# NumDecoders at SemEval-2024 Task 7: FlanT5 and GPT enhanced with CoT for Numerical Reasoning

**H. Andres Gonzalez Gongora**♣[1]*, **Md Zobaer Hossain**♣[2], **Jahedul Alam Junaed**♠[3]
♣ University of Lorraine, Nancy, France
♠ Shahjalal University of Science and Technology, Sylhet, Bangladesh
nanoandres_24@hotmail.com[1]
rowan.hossain@gmail.com[2]
jahedul25@student.sust.edu[3]

## Abstract

In this paper we present a Chain-of-Thought enhanced solution for large language models, including flanT5 and GPT 3.5 Turbo, aimed at solving mathematical problems to fill in blanks from news headlines. Our approach builds on a data augmentation strategy that incorporates additional mathematical reasoning observations into the original dataset sourced from another mathematical corpus. Both automatic and manual annotations are applied to explicitly describe the reasoning steps required for models to reach the target answer. We employ an ensemble majority voting method to generate final predictions across our best-performing models. Our analysis reveals that while larger models trained with our enhanced dataset achieve significant gains (91% accuracy, ranking 5th on the NumEval Task 3 leaderboard), smaller models do not experience improvements and may even see a decrease in overall accuracy. We conclude that improving our automatic annotations via crowdsourcing methods can be a worthwhile endeavor to train larger models than the ones from this study to see the most accurate results.

## 1 Introduction

NumEval is a task first introduced in 2024 (Chen et al., 2024) building on previous work such as Cortis et al. (2017)'s fine-grained sentiment analysis (SemEval-2017 Task 5) and Jullien et al. (2023)'s clinical inference (SemEval-2023 Task 7). These prior tasks highlighted the importance of understanding numerical values in legal and medical contexts for determining outcomes. The primary objective of NumEval is to perform quantitative reasoning to generate numerical values corresponding to provided contexts.

In this project, we particularly focused on subtask 1 of task 3 (Huang et al., 2023) where our system must execute several mathematical calculations based on information from a provided passage to yield a numerical result used to fill in a headline with a blank. For instance, to complete the *CIA Cited Concerns About Snowden ____ Years Ago* headline, the model must subtract the article's publishing date by the explicitly stated date in the article (2009). Some entries involve a series of multiple mathematical operations that the model must perform.

Although numerical reasoning continues to present challenges to large language models (LLMs), advancements in larger models like DeepSeekMath (Shao et al., 2024) demonstrate promising capabilities in solving mathematical computations. DeepSeekMath is finetuned using different mathematical datasets and evaluated using Chain-of-Thought (CoT) prompting to provide intermediate reasoning steps. Inspired by CoT systems, we have developed a system pipeline that trains an encoder-decoder flanT5 (Chung et al., 2022) and an open source GPT 3.5 version[1] with additional mathematical corpora. These corpora include the Discrete Reasoning Over the Content of Paragraph (DROP) dataset (Dua et al., 2019) and another dataset which was manually and automatically annotated to include reasoning steps to reach the desired response. The core idea is that explicit intermediate reasoning, akin to chain-of-thought prompts, can enhance a model's quantitative reasoning capabilities (Wei et al., 2023).

In our revised approach, not only do we use smaller models ($\theta \leq 1B$)[2], but we also utilize multiple pipelines to determine the conditions under which our model achieves the highest accuracy. Firstly, we establish a baseline by fine-tuning with the provided dataset (Huang et al., 2023), then we incorporate additional observations from the DROP dataset into our training data. Thirdly, we adopt

*All authors have equal contributions

[1]List of open source OpenAI GPT models
[2]$\theta$ refers to model parameters



*Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1260–1268
June 20-21, 2024 

a Chain-of-Thought (CoT) approach, fine-tuning both a flanT5 model and a generative open-source OpenAI model (GPT 3.5 Turbo) with more detailed inputs and outputs, including string normalizations and quantitative reasoning steps. Finally, we employ an ensemble majority voting method to select the best results from these models, resulting in a 91% accuracy and 5th place on the leaderboard of the NumEval competition [3].

## 2 Related Works

Through pre-training on a vast amount of text data, LLMs can develop a broad knowledge base encompassing numerical concepts, arithmetic operations, and mathematical relationships. Lewkowycz et al. (2022) propose a language model named Minerva, which demonstrates strong performance on various quantitative reasoning tasks, including undergraduate-level physics or chemistry problems.

Numerical reasoning has been extensively studied across diverse contexts, including word embedding (Wallace et al., 2019; Naik et al., 2019; Sundararaman et al., 2020) and math word problems (Wang et al., 2018; Cobbe et al., 2021). Within the domain of Question Answering, several approaches have been proposed. Xu et al. (2022) present a framework called Diagnosing Numerical Capabilities (DNC), which involves two stages: recognition of numbers in the context and question to treat them as candidate operands, followed by the correct selection of operands and operations based on understanding questions and context. Kim et al. (2022) proposes an attention-masked reasoning model that learns to leverage the number-related context to alleviate the over-reliance on parametric knowledge and enhance the numerical reasoning capabilities of the QA model. Other studies, such as those by Geva et al. (2020) and Feng et al. (2021), explore the infusion of external knowledge to augment the numerical reasoning skills of the models. Yang et al. (2021) focus on Numerical Reasoning over Text (NRoT) using T5 models, employing five training pipelines and multitasking training to progressively enhance model performance through tasks such as general reading comprehension and fine-tuning on the DROP dataset (Dua et al., 2019). Additionally, in reasoning tasks, Chain-of-Thought prompting has shown promise in improving the performance of large language models (Ling et al., 2024). While Chain-of-Thought (CoT) allows models to generate more comprehensive reasoning processes, it also introduces challenges such as hallucinations and accumulated errors. To mitigate these issues, the authors propose enabling explicit and deductive rigorous reasoning within language models. They emphasize the importance of self-verification for trustworthiness, which leads to significantly improved answer correctness in reasoning tasks. Drawing inspiration from these CoT-based methods, we incorporate them into our approach due to their superior performance in numerical reasoning tasks.

## 3 System Description

In our system, we defined three main pipelines that were compared against a baseline encoder-decoder model. Specifically, we used an instruction finetune model version (flan) of the Text-To-Text Transfer Transformer (T5) (Chung et al., 2022). This flanT5 model underwent fine-tuning in its small, base, and large versions, employing a learning rate of 5e-5 for 5 epochs and a batch size of 2.

### 3.1 DROP Dataset

To enhance performance beyond the baseline, we merged the Discrete Reasoning Over the Content of Paragraph (DROP) dataset (Dua et al., 2019) with the original numerical headline generation dataset (Huang et al., 2023). The DROP dataset consists of paragraphs with answer spans to given questions, often referencing multiple positions in the provided passage. With a total of 77400 observations in the training data split, we filtered out 46973 observations related to numerical reasoning tasks. Due to computational constraints, we merged only 20000 of these filtered observations with the original headline generation dataset. The selection of these 20,000 entries was based on a random seed of 43. Additionally, it's important to note that while the input text in the DROP dataset is structured as questions, unlike the fill-in-the-blank format used Huang et al. (2023)'s dataset, we transformed the questions into masked headlines by locating the answer in the original dataset and masking it from the passage's headline.

### 3.2 GPT 3.5 turbo

For this task, we utilized the GPT 3.5 Turbo model to extract numerical reasoning and explanations from the NumHG dataset (Huang et al., 2023).

[3]GitHub repository for our system

Prompt selection plays a critical role in obtaining optimal output from the GPT model. White et al. (2023) outline various prompt engineering techniques in a pattern-based catalog that have been successfully applied to improve the outputs of large language models (LLMs) in conversations. Drawing from the insights provided by White et al. (2023), we adopt three distinct patterns into our prompt design: the Persona Pattern, the Context Manager Pattern, and the Recipe Pattern. Each pattern was carefully selected to address specific challenges and enhance the interpretability of the generated responses.

**Persona Pattern:** It assists the GPT model in determining the types of output to generate and which details to prioritize. By incorporating persona-based prompts, we guide the model to discern the essential information to emphasize in its responses.

**Context Manager Pattern:** The goal of this pattern is to focus on specific topics and exclude unrelated ones from consideration. Through careful manipulation of contextual cues, we enhance the model's ability to generate contextually relevant and coherent numerical explanations.

**Recipe Pattern:** It introduces constraints to ultimately output a sequence of steps based on partially provided "ingredients" required to achieve a specified goal. Serving as a structured framework for our prompt design, the Recipe Pattern guides the model in constructing step-by-step sequences.

| Role | Content | Matched Pattern |
|---|---|---|
| System | You are a helpful assistant, skilled in providing numerical reasoning. | Persona Pattern |
| User | context: [news] + [masked headline] | – |
| User | The answer to the fill-in-the-blank question is [ans]. Please provide a complete sequence of numerical reasoning steps in a paragraph format that is used to derive this answer. Begin your response by discussing the relevant sentences, and then outline the numerical reasoning steps. Conclude your response with: 'So the answer is [ans].' | Context Manager & Recipe Patterns |

Table 1: Conversation prompt with matched patterns. Here, placeholder values are from the dataset.

### 3.3 Chain of Thought (CoT)

To further steer the capabilities of both the decoder GPT 3.5 Turbo and our trained flanT5, we incorporated chain-of-thought (CoT) prompting (Wei et al., 2023). This involved adding specific reasoning steps in the output text that the model relied on to produce the numerical response. According to Wei et al. (2023), CoT outperforms traditional prompting and finetuning approaches by providing intermediate reasoning steps that facilitate model interpretation. Moreover, in large models, even a few CoT sequences can outperform some finetuned pre-trained models in arithmetic and symbolic reasoning tasks (Wei et al., 2023).

In our CoT pipeline, our initial approach involved an automatic annotation step, which we supplemented with manual annotation to handle more complex calculations. Below, we outline this annotation process, including additional preprocessing steps implemented to normalize the input and output data.

**Automated Annotation:** In the original news articles, dates are written in abbreviated form and placed within brackets before the passage. Since many headline completion tasks involve subtracting a given number of years mentioned in the article from the publishing date, we extract this metadata date and transform it to prefix the overall passage with a descriptive sentence. For instance, an article with the date *(Feb 13, 2013 6:54 PM)* is transformed to *The news was published on 13th February in the year of 2013*. This approach enables our models to retrieve explicit and normalized dates for performing the corresponding mathematical operations.

Answer extraction is conducted using the *spacy* module to tokenize each passage and iterate over each resulting sentence with a custom placeholder function. If the answer is found within a sentence, it is extracted. The main answer extraction function is then applied to our main 7 placeholder functions to automate the annotation of the simpler calculations. Among these, 5 (*copy, translation, round, sround, and paraphrase* (Huang et al., 2023)) are much more straightforward, whereas *subtract* and *span* require a heuristic-based annotation, where the answer string is preprocessed to fit the appropriate format.

For instance in our span recipe, (*get_span_placeholder*), we modify the resulting string if the blank contains the following tokens.

- **No._** which we pass to the model as output with the following explanation: *No. 1 typically refers to the topmost or the best-ranked item in a list or a competition.*

- **_M** which we pass to the model as output with the following explanation: *The letter 'M' in the headline indicates that the answer refers* *to an amount that should be transformed to millions*

- **_st** which we pass to the model as output with the following explanation: *The presence of 'st' in the headline gives a clue that the answer is 1*

Otherwise, we specify that the span containing the answer may refer to a person, object or event.

As previously stated, each of the simple aforementioned calculations has its own placeholder function, which we further examine in Table 6 and pass to our main algorithm in Figure 1.

One of the biggest challenges the automation system faced was inconsistent annotations from the original dataset wherein certain passages would not contain references to the answers at all or, more egregiously, wrong calculations. For the headline *Wife who got $1B in Divorce: Not Enough* where 1 corresponds to the answer, the calculation is as follows *Round(Paraphrase(995, K), 0)*. Nevertheless, the paraphrase should have an *M* instead of a K as the value is given in the millions rather than the thousands.

Furthermore, apart from the provided calculation, the passages often lack explicit numerical reasoning to justify why a model should yield the floor value of a decimal number for a headline instead of rounding it up. For example, in the article *Woman Places $615K Bet on Hillary Clinton* the value must be paraphrased and then rounded up to the nearest whole number to reach the answer of 615. However, the passage states that “"a 46-year-old woman just placed a $615,862 bet on Clinton". Mathematically, the number should be converted to thousands by dividing by 1000 and then rounded up, resulting in an answer of 616. Notwithstanding, the headline reports 615.

**Manual Annotation:** We employed manual annotation to address more complex operations, including addition, subtraction, multiplication, and division. In each case, we began with an automated step using GPT 3.5, as described in Table 1 and then manually cleaned up the reasoning steps, as well as, overall responses using a frontend system built with *streamlit*. Figure 2illustrates an example where we manually corrected the automated annotation to describe the steps for solving both simple and more complex calculations in a fill-in-the-blank question. In some instances, answers were incorrect, or the original logic provided by the model was overly redundant or incorrect. Consequently, we relied on 3 main human annotators[4] to review the 1K annotations completed by GPT 3.5 turbo. In Table 7, we can see some examples of patterns annotators followed to make sure the dataset would be consistent.

With both our automatic and manual annotations combined, we proceeded to fine-tune our GPT 3.5 Turbo and flanT5 models to evaluate whether the improved dataset yielded any advantages over the baseline. For this fine-tuning process, we maintained the same hyperparameters as before, except for the batch size. The batch size was increased for the small and base-sized flanT5 models to 16 and 8, respectively. This adjustment was necessary because we trained these models using a larger GPU, an A100 40GB GPU.

### 3.4 Ensemble

Lastly, we implemented an ensembling method using majority voting, wherein for each passage, we selected the numerical answer with the most votes as the correct one. In our ensembling pipeline, we narrowed down our majority voting to 4 models, consisting of our best-performing models: one version of large flanT5 trained for 3 epochs using only the NumHG dataset, another large flanT5 trained using NumHG for 2 epochs, a flanT5 trained for 2 epochs using the DROP dataset, and a CoT fine-tuned version of GPT 3.5 Turbo. We included versions that were trained for 2 epochs instead of 3 as they outperformed their 3-epoch counterparts, particularly the DROP-trained flanT5. However, this was only the case with the large models, as the base and small ones consistently performed better after training for 3 epochs rather than 2. During the evaluation period, we were unable to finish training the CoT models; therefore, we only used the available top 4 models for ensembling.

Since we employed an even number of models for this method, the likelihood of encountering ties is high. In instances of a tie, where a unanimous answer majority was absent, we resorted to the answer generated by our top-performing model—FlanT5 fine-tuned exclusively with NumHG.

## 4 Results

Based on the results presented in Tables 2 and 3, we observe that the difference in performance between the small and base flanT5 models is not par-

[4]These annotators are the authors of this paper

```python
def get_ans_sent(item):

   operations = {"Copy":get_copy_placeholder,"Trans":get_trans_placeholder,
              "Span":get_span_placeholder,"Round":get_round_placeholder,"Paraphrase":
                   ↪ get_paraphrase_placeholder,
              "Subtract":get_subtract_placeholder, "SRound":get_round_placeholder}

   for operation, function in operations.items():

      if check_calculation(item, operation):

         return function(item)

   return f"So␣the␣answer␣is␣{item['ans']}"
```

Figure 1: Main function used to annotate our data automatically. Each placeholder contains the find answer function, which tracks the main spans needed to fill in the blank question.

| | T5 Flan Small | T5 Flan Base | T5 Flan Large |
|---|---|---|---|
| NumHG | 0.83 | 0.89 | 0.91 |
| NumHG+DROP | 0.84 | 0.88 | 0.90 |
| COT | 0.58 | 0.83 | 0.88 |

Table 2: Results of T5 Flan models trained on three different datasets with the validation set.

| | T5 Flan Small | T5 Flan Base | T5 Flan Large |
|---|---|---|---|
| NumHG | 0.82 | 0.84 | 0.90 |
| NumHG+DROP | 0.83 | 0.88 | 0.90 |
| COT | 0.58 | 0.83 | 0.88 |

Table 3: Results of T5 Flan models trained on three different datasets with the test set.

ticularly notable, except when employing the CoT method, where the small models significantly underperform. Additionally, as shown in Table 4, it is surprising to note that a finetuned GPT 3.5 Turbo model underperforms compared to the other flanT5 models, despite its larger size. Overall, our team ranked 5th out of 16 teams, including the baseline, on the final leaderboard, achieving 91% accuracy with our majority model.

| | dev | test |
|---|---|---|
| NumHG | 0.91 | 0.90 |
| NumHG+DROP | 0.90 | 0.90 |
| COT | 0.88 | 0.88 |
| GPT 3.5 | 0.85 | 0.84 |
| GPT 3.5 (fine tuned) | 0.81 | 0.82 |
| Ensemble (Majority) | 0.92 | 0.91 |

Table 4: Best Results of the models on validation and test set.

## 5 Discussion

Our CoT results, as observed in Tables 2 and 3 align with the findings reported by Wei et al. (2023), indicating that smaller models do not experience significant gains when using prompting, partly due to their fewer parameters. In their study, it is explicitly mentioned that models in the range of 100 billion parameters or more exhibit the highest gains. However, all of the flanT5 models we utilized have significantly fewer parameters, failing to reach the 1 billion mark (Chung et al., 2022). We believe that conducting CoT experiments with the XL and XXL versions of these models would likely result in much more significant improvements.

### 5.1 Error Analysis

For our error analysis, we converted our model predictions into strings to facilitate comparison with their corresponding ground truths. It's important to note that while the competition required numerical values to be uploaded, some ground truths were formatted with commas (e.g., 1,500 instead of 1.5) or included important dates such as 9/11. In cases like the latter, where the ground truth couldn't be converted to a real number, we cast our results to string values. However, even with this adjustment, discrepancies in formatting, such as our model yielding 4.5 while the ground truth is 4.50, resulted in evaluations as incorrect. When accounting for these differences, the accuracy rate of the majority voting ensembling method reached 93%. Additionally, some answers in the test set were tagged as unanswerable.

In Table 5, we observe the error rate of our best majority voting ensembling method. Despite our best-performing model achieving a 91% accuracy rate, as noted in Table 4, we can see a high error rate for complex operations such as addition, multiply, and subtraction. Additionally, the surprisingly high error rate for the round operation may stem from inconsistencies in the annotation process. As mentioned in Section 3, there are no contextual hints in the passage besides the calculation to aid the model in flooring a value instead of rounding it up. Moreover, certain calculations that instruct the model to round up a value, such as *2.8*, have a ground truth of 2 instead of 3.

Nevertheless, our models encountered several round-up errors where they failed to generalize properly, particularly when rounding up to the nearest tenth. For example, in an operation yielding 4.831 where rounding up to the nearest tenth should result in 4.83, our models rounded it up to 4.8. Similarly, in cases where 4.8 should be rounded up to the nearest whole number, our flanT5 models often failed to round it up to 5, opting instead for 4. In approximately 80% of cases where round operations were inaccurately predicted, the primary issue was the selection of an incorrect upper or lower bound for the rounding operation.Many of these mistakes involved multiple complex calculations, where a round operation had to be computed after 2 or 3 additional computations. An example of this issue can be seen in the operation: *Round(Divide(85,12),0)* where the result is supposed to be 7, but the model incorrectly yields 85 to complete the headline *Robert Durst Gets ____ Years for Gun Charges*. However, the article explicitly states “"Robert Durst, millionaire oddball and star of HBO’s true-crime documentary The Jinx, pleaded guilty to gun charges Wednesday in New Orleans, earning him 85 months in prison". While the headline requests years, the model fails to convert the value in months to years by dividing by 12, instead simply copying the number *85* from the span.

Similarly, we encountered errors with the copy operation, where either the model would copy an incorrect value or, more egregiously, round it up to another value. For instance, in the headline *Spanish Bank Offers $____B to Madoff Victims* the article states that “Spanish banking giant Banco Santander, whose clients lost nearly $3.1 billion in Bernard Madoff’s Ponzi scheme, has offered to pay back customers some $1.82 billion, reports Bloomberg.". Therefore, the correct answer should be 1.82, but our model incorrectly rounds this value to 1.8. We estimate that if we account for these errors, our majority ensembling could potentially achieve a 2% increase over our 91% result.

Finally, the divide calculation presents fewer errors in our models, with most mistakes occurring within complex operations involving multiple calculations. However, it’s worth noting that ratio conversion, specifically converting a ratio to a percentage and viceversa, poses a challenge for our models. This challenge is evident in the *Divide(1,20%)* calculation where the expected result is 5 to complete the headline *Odds of a Depression? 1 in ____*. Unfortunately, our model yields 4, indicating that it interpreted 20% as 25%. While such corner cases underscore that our models may not always accurately translate fractions to their corresponding real numbers (i.e., dividing the percentage number by 100), it’s important to consider the context. In the article, multiple percentages were mentioned in the following spans: “The bad news is that this recession is likely to be America’s worst since WWII—but the good news is there’s only a 20% chance it will become a depression (...) The lack of any major global conflicts means the chance of a depression being a major one—a decline of 25% or more—is only 2%". In other words, the model did not identify the span with the right percentage to convert to its corresponding ratio, which is 20% rather than 25%.

## 6 Conclusion

In this paper we observed that introducing additional observations with detailed reasoning steps can enhance a model’s ability to solve mathematical problems while also highlighting areas where its reasoning may fall short. Nevertheless, our results suggest that larger models may derive the most benefit from a CoT + finetuning approach. Because of this, we argue that leveraging larger LLMs could lead to even greater gains in quantitative reasoning tasks. Furthermore, while we have provided access to our open dataset[5], we recognize the importance of improving automatic annotations through crowdsourcing to achieve more accurate results. Given that our automatic annotations sometimes exhibit issues in reasoning steps compared to manual annotations, and certain entries in the original dataset are ambiguous or erroneous, we emphasize the necessity of data cleanup to enhance mathematical reasoning in language models.

[5] COT Automatic and Manually annotated dataset

## References


Chung-Chi Chen, Jian-Tao Huang, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2024. Semeval-2024 task 7: Numeral-aware language understanding and generation. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

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

Keith Cortis, André Freitas, Tobias Daudert, Manuela Huerlimann, Manel Zarrouk, Siegfried Handschuh, and Brian Davis. 2017. SemEval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 519–535, Vancouver, Canada. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Yu Feng, Jing Zhang, Xiaokang Zhang, Lemao Liu, Cuiping Li, and Hong Chen. 2021. Injecting numerical reasoning skills into knowledge base question answering models. *arXiv preprint arXiv:2112.06109*.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. *arXiv preprint arXiv:2004.04487*.

Jian-Tao Huang, Chung-Chi Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2023. Numhg: A dataset for number-focused headline generation. *arXiv preprint arXiv:2309.01455*.

Maël Jullien, Marco Valentino, Hannah Frost, Paul O'regan, Donal Landers, and André Freitas. 2023. SemEval-2023 task 7: Multi-evidence natural language inference for clinical trial data. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 2216–2226, Toronto, Canada. Association for Computational Linguistics.

Jeonghwan Kim, Junmo Kang, Kyung-min Kim, Giwon Hong, and Sung-Hyon Myaeng. 2022. Exploiting numerical-contextual knowledge to improve numerical reasoning in question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1811–1821, Seattle, United States. Association for Computational Linguistics.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2024. Deductive verification of chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 36.

Aakanksha Naik, Abhilasha Ravichander, Carolyn Rose, and Eduard Hovy. 2019. Exploring numeracy in word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3374–3380.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Dhanasekar Sundararaman, Shijing Si, Vivek Subramanian, Guoyin Wang, Devamanyu Hazarika, and Lawrence Carin. 2020. Methods for numeracy-preserving word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4742–4753.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do nlp models know numbers? probing numeracy in embeddings. *arXiv preprint arXiv:1909.07940*.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating a math word problem to an expression tree. *arXiv preprint arXiv:1811.05632*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Jialiang Xu, Mengyu Zhou, Xinyi He, Shi Han, and Dongmei Zhang. 2022. Towards robust numerical question answering: Diagnosing numerical capabilities of nlp systems. *arXiv preprint arXiv:2211.07455*.

Peng-Jian Yang, Ying Ting Chen, Yuechan Chen, and Daniel Cer. 2021. Nt5?! training t5 to perform numerical reasoning. *arXiv preprint arXiv:2104.07307*.

## A Error Analysis

| | **Percentage Error Rate** | **Count Error Rate** |
|---|---|---|
| Addition | 46% | 42 |
| Copy | 4% | 148 |
| Divide | 36% | 4 |
| Multiply | 74% | 20 |
| Paraphrase | 4% | 16 |
| Round | 55% | 101 |
| Span | 7% | 1 |
| Subtract | 63% | 59 |
| Translation | 2% | 20 |

Table 5: The error rates are over the total amount of a given operation, not for the whole dataset. Note that Sround error rate is computed with the round operation.

## B Data Annotation

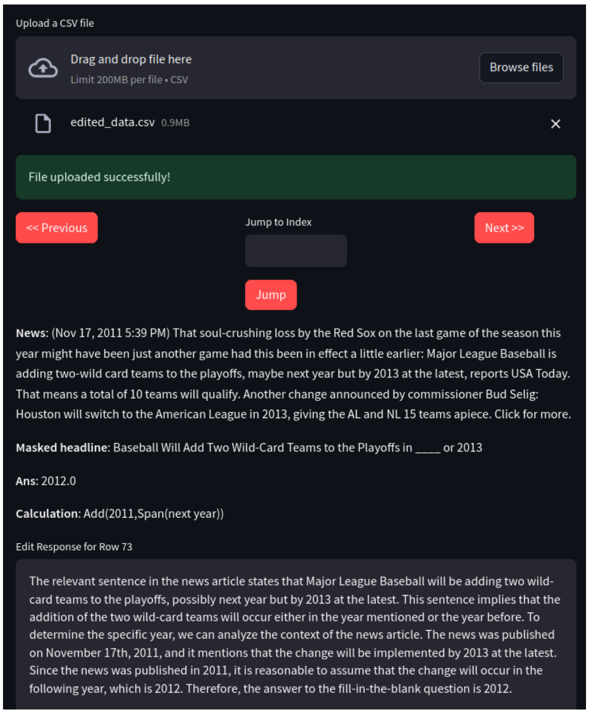


Figure 2: Example of our manual annotation system on streamlit

| Recipe | Function | Example | Operation |
|---|---|---|---|
| Copy | The simplest placeholder as the model simply takes the exact response taken from a given span in the passage | A union repping 2 million health care workers has made quite a find: 39 million N95 masks | *39* $\rightarrow$ 39 |
| Translation | Similarly to the copy placeholder, the answer is present in a given sentence. Thus, we state that the answer must be converted to its corresponding numerical value. | A University of Utah student paid his tuition bill with 2,000 one-dollar bills | *one* $\rightarrow$ 1 |
| Paraphrase | Involves paraphrasing a value that is appended in the headline by *K*, *M*, or *B*. That is to say, if the value is to be expressed in the thousands (K), millions (M), or billions (B), the numerical value found in the passage must be transformed accordingly. | A Florida travel insurance company has awarded a Georgia high school teacher \$10,000 | *\$10,000* $\rightarrow$ 10 |
| Round | Akin to paraphrase, round implies rounding a value to its nearest whole number or tenth depending on the specified decimal in the calculation. | Hackers made public the email addresses, usernames, and passwords of 790,724 Brazzers members. | *\$790,724* $\rightarrow$ 791 |
| Sround | Instead of approximating to the greater value, in *sround* the model must transform the value to its nearest floor value. | Today's after-hours bad news from the credit-crunch front comes from insurer AIG, which reported a fourth-quarter loss of \$5.29 billion | *\$5.29* $\rightarrow$ 5 |
| Span | It fetches the span in the given passage the headline is referring to | Brooklyn store owner Jacob Hamula could have ended up a victim of Salvatore Perrone, the suspected serial killer believed to have gunned down three other store owners before police nabbed him. | *Brooklyn store owner Jacob Hamula* $\rightarrow$ 1 |
| Subtract | It implements a heuristic whereby the model subtracts between the published date and the date mentioned in the passage as long as these dates are present in the metadata date and the article | (Apr 1, 2014 4:03 AM CDT) Steve Jobs did it; Google founders Sergey Brin and Larry Page did, too. Now Mark Zuckerberg is joining the ranks of the \$1-a-year CEOs, Bloomberg reports. That's what the Facebook boss earned in salary last year | *(Apr 1, 2014 4:03 AM CDT)* & *last year* $\rightarrow$ 2014 - 1 = 2013 |

Table 6: Placeholder functions used in our automatic annotation. Note that the round operation includes a paraphrase one in the given example. Additionally, the recipe for round and sround is virtually the same. Finally, the only subtract operations that were automatically annotated with this method involve dates. Otherwise, they are deemed as more "complex" operations that were manually annotated.

| Recipe | Pattern Example |
|---|---|
| Paraphrase | From the presence of "M" at the end of the fill-in-the-blank, we can infer that the blank in the question is asking for the value in millions. The sentence states that the population will be 308,400,408, so we need to convert this value to millions. To do this, we divide 308,400,408 by 1,000,000 which gives us 308.400. Since the question asks for the value in millions, we round down to the nearest whole number, which is 308. So the answer is 308. |
| Translation (transform dates) | The presence of both the apostrophe (') and "s" surrounding the blank strongly indicates that the number is abbreviated and pertains to a decade. Taking the example of the '80s, which covers the years 1980 to 1989, when individuals refer to "the '80s," they are typically referring to the complete decade. Since 1987 is within the timeframe of the '80s, it logically follows that the appropriate response is 80. So the answer is 80. |
| Translation (transform to ratio) | The term "quarter" refers to one part out of four equal parts. In the context of numbers or fractions, "1 in 4" is used to express the concept of a quarter. This means that when something is divided into four equal parts, you are referring to one of those parts. |

Table 7: Some example patterns used by the annotators to keep annotations consistent across some example tasks.