene or adds story flavor (e.g. "It was John's birthday." preceding the actual math events) without introducing a new equation. Second, and equally important, when Arg-2 is the final question clause of the problem (e.g. "How many white tennis balls are there?" in Figure 1). The question clause is highly salient for problem solving, since it identifies the target quantity to be computed, but it does not itself introduce a new arithmetic operation between quantities. By labeling it as Other, we distinguish it from operation-bearing relations (Change, Combine, Compare, Equalize) while still flagging it as a structurally distinct unit in the discourse.

3.2 Level-2 Relation Subtypes

Each top-level category (except Other, which has no subtypes) is further refined into Level-2 senses capturing the core semantic operation or relation type. We list them below:

Change.Increase: Arg-2 increases the quantity from Arg-1. This corresponds to events like gaining, adding, receiving, or any scenario where the net effect is an increase in the amount. For example: "She had 5 apples. She bought 3 more." would be Change.Increase (the 3 more increases the count).

Change.Decrease: Arg-2 decreases the quantity from Arg-1. Events like losing, spending, giving away, or any removal cause a decrease. E.g.: "He had 12 tickets. He gave 4 to his friend." is Change.Decrease (4 is subtracted from 12).

Both increase and decrease are about the same item/count evolving. Often, these imply an equation $\text{Arg-1}_{\text{quantity}} \pm \text{change} = \text{new}_{\text{quantity}}$. If explicit, `Arg-2` may contain words like "more" or "left" or "remaining" that hint at the change.

Combine.PartWhole: Arg-1 and Arg-2 describe parts and/or a whole in an additive part-whole structure. Typically, either:

Arg-1 and Arg-2 are two parts contributing to a total, or Arg-1 is a whole and Arg-2 gives a missing part.

For instance: "There are 7 red marbles. There are 5 blue marbles.", Arg-2 (blue marbles) is another part to combine with Arg-1 (red marbles) into a total (implicitly 12 marbles). If later a question asks for total marbles, the relation between the two sentences is Combine.PartWhole. This subtype covers classic "aggregation" statements ("in total", "altogether" often signal this).

Combine.Product: Arg-1 and Arg-2 together set up a multiplicative situation, like two factors or a structured array. A common scenario is one

sentence gives a number of groups/items per group, and the other gives another dimension (number of groups), so the total is a product. Example: "Each box has 6 eggs. There are 4 boxes.", Arg-2 relates to Arg-1 as Combine.Product.

Compare.Difference: Arg-2 introduces an additive comparison, i.e. a difference. Typically phrased as "how many more/less" or stating one is N more than the other. For example: "Alice has 8 candies. Bob has 5 candies." If followed by "How many more does Alice have than Bob?", the relation between the two statements is Compare.Difference (Alice's count vs Bob's count).

Compare.Ratio: Arg-2 introduces a multiplicative comparison or a rate. Keywords include "times as many", "twice/half", or per-unit rates ("per", "each"). For instance: "Tom has 3 marbles. Jerry has twice as many marbles as Tom.", the second sentence is Compare.Ratio (Jerry's count is a multiple of Tom's).

Equalize.MatchDifference: Arg-2 frames a difference between two quantities as the adjustment needed to reach equality. Compare this to `Compare.Difference`: "Alice has 8 candies. Bob has 5. How many more does Alice have?" (the answer is a measurement of the gap) versus "Alice has 8 candies. Bob has 5. How many more does Bob need to have the same as Alice?" (the answer is the same number, 3, but interpreted as the action required to equalize). The arithmetic is identical; the discourse intent is not.

Equalize.MatchRatio: Arg-2 is about making quantities equal by scaling one of them, a multiplicative adjustment. E.g.: "John has 4 apples, which is twice as many as Jim. How many apples does Jim need to have the same as John?" Here, one could interpret it as Jim's amount must be scaled up to John's. [1]

[1]Both Compare.Difference and Equalize.MatchDifference involve subtraction between two quantities, and similarly both Compare.Ratio and `Equalize.MatchRatio` involve a multiplicative relationship. The distinction is not arithmetic but pragmatic: `Compare` relations *report* the gap or ratio between two independent quantities that remain distinct ("How many more does Alice have than Bob?"), whereas `Equalize` relations *frame the gap or ratio as an adjustment* required to reach a target state of equality ("How many more does Bob need to have the same as Alice?"). In other words, `Compare` stops at measurement, while `Equalize` prescribes an action toward equality. This pragmatic distinction matters for LLM guidance because the implied solution step differs: a `Compare` prompts a difference computation as the final answer, whereas an `Equalize` prompts the same computation but reinterprets it as "the amount to add to the smaller quantity."

4 Annotation of GSM-Symbolic with Math-DB

We applied the Math-DB framework to the GSM-Symbolic benchmark, which comprises 12,500 problem instances derived from 100 GSM8K source templates. Of these, 11,414 instances passed our semi-automatic pipeline and received Math-DB sense labels, yielding 47,815 discourse relations; the remaining 1,086 instances are documented in the released diagnostic logs (see §4.4). The full annotated corpus, with all 47,815 discourse relations and accompanying metadata, is released publicly.[2] This section describes the dataset, our annotation procedure, corpus statistics, and pipeline-yield analysis.

4.1 Data

GSM-Symbolic was introduced by Mirzadeh et al. (2024) as an improved benchmark for mathematical reasoning. It was implemented by creating symbolic templates from GSM8K problems and then generating multiple instantiations by varying numbers, names, and other surface details. The goal was to test LLMs on the same underlying problems but with different wordings, to see if they truly understood the logic or just memorized specific wordings. The dataset is divided into:

Symbolic (Main): 5,000 problems that mirror original GSM8K questions in structure (50 instantiations per template), of which 4,621 were successfully annotated by our pipeline.

GSM-P1 (Symbolic-Plus-One): 5,000 problems where an additional clause has been inserted into each problem to increase complexity by one extra sentence, of which 4,535 were successfully annotated.

GSM-P2 (Symbolic-Plus-Two): 2,500 problems where two clauses were inserted, making the problems longer to confuse a solver, of which 2,258 were successfully annotated.

4.2 Annotation Procedure

Before describing the pipeline, we note an important property of GSM-Symbolic that shaped our annotation effort. Because instances within a given template differ only in surface details (numerical values, proper names, units) and not in semantic structure, all instances of a single template share the same underlying discourse structure. We

[2]https://github.com/erolcan-er/Math-DB

therefore performed template-level (id-based) annotation for the main GSM-Symbolic subset: one canonical sense sequence was annotated per template and propagated to instances of that template for which our Stage-1 pipeline could re-apply the canonical clause boundaries to the variant surface form. Instances for which canonical alignment failed (i.e., the detected number of discourse relations did not match the template) were excluded and are documented in the diagnostic logs (§4.4). This reduced the annotation effort on the main subset from approximately 21k relations to a much smaller set of unique templates.

However, for GSM-P1 and GSM-P2, we adopted an instance-level annotation strategy. Although surface variation within a template is still semantic-preserving, the inserted clauses in P1 and P2 are themselves generated from a template, so different instances of the same id can include structurally different inserted clauses (e.g., a distractor in one instance versus a real additional reasoning step in another). We therefore annotated each P1/P2 instance independently to capture this variation; in practice, different instances of the same P1/P2 id can differ in both the number and the sense of their discourse relations. The corpus statistics in Table 1 are reported over all 47,815 final relations across all instances (main, P1, and P2) rather than over unique templates.

We used a two-stage annotation strategy: automatic pre-processing followed by manual sense labeling.

4.2.1 Stage 1: Connective Detection and Argument Segmentation

We fine-tuned BERT large (Devlin et al., 2019) using a sub dataset of GSM8K labeled by BIO tags (following DISRPT shared task formulations (Zeldes et al., 2021; Braud et al., 2023)) to identify explicit discourse connectives in the text (words like "then", "so", "after", "because", etc.) and to segment sentences into two arguments (Arg-1 and Arg-2) around each connective. If two arguments are detected without explicit discourse connectives, we accept these arguments as an implicit discourse relation[3].

This step provided discourse relations consist of Arg-1 and Arg-2 spans together with optional discourse connective spans for explicit relations.

[3]See Appendix A for detailed model specifications and training hyperparameters.

4.2.2 Stage 2: Sense Labeling by Annotators

For the Stage 2, we recruited six undergraduate students as annotators from the mathematics department (fluent in English and with strong math aptitude). Each discourse relations were assigned to three annotators independently for sense labeling. Annotators were provided with the Math-DB annotation manual (as described in Appendix B, with examples and decision criteria). For each discourse relation, they choose the appropriate Level-1 and Level-2 sense that best describes how Arg-2 relates to Arg-1 in the context of solving the problem. We allowed annotators to see the full problem text (all sentences) so they had context, but instructed them to label relations in a forward sequential manner, treating each adjacent link in turn.

To ensure quality, we conducted an initial training phase where all annotators labeled a common set of 50 problems, then discussed disagreements with an adjudicator (the expert author). This helped clarify guidelines (e.g., distinguishing Combine vs. Change when an addition happens over time: if a time/event is involved, it is Change, but if it is just combining two existing quantities, it is Combine). After refining the instructions, the annotators proceeded with the full dataset.

Each relation was assigned to three of the six annotators (sampled to balance load), so that every relation ultimately received three independent labels. We aggregated them by majority vote: if at least two annotators agreed on the same sense, that was assigned as the final label.

For cases where all three annotators gave different labels or one label was not in majority (this happened for 1,342 relations, 2.8% of the total), we invoked the expert adjudication. The expert (a computational linguist with a strong math background, also an author) reviewed those cases and chose the final sense label among the proposed ones. After adjudication, every relation in the dataset had a final agreed-upon label.

4.3 Corpus Statistics and Analysis

Our Math-DB annotated corpus comprises 47,815 discourse relations across 11,414 successfully-aligned instances (out of 12,500 source problems; the remaining 1,086 are analyzed in §4.4).

Explicit vs Implicit: Of the 47,815 relations, 15,011 (≈31.4%) are explicit discourse relations. The remaining 32,804 (≈68.6%) are implicit discourse relations, meaning no connective is overtly

present and the relation is assumed between each adjacent clause. This implicit majority is expected, since many math problem sentences follow one another without "so/then/because" even though a logical link exists. It underscores the need for reasoning beyond surface cues.

Relation Sense	Count	%
Change	**14,891**	**31.1%**
Change.Increase	8,412	17.6%
Change.Decrease	6,479	13.5%
Combine	**7,484**	**15.7%**
Combine.PartWhole	5,401	11.3%
Combine.Product	2,083	4.4%
Compare	**1,498**	**3.1%**
Compare.Ratio	1,020	2.1%
Compare.Difference	478	1.0%
Equalize	**504**	**1.1%**
Equalize.MatchDifference	430	0.9%
Equalize.MatchRatio	74	0.2%
Other	**23,438**	**49.0%**
Total	**47,815**	**100.0%**

Table 1: Distribution of Math-DB discourse relations across Level-1 categories and Level-2 subtypes, computed over 47,815 relations from 11,414 successfully-annotated instances.

Distribution of Level-1 senses: Table 1 reports the full sense distribution. The single largest category is *Other* at 49.0%, reflecting that nearly half of all adjacent clause pairs are either question clauses or narrative-context links that do not introduce a new arithmetic operation. Among the operation-bearing categories, *Change* is the most frequent (31.1%), followed by *Combine* (15.7%); together they account for roughly 47% of all relations, consistent with the fact that most problems involve accumulating or modifying a running total (Change) and aggregating pieces (Combine). *Compare* (3.1%) and *Equalize* (1.1%) are substantially less frequent, which aligns with the GSM-style problem inventory: comparison and equalization scenarios appear but are not the dominant problem types.

Level-2 senses: Within Change, increases (8,412) outnumber decreases (6,479), reflecting that problem narratives more often involve gaining (earning, buying, receiving) before possibly losing or spending later. Within Combine, PartWhole (5,401) is roughly 2.6× more frequent than Product (2,083), indicating that pure additive composition is more common than explicit multiplicative composition. For Compare, Ratio (1,020) outnumbers Difference (478) by roughly 2×, which is consistent with the prevalence of unit-rate and "times as many" constructions in GSM-style problems. For Equalize, MatchDifference (430) is far more common than MatchRatio (74), aligning with typical curricula in which "how many more to have same" scenarios appear more often than scale-equalize scenarios.

We also tracked the occurrence of connectives. Some explicit connectives we saw frequently: temporal connectives like "then", "after that" signaling Change relations, additive connectives like "in total" or "altogether" signaling Combine.PartWhole, contrastive phrases like "more than" or "less than" indicating Compare.Difference, phrases like "times as many" for Compare.Ratio, and equality phrases like "the same as" for Equalize. We note that our connective inventory is open-ended: the span-based Stage-1 detector occasionally identifies multi-token linking phrases (e.g., "On top of that,", "each having") as connectives. We retain these as found rather than collapsing them to a closed PDTB-style lexicon, since they often carry semantic content relevant to the relation.

Inter-annotator agreement before adjudication was substantial: Fleiss' $\kappa = 0.92$ on Level-1 senses and $\kappa = 0.84$ on Level-2 senses (full IAA details and confusion analysis in Appendix D).

4.4 Pipeline Yield and Failure Modes

Of the 12,500 GSM-Symbolic instances we set out to annotate, our semi-automatic pipeline successfully produced Math-DB labels for 11,414 instances (91.3%). The remaining 1,086 instances were rejected by the alignment stage with the diagnostic reason `dr_count_mismatch`: the number of discourse relations detected in the variant did not match the canonical template's expected count, indicating that the variant's surface form deviated from the template structure in ways our segmenter could not reconcile. Table 2 reports the per-subset yield.

Subset	Total	Annotated	Yield
Symbolic (Main)	5,000	4,621	92.4%
GSM-P1	5,000	4,535	90.7%
GSM-P2	2,500	2,258	90.3%
Total	**12,500**	**11,414**	**91.3%**

Table 2: Per-subset pipeline yield for Math-DB annotation. Rejected instances are documented in the released diagnostic logs.

We release the full diagnostic logs (instance identifiers, detected vs. expected relation counts, and failure reasons) alongside the annotated corpus. These logs serve a dual purpose: they make pipeline coverage transparent to downstream users, and they provide a targeted benchmark for future work on robust automatic discourse segmentation of mathematical text.

5 Using Discourse Annotations to Enhance CoT Prompting

In this section, we leverage our annotations to improve LLM reasoning. Our approach operates in the context of CoT prompting. The key idea is to supply the LLM with Math-DB guided hints about what each step of the reasoning should do, in addition to the problem text itself.

5.1 Discourse-Augmented Prompt Design

All experiments in this section use GPT-4o-mini as the underlying LLM, chosen to match the model evaluated in the original GSM-Symbolic study (Mirzadeh et al., 2024) for direct comparability. We discuss the implications of this single-model setup, and the expected behavior across other LLMs and prompting strategies, in the Limitations section.

A straightforward way to include discourse information is to present it as an annotated version of the problem. We designed a prompting format that uses a structured format with discourse sense information for each targeted problem.

For each targeted problem, we present a brief instruction for Math-DB annotation scheme. Then, we append 8-shot demonstrations with Math-DB discourse relations and step-by-step answer explanations. In the inference phase, discourse relations found in the targeted problem are provided. Finally, we ask the model to solve the given problem using all provided information. Simplified prompt template can be seen in Figure 2 [4]:

For our baseline, we follow the standard evaluation protocol defined in the Mirzadeh et al. (2024). The baseline uses an 8-shot CoT prompt using the original GSM8K demonstrations. The prompt template consists of a general system instruction followed by eight fixed question-answer pairs that provide only the problem text and the step-by-step reasoning chain, without any discourse-level annotations. All experiments were conducted using greedy decoding to ensure consistency. The full prompt template for the baseline approach is provided in Appendix C.

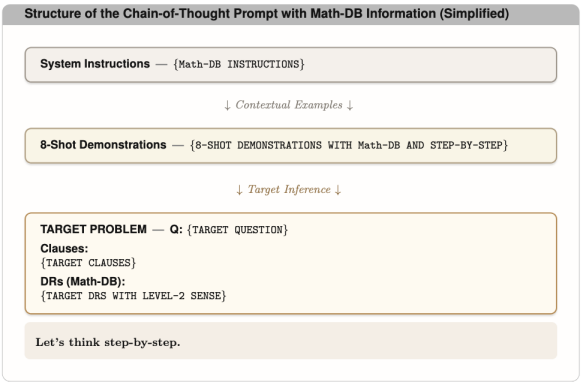

Figure 2: Simplified prompt template

In our evaluation, the discourse relations provided to the model during the inference phase are derived from the gold-standard annotations in Math-DB. We intentionally adopt this "Oracle" experimental setup to isolate the impact of the discourse framework on mathematical reasoning from the potential noise and error propagation of an automated parser. By using ground-truth relations, we establish the upper-bound performance gain achievable when the underlying semantic structure of a problem is explicitly represented for the LLM.

5.2 Results

Our discourse-augmented prompting consistently outperforms the standard CoT prompting on all subsets of GSM-Symbolic. We provide a Figure 3 of accuracies below.

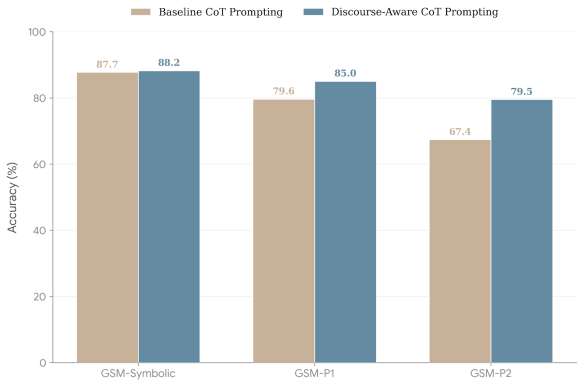

Figure 3: Comparison of accuracy between Baseline CoT Prompting and Discourse-Aware CoT Prompting across three GSM-based datasets.

On the GSM-Symbolic dataset, GPT-4o-mini with standard CoT achieved an accuracy of 87.7%,

[4]The full prompt template is provided in Appendix C.

whereas with Math-DB prompting it achieved 88.2%.

On GSM-P1, the baseline performance dropped to 79.6% (highlighting the brittleness when an extra irrelevant clause is present, similar to observations from Mirzadeh et al. (2024)). In contrast, our method achieved 85% on P1, which is substantially higher, and notably close to the Symbolic dataset performance. The relative drop from Symbolic → P1 was much smaller with our approach compared to the baseline CoT prompting strategy.

On GSM-P2, the trend is similar. Baseline CoT struggled even more with two distractors, achieving 67.4%, whereas our discourse-guided approach obtained 79.5%. While GSM-P2 is hardest for both, the gap between our method and baseline widened further here, showing that the more complex the input, the more the structured guidance helps.

5.3 Error Analysis

We performed a qualitative analysis of cases where the baseline CoT failed but the discourse-augmented CoT succeeded, and vice versa. [5]

Baseline CoT often failed on P1/P2 datasets with following reasons:

- getting distracted by a fake number or event (e.g., trying to use a number from an irrelevant clause in the calculation).
- losing track of the problem structure (like forgetting a previous value or mixing up which numbers to add vs subtract).

With Math-DB prompts, such errors were moderately reduced. We saw the model explicitly ignore irrelevant sentences labeled "Other" in its solution, which is precisely what we want it to do. We also observed that sense labels disambiguated keyword-driven errors. For instance, the keyword "more" no longer reliably triggered addition in clauses labeled `Compare.Difference`. This indicates that the model correctly used the relation information both to filter out distractions and to select the intended arithmetic operation.

[5]While the error analysis provided here is qualitative, it is intended to illustrate one potential application of the Math-DB framework. Specifically, its utility in enhancing prompt-based reasoning rather than to serve as an exhaustive evaluation of LLM performance. The primary contribution of this work remains the definition, annotation, and validation of the Math-DB schema itself.

5.4 Discussion

The gains stem from two effects. First, "Other" labels prune the search space by explicitly marking clauses to exclude from the math, preventing distractor errors. Second, sense labels disambiguate operation-triggering keywords (e.g. "more" under `Compare.Difference` no longer defaults to addition), reducing logical errors. LLMs benefit from explicit guidance that plays to their strength in pattern-following.

6 Conclusion and Future Work

We introduced Math-DB, a domain-specific discourse framework and annotated corpus for mathematical word problems, and showed that Math-DB annotations enable GPT-4o-mini to maintain high accuracy on perturbed GSM-Symbolic problems, where standard CoT degrades. Future work includes:

Generality to Other Datasets: We focused on GSM-Symbolic. It would be interesting to apply discourse guided prompting to other math problem sets (like the original GSM8K, or more complex ones like MAWPS or Math23K, etc.). The relations are quite general for basic arithmetic word problems, but extending the taxonomy (e.g., for multi-step algebra or physics problems) could be needed.

Broader Reasoning Domains: The idea of domain-specific discourse frameworks could be applied beyond math. Other reasoning tasks (like scientific question answering, logical deduction puzzles, or multi-hop reading comprehension) might benefit from a similar approach.

End-to-End Pipeline: Our experiments use gold-standard Math-DB annotations as a proof of concept, establishing the upper-bound benefit of structured guidance. A natural next step is to build a fully autonomous parser-to-LLM pipeline in which discourse relations are predicted automatically at inference time and fed directly into the reasoning model.

In conclusion, the Math-DB demonstrates a synergy between discourse theory and LLMs.

Limitations

While Math-DB demonstrates clear benefits for mathematical reasoning, several limitations for this study should be considered:

- **Dataset Specificity:** This study focused primarily on the GSM-Symbolic dataset. While

the discourse relations are generalizable to basic arithmetic word problems, their applicability to more advanced domains, such as multi-step algebra, calculus, or physics may require further refinement and extension of the existing taxonomy.

- **Annotation Intensity and Scaling:** The current dataset was developed using a semi-automatic pipeline involving manual sense labeling by six trained annotators with mathematics backgrounds. While this ensured high-quality data, the reliance on human expertise for labeling represents a potential bottleneck for scaling the framework to significantly larger or more diverse datasets. A promising direction is to leverage the existing 47k annotated relations as training data for a supervised sense classifier, which could then propagate labels to new problems via bootstrapping or active learning, with human annotators verifying only the low-confidence cases. We leave the development and evaluation of such a scalable annotation pipeline to future work.

- **Pipeline Dependence and Segmentation Errors:** The annotation pipeline relies on an automatic preprocessing stage (connective detection and argument segmentation with a fine-tuned BERT model), followed by manual sense labeling. Any systematic segmentation errors (e.g., incorrect clause boundaries or misidentified connectives) can affect the validity of the human annotation process (sense labeling) negatively.

- **Oracle Evaluation and Pipeline Dependencies:** Our experiments utilize ground-truth discourse labels to evaluate the framework's reasoning potential, characterizing our results as an upper-bound analysis rather than a system-wide evaluation of a fully automated pipeline. In a practical deployment, any systematic errors from the initial segmentation or labeling stages (as discussed in Appendix A) would propagate to the reasoning model, potentially degrading performance. Future work is required to develop high-accuracy automated discourse parsers that can bridge this gap for end-to-end applications.

- **Focus on Adjacent Relations:** Our current framework primarily annotates adjacent clause pairs, as long-distance links are relatively rare in the short problems found in GSM-Symbolic, GSM-P1 and GSM-P2. However, more complex, multi-paragraph math problems may require capturing longer-distance discourse dependencies that the current sequential approach might overlook.

- **Language Scope:** The annotated corpus and experiments were conducted solely in English. Future work is needed to determine the framework's effectiveness across different languages to varied linguistic structures in a multilingual context.

- **Pipeline Coverage:** Our semi-automatic pipeline successfully annotated 11,414 of 12,500 GSM-Symbolic instances (91.3%); the remaining 1,086 instances failed at the alignment stage with detected discourse-relation counts inconsistent with their canonical templates. While we release full diagnostic logs for these instances, improving the robustness of the Stage-1 segmenter to reduce this failure rate is left to future work.

- **Single-Model Evaluation:** Our experiments use GPT-4o-mini as the underlying LLM, chosen because it was the model evaluated in the original GSM-Symbolic study (Mirzadeh et al., 2024), enabling direct comparison. The interaction between Math-DB guidance and (i) larger or more capable reasoning models (e.g., GPT-4, Claude, or specialized reasoning variants), and (ii) alternative structured prompting strategies (e.g., Least-to-Most, Tree-of-Thoughts, self-consistency) remains an open question. We hypothesize that discourse guidance provides complementary signal to these strategies rather than a substitute, but empirical verification is left to future work.

References

Daniel G Bobrow. 1964. A question-answering system for high school algebra word problems. In *Proceedings of the October 27-29, 1964, fall joint computer conference, part I*, pages 591–614.

Chloé Braud, Yang Janet Liu, Eleni Metheniti, Philippe Muller, Laura Rivière, Attapol Rutherford, and Amir Zeldes. 2023. The disrpt 2023 shared task on elementary discourse unit segmentation, connective detection, and relation classification. In *Proceedings*

of the 3rd Shared Task on Discourse Relation Parsing and Treebanking (DISRPT 2023), pages 1–21.

Thomas P Carpenter and James M Moser. 1979. An investigation of the learning of addition and subtraction.

Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. *arXiv preprint arXiv:2402.08939*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Gabriella Daroczy, Magdalena Wolska, Walt Detmar Meurers, and Hans-Christoph Nuerk. 2015. Word problems: A review of linguistic and numerical factors contributing to their difficulty. *Frontiers in psychology*, 6:348.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Charles R Fletcher. 1985. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5):565–571.

Karen C Fuson. 2012. *Children's counting and concepts of number*. Springer Science & Business Media.

James G Greeno. 1978. Natures of problem-solving abilities. In *Handbook of learning and cognitive processes: Vol. 5. Human information processing*, pages 240–269. Erlbaum Hillsdale, NJ.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 523–533.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281.

Zhenwen Liang, Jipeng Zhang, Jie Shao, and Xiangliang Zhang. 2021. Mwp-bert: A strong baseline for math word problems.

William C Mann and Sandra A Thompson. 1987. Rhetorical structure theory: Description and construction of text structures. In *Natural language generation: New results in artificial intelligence, psychology and linguistics*, pages 85–95. Springer.

Eleni Miltsakaki, Livio Robaldo, Alan Lee, and Aravind Joshi. 2008. Sense annotation in the penn discourse treebank. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 275–286. Springer.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Pearla Nesher. 2020. Levels of description in the analysis of addition and subtraction word problems. In *Addition and subtraction*, pages 25–38. Routledge.

Rashmi Prasad, Bonnie Webber, and Aravind Joshi. 2017. The penn discourse treebank: An annotated corpus of discourse relations. In *Handbook of linguistic annotation*, pages 1197–1217. Springer.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Mary S Riley and 1 others. 1984. Development of children's problem-solving ability in arithmetic.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. *Transactions of the Association for Computational Linguistics*, 6:159–172.

Krish Sharma, Niyar R Barman, Akshay Chaturvedi, and Nicholas Asher. 2025. Dimsum: Discourse in mathematical reasoning as a supervision module. *arXiv preprint arXiv:2503.04685*.

Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil Heffernan, Xintao Wu, Ben Graff, and Dongwon Lee. 2021. Mathbert: A pre-trained language model for general nlp tasks in mathematics education. *arXiv preprint arXiv:2106.07340*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1132–1142.

Gérard Vergnaud. 2020. A classification of cognitive tasks and operations of thought involved in addition and subtraction problems. In *Addition and subtraction*, pages 39–59. Routledge.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Bonnie Webber, Rashmi Prasad, Alan Lee, and Aravind Joshi. 2019. The penn discourse treebank 3.0 annotation manual. *Philadelphia, University of Pennsylvania*, 35:108.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Amir Zeldes, Yang Janet Liu, Mikel Iruskieta, Philippe Muller, Chloé Braud, and Sonia Badene. 2021. The disrpt 2021 shared task on elementary discourse unit segmentation, connective detection, and relation classification. In *Proceedings of the 2nd Shared Task on Discourse Relation Parsing and Treebanking (DISRPT 2021)*, pages 1–12.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Appendix: Data Preparation and Parameter Tuning for the BERT Fine-Tuning in Discourse Connective Detection and Argument Span Labeling Tasks

This appendix describes how we fine-tune a BERT-based sequence labeling model for (i) discourse connective detection and (ii) argument span labeling (Arg-1/Arg-2) in mathematical word problems (MWPs). We formulate both subtasks as a single token-level named entity recognition (NER) problem using a BIO tagging scheme. Tokenization is performed with `Stanza` (Qi et al., 2020), and each token is assigned one label from the set

$$\mathcal{Y} = \{\texttt{O}, \texttt{B-Conn}, \texttt{I-Conn}, \texttt{B-Arg1}, \texttt{I-Arg1}, \texttt{B-Arg2}, \texttt{I-Arg2}\}$$

Tokens outside any discourse span are labeled `O`. We use the convention that `B-` marks the beginning of a span and `I-` marks tokens inside the span.

A.1 Task formulation as token-level NER

Given an input text (a window containing a discourse relation instance), we produce a token sequence $(x_1, \ldots, x_n)$ and a corresponding label sequence $(y_1, \ldots, y_n)$ with $y_i \in \mathcal{Y}$. The goal is to predict the discourse connective span and the two discourse arguments (Arg-1 and Arg-2) as contiguous token spans. This converts connective detection and argument span labeling into standard BIO sequence labeling.

Relation-instance encoding: A single MWP typically contains multiple adjacent clause relations (e.g., $\text{Clause}_1 \rightarrow \text{Clause}_2$, $\text{Clause}_2 \rightarrow \text{Clause}_3$, etc.). To avoid label conflicts in a single long sequence, we construct the dataset using *relation instances*. For each adjacent clause pair (Arg-1→Arg-2), we create one training example consisting of a local text window that contains Arg-1, an optional connective (if explicit), and Arg-2. Each relation therefore yields one NER instance with a single Arg-1 span, a single Arg-2 span, and optionally one connective span. This representation is both simpler and empirically more stable than whole-problem multi-span tagging.

BIO labeling rules: For each gold span type $S \in \{\texttt{Conn}, \texttt{Arg-1}, \texttt{Arg-2}\}$, we label the first token overlapping the span as `B-`S and any subsequent overlapping tokens as `I-`S. Tokens that do not overlap any gold span are labeled `O`.

A.2 Illustrative example

Table 3 shows an example relation instance and its BIO labels in a CoNLL-style format. Arg-1 is the first sentence, *Then* is an explicit connective, and Arg-2 is the second sentence.

Token	Label
Lina	B-Arg1
has	I-Arg1
8	I-Arg1
marbles	I-Arg1
.	I-Arg1
Then	B-Conn
she	B-Arg2
gives	I-Arg2
3	I-Arg2
marbles	I-Arg2
to	I-Arg2
her	I-Arg2
friend	I-Arg2
.	I-Arg2

Table 3: Example of tokenization and BIO labels for a discourse relation instance.

For *implicit* relations, we do not fine-tune any model since we assume all adjacent clauses build an implicit discourse relation if they do not build an explicit discourse relation.

A.3 Model architecture

We fine-tune a pretrained BERT encoder with a token classification head: (i) a Transformer encoder (e.g., `bert-base-uncased` or `bert-base-cased`), and (ii) a linear layer mapping contextual representations h_i to label logits over $\mathcal{Y}$. We optimize token-level cross-entropy loss over non-ignored positions.

A.4 Training objective and evaluation

Objective: Given token sequence $(x_1, \ldots, x_n)$, the model predicts label distribution $p(y_i \mid x)$ for each token. We minimize the negative log-likelihood (cross-entropy) over labeled positions:

$$\mathcal{L} = -\sum_{i=1}^{n} \mathbf{1}[y_i \neq \texttt{IGNORE}] \log p(y_i \mid x).$$

Metrics: Table 4 reports span-level (entity-level) F_1 scores for each predicted span type. The model achieves very strong performance on connective span detection (F_1=0.96), reflecting that explicit connectives are often short and lexically distinctive. Arg-2 spans are also identified reliably (F_1=0.91). Arg-1 span labeling is comparatively harder (F_1=0.86), which we attribute to more variable Arg-1 boundaries and frequent adjacency to preceding context, making precise start/end decisions less consistent than for connectives or Arg-2.

Task	Precision (%)	Recall (%)	F_1 (%)
Connective span detection	97.11	95.50	96.30
Arg-1 span labeling	87.32	84.58	85.93
Arg-2 span labeling	92	89.78	90.88

Table 4: Span-level (entity-level) performance of the BERT token-classification model on the test set.

Discussion: Table 4 shows strong span extraction performance across all subtasks. Connective span detection is the easiest (F_1=0.96), likely because explicit connectives are typically short and lexically distinctive. Arg2 spans are also identified reliably (F_1=0.91). Arg1 span labeling is comparatively harder (F_1=0.86), which we attribute to greater boundary variability (e.g., optional lead-in context and punctuation) and stronger dependency on discourse segmentation decisions.

A.5 Hyperparameter tuning

We treat the task as standard NER fine-tuning and tune hyperparameters on the dev set, selecting the checkpoint with the best dev micro span-level F_1. We recommend reporting the full search space and the chosen configuration.

Core hyperparameters (tuned):

- Learning rate: $\{1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}$
- Batch size (effective): $\{16, 32\}$
- Epochs: $\{3, 5, 10\}$ with early stopping on dev F_1
- Max sequence length: $\{128, 256, 384\}$ (relation windows typically fit within 256)
- Warmup ratio: $\{0.0, 0.06, 0.1\}$
- Dropout: BERT default (0.1)

We use AdamW with weight decay 0.01 and a linear learning rate schedule with decay.

Seeds and reporting: We run each configuration with multiple random seeds (e.g., 3) and report mean$\pm$std on test F_1 for robustness.

B Appendix: Math-DB Annotation Manual

B.1 Purpose and Scope

Math-DB adapts the Penn Discourse TreeBank (PDTB) framework to math word problems.

- A **problem** is represented as an **sequence of clauses** $C_1, C_2, \ldots, C_n$ in text order.
- For every **adjacent clause pair** $(\text{Arg-1} \rightarrow \text{Arg-2}) = (C_i \rightarrow C_{i+1})$, annotate **exactly one** discourse relation (DR) sense.
- DR senses describe how **Arg-2 relates to Arg-1** in a math-reasoning way and are intended to scaffold quantitative reasoning steps.

Math-DB is designed for grade-school arithmetic word problems (e.g., GSM8K-style) and emphasizes relations that map directly to common quantitative operations.

B.2 Units of Annotation

B.2.1 Clause

A **clause** is the smallest meaningful proposition used for math reasoning. Clauses typically contain: (i) a quantity (explicit number or implied variable), (ii) an entity (e.g., apples, money, students), and (iii) a predicate expressing a state/event/relation (e.g., "has", "buys", "costs", "is", "more than").

Clauses are labeled in order: $C_1, C_2, \ldots, C_n$.

B.2.2 Discourse Arguments (Arg-1, Arg-2)

For each adjacent pair:

$$\text{Arg-1} = C_i, \quad \text{Arg-2} = C_{i+1}.$$

The relation direction is always forward in text order (Arg-1 precedes Arg-2). Only **adjacent pairs** are annotated.

B.3 Sense Inventory

Each discourse relation has the form `Level1.Level2`, where

$$\texttt{Level1} \in \{\texttt{Change}, \texttt{Combine}, \texttt{Compare}, \texttt{Equalize}, \texttt{Other}\}.$$

B.3.1 Change (State transition over time)

Arg-2 updates a quantity introduced in Arg-1 via an event over time.

- `Change.Increase`: the tracked quantity increases (gain, receive, buy, plant, add, arrive).
- `Change.Decrease`: the tracked quantity decreases (lose, spend, give away, remove, eat, sell, break).

Diagnostic: Arg-1 and Arg-2 refer to the **same tracked quantity** at different times.

B.3.2 Combine (Composition into a whole)

Arg-2 contributes to forming a whole, either additively or multiplicatively.

- `Combine.PartWhole`: aggregation of parts into a total (*total, altogether, combined, in all, the rest*).
- `Combine.Product`: structured multiplication ($m \times n$), including arrays/area, repeated groups, combinations, and unit-rate multiplication ("$3 each", "5 per day").

Diagnostic: Arg-1 and Arg-2 jointly define a **composed quantity** (sum or product), not a time update.

B.3.3 Compare (Relational linking without merging)

Arg-2 relates two quantities by difference or ratio without forming a single whole.

- `Compare.Difference`: additive comparison ("how many more/less", difference).
- `Compare.Ratio`: multiplicative comparison or rate ("times as many", "per", "each", unit price/speed/-density).

Diagnostic: Two quantities are linked for comparison, not combined into one total.

B.3.4 Equalize (Reach equality)

Arg-2 frames a goal of making quantities match, either by additive adjustment or scaling.

- `Equalize.MatchDifference`: reach equality by adding/removing an amount ("how many more needed to be the same").
- `Equalize.MatchRatio`: reach equality by scaling ("make it k times", "scale until equal").

Diagnostic: The intended target state is **equality**.

B.3.5 Other (Non-quantitative or auxiliary)

Use `Other` when the clause pair does not introduce a new quantitative step or core math relation. This covers two main cases: (i) narrative context or descriptive details not used in the computation, and (ii) the final question clause of the problem, which identifies the target quantity but does not itself introduce an arithmetic operation between quantities. Prefer a non-`Other` sense when Arg-2 changes, composes, compares, or equalizes quantities in a way used for solving.

B.4 Clause Segmentation Guidelines

B.4.1 General rule

Split into clauses so that each clause expresses **one main proposition** relevant to the math.

B.4.2 Strong split cues

Split at:

- sentence boundaries,
- semicolons,
- discourse markers (*then, after, later, before, when, while, however, but*) when they introduce a new event or relation,
- enumerations that introduce separate quantities.

B.4.3 Keep together when

Do **not** split when a phrase is tightly bound as one atomic math fact (e.g., "5 bagels for $3 each"), unless separate handling is needed.

B.4.4 The question

The final interrogative typically becomes its own clause (C_n).

B.5 Relation Selection Heuristics

Use these diagnostics in order:

1. Does Arg-2 **update a previously tracked quantity over time**? → `Change.Increase` or `Change.Decrease`
2. Does Arg-2 **compose a whole** from parts or groups? → `Combine.PartWhole` or `Combine.Product`
3. Does Arg-2 **compare** quantities (difference or ratio) without merging them? → `Compare.Difference` or `Compare.Ratio`
4. Does Arg-2 express **reaching equality**? → `Equalize.MatchDifference` or `Equalize.MatchRatio`
5. Otherwise → `Other`

B.6 Common Constructions and Recommended Labels

B.6.1 "Each / per / at a rate"

- If used to compute a total (price × quantity, speed × time): `Combine.Product`.
- If emphasized as a relational comparison: `Compare.Ratio`.
- When uncertain in GSM-style problems, prefer `Combine.Product` for "each/per" totals.

B.6.2 "X more / fewer / less than"

- Usually `Compare.Difference`.
- If defining a required adjustment to match a target: `Equalize.MatchDifference`.

B.6.3 "Times as many / double / triple / half"

- If comparing two quantities: `Compare.Ratio`.
- If computing a produced total across repeated groups: `Combine.Product`.
- "Half ... swam away" is typically `Change.Decrease` (event reducing count).

B.6.4 Multi-step updates

Successive events on the same quantity are annotated as a chain of `Change.*` relations across adjacent clauses.

B.6.5 "The rest"

Often `Combine.PartWhole` (whole/part inference), possibly followed by `Change.Decrease` if phrased as removal.

B.7 Worked Example

Problem: "A juggler can juggle 640 balls. An eighth of the balls are tennis balls, and 1/10 of the tennis balls are white. How many white tennis balls are there?"

Clauses

- C_1: A juggler can juggle 640 balls.
- C_2: An eighth of the balls are tennis balls.
- C_3: 1/10 of the tennis balls are white.
- C_4: How many white tennis balls are there?

DRs

- Arg-1 $= C_1 \rightarrow$ Arg-2 $= C_2$: `Combine.Product`
- Arg-1 $= C_2 \rightarrow$ Arg-2 $= C_3$: `Combine.Product`
- Arg-1 $= C_3 \rightarrow$ Arg-2 $= C_4$: `Other`

C Appendix: Prompt Templates for CoT Prompting with Math-DB and Baseline Study

Prompt Template: Math-DB Reasoning

INSTRUCTIONS:
You are a mathematical reasoning engine. Solve the question step by step.
A) Mathematical DB Framework (Math-DB)
Some problems may come with a DB-style sequence of discourse relations (DR) between adjacent clauses. Each DR is a label of the form Level1.Level2 describing how Arg-2 relates to Arg-1.
Level-1 senses:

- Change: Arg-2 updates a quantity from Arg-1 via an event.
- Combine: Arg-1 and Arg-2 compose into a whole (additively or via a structured product).
- Compare: Arg-2 relates two quantities by difference or by ratio/rate.
- Equalize: Arg-2 expresses making quantities match (equality), by difference or by scaling.
- Other: adjacency without a quantitative relation from this inventory.

Level-2 senses:

- Change.Increase / Change.Decrease
- Combine.PartWhole (part-part-whole aggregation) / Combine.Product (structured product: m×n, area/arrays, combinations)
- Compare.Difference (more/less, difference) / Compare.Ratio (times-as-many, each/per, rates)
- Equalize.MatchDifference (add/remove to match) / Equalize.MatchRatio (scale to match)
- Other

B) Arithmetic operations (use when building equations)
1. ADDITION (+): combine distinct quantities to find a total (sum, total, altogether, receive).
2. SUBTRACTION (-): difference, removing a quantity, or additive comparison (left, fewer, lost, give away).
3. MULTIPLICATION (*): repeated groups, scaling, or structured product (times, each, double, rows×columns).
4. DIVISION (/): splitting into equal parts or solving inverse of a multiplicative relation (share, split, average, per-unit).
Always ground equations in the entities and relations described in the problem.

DEMONSTRATIONS:

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
Discourse Information:
1. "There are 15 trees in the grove."
2. "Grove workers will plant trees in the grove today." (Relation: Change.Increase)
3. "After they are done, there will be 21 trees." (Relation: Change.Increase Result)
4. "How many trees did the grove workers plant today?"
A: Math-DB: The situation is a Change relation: planting increases the number of trees, and we are given the final result.
Let start = 15 and result = 21. This is Change.Increase with unknown_role = delta (how many were added). So delta = result - start = 21 - 15 = 6. The final answer is 6.

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
Discourse Information:

[...Discourse Clauses and Relations...]

A: Let start = 3 and delta = 2, with unknown_role = result. So result = start + delta = 3 + 2 = 5. The final answer is 5.

[...Remaining Demonstrations Omitted...]

Q: {target_question}
Discourse Information:
{target_discourse_info}
A: Let's think step by step.

Prompt Template: Baseline CoT (GSM-Symbolic)

```
// preamble or system instruction
As an expert problem solver, solve step by step the following mathematical questions.
```

```
// shot-1
Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they
are done, there will be 21 trees. How many trees did the grove workers plant today?
A: Let's think step by step. There are 15 trees originally. Then there were 21 trees after some
more were planted. So there must have been 21 - 15 = 6. The final answer is 6.
```

.
.
.

```
// shot 8
Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
A: Let's think step by step. Olivia had 23 dollars. 5 bagels for 3 dollars each will be 5 * 3 =
15 dollars. So she has 23 - 15 dollars left. 23 - 15 is 8. The final answer is 8.
```

```
// target question
Q: {{{question}}}
A: Let's think step by step.
```

D Appendix: Inter-Annotator Agreement

Although we used majority vote rather than measuring exact agreement, we did compute inter-annotator agreement on a randomly sampled subset of 1,000 relations to measure the difficulty of the annotation task. Among the three annotators on each relation, the initial agreement (before adjudication), measured by Fleiss' κ, was 0.84 on the Level-2 sense. This is reasonably high, indicating that the framework was clear for most cases. Fleiss' κ for the five Level-1 sense categories was 0.92, showing substantial agreement. The most confusion was in differentiating Change vs Combine in cases of additive actions, and identifying Other vs a weak Combine/Change link. After discussions and adjudication, those were resolved in the final version of the Math-DB.

Overall, the Math-DB annotation process yielded a high-quality dataset of discourse-labeled math problems. This resource not only enables our experiments but can also help future studies, such as training automated discourse parsers for math text or analyzing common patterns in math problem structure.