Beyond Words: Enhancing Reasoning in Entity Tracking

Kimberly Llajaruna, Emilia Mazzolenis, Michael Sam, and **Clara Ye**
{kllajaru, emazzole, msamchec, claraye}@mit.edu

Abstract

In the field of NLP, the tracking of evolving entities within textual data is of vital importance. While Named Entity Recognition (NER) locates and categorizes entities, Entity Tracking monitors the temporal evolution of these entities. This paper explores the impact of fine-tuning models equipped with reasoning capabilities on entity tracking. Leveraging a T5-base model, we evaluate the effects of fine-tuning via two distinct avenues: mathematical reasoning using mathematical question-answer pairs and computational reasoning with coding-related question-answer pairs. The study investigates the models' performances across individual datasets—general knowledge, code, and math—as well as their combinations. Results demonstrate that models trained on code-only datasets exhibit superior entity tracking capabilities compared to general knowledge-based models, while those trained on mathematical reasoning exhibit challenges due to out-of-vocabulary symbols. We also explore the influence of learning rates and dataset sizes on model performance, revealing optimal configurations for training. This research provides insights into enhancing entity tracking through reasoning-enabled models, emphasizing the need for diverse and extensive code-based datasets in NLP tasks.

1 Introduction

In the realm of natural language, entities—individuals, places, organizations, or objects we reference in our narratives (Jurafsky and Martin, 2020)—serve as the cornerstone of meaning, facilitating the transmission of information across generations. Such entities are far from static; they undergo continuous evolution, a concept inherently comprehensible to humans.

The ability to monitor the dynamic nature of entities over time holds significant relevance within NLP. While Named Entity Recognition (NER) is also a vital task that allows us to locate and categorize entities mentioned in a text (Sharma et al., 2022), entity tracking enables us to trace the progression of entity mentions within a textual corpus, providing invaluable insights into the temporal changes and narrative evolution (Toshniwal, 2022). The sentences below exemplify the importance of the task within NLP.

> *My dog Berta was born in Argentina 14 years ago and was as a playful puppy who loved the vibrant atmosphere of Latin America. Once in the US, Berta aced the relocation game, settling into her new home like a pro. Now, she's head over paws for her backyard—no nostalgia for her apartment days at all!*

The snippet above introduces the entity of a dog (Berta), and the usage of "Berta", "her", and "she" effectively tracks the continuity of the entity across different contexts. Through entity tracking, we emphasize Berta's relocation and subsequent adjustment, effectively capturing the evolving aspects of her life journey.

While entity tracking serves as a fundamental pillar in NLP, its complexity far exceeds the simplicity with which humans track entities. Models must accurately comprehend and connect diverse entity references throughout text to preserve coherence and context thereby supporting downstream tasks like summarization, machine translation, or question answering. However, entity tracking has not received as much extensive study as NER, likely due to its higher complexity in linking and maintaining connections between various mentions of entities, posing challenges that demand more sophisticated algorithms and annotated datasets. Moreover, while research has shown that the inclusion of intermediate chain of though prompting enables LLMs to engage in complex reasoning (Wei et al., 2022), the application of reasoning in entity tracking has not yet been widely studied. Thus, there is ample room for research to enhance model performance

in these tasks.

To comprehensively explore the impact of finetuning models equipped with reasoning capabilities, we aim to extend the existing literature. We specifically assess the effects of finetuning by integrating reasoning through two distinct avenues: mathematical reasoning— involving problems with mathematical questions and comprehensive step-by-step answers—, and computational reasoning—encompassing problems with coding-related questions and well-defined, step-by-step coding solutions. Employing a T5-base model (Raffel et al., 2020) across these scenarios, we establish a baseline by utilizing a model finetuned solely on factual information without any reasoning. Our evaluation criteria encompass accuracy and entity precision to gauge the performance of our models. The current work has the potential to inform the design of more effective models and enhance their performance across various applications.

2 Related Work

Entity Tracking: Traditionally, research emphasis was placed on developing complex model architectures to address entity-tracking tasks. Previous research has highlighted the significance of entity tracking in enhancing reading comprehension tasks, particularly when incorporating additional entity features and employing a multi-task tracking objective (Hoang et al., 2018). However, recent trends lean toward leveraging fine-tuning or pre-training techniques to enhance model performance. For instance, studies have revealed that mere fine-tuning with specialized question-answer formats and augmenting decoding mechanisms significantly boosts performance (Singh et al., 2023). Moreover, investigations into pre-trained models using both text and code have unveiled non-trivial entity-tracking behavior under a two-shot setting, a proficiency that models pre-trained solely on text demonstrate only after fine-tuning (Kim and Schuster, 2023). However, this body of research confronts limitations and the additional study of fine-tuning on similar data is needed to evaluate its findings.

Reasoning: In the realm of social sciences, it has been posited that learning loses its capacity for generalizability when it arises exclusively from mimicking rather than from reasoning (Bandura, 2008). In the NLP domain, incorporating reasoning abilities into language models has been shown to positively impact subsequent tasks like question answering (Rajani et al., 2019) and story completion (Chen et al., 2019). The incorporation can be quite simple, as merely the usage and improvement of chain of thought prompting can help lead to complex reasoning capabilities (Wei et al., 2022). Most current approaches to studying entity tracking, however, overlook the potential effect of reasoning to help in the task. Nevertheless, reasoning can help by providing a cognitive framework for understanding relationships and connections between entities across text. Existing work has shown that graph-based reasoning models improve the tracking of entities (Huang et al., 2021). Incorporating reasoning capabilities into entity tracking models could enable the recognition of implicit connections or references to entities, especially when they appear under varied linguistic forms or contexts. Incorporating reasoning capabilities in a different manner could, therefore, help models handle ambiguous references, making informed decisions by leveraging contextual information and world knowledge, thus, improving the entity tracking task.

Leveraging Code in Pretraining and Finetuning: Significant efforts have been dedicated to pretraining and finetuning models using code. For instance, CodeBERT, an encoder-only model, was specifically designed for natural language code search and code documentation generation (Feng et al., 2020), while CodeT5, an encoder-decoder transformer model, targeted code defect detection, clone detection, and code generation (Wang et al., 2021). The integration of pretraining with code has demonstrated value across various downstream applications, including editing tasks (Zhang et al., 2022), and has revealed additional insights about the source code and even natural language naming (Troshin and Chirkova, 2022). In the entity tracking domain, researchers have found that large models like GPT-3.5 pretrained on code can infer the evolving state of entities within text (Wei et al., 2022). Yet, a drawback in this research lies in the probable impact of differences in model architectures. The models studied, encompassing GPT-3 iterations and Flan-T5, diverge not only in their training data but also in their foundational structural designs. This potential discrepancy in entity-tracking capabilities might be ascribed as much to the model designs themselves as, if not more than, solely to their training data. Therefore, additional research is needed to confirm the paper's findings. Nevertheless, investigating the re-

lationship between code LLMs and entity tracking holds significance due to the potential synergy between understanding code-based information and tracking entities within text. Code LLMs possess inherent capabilities to comprehend and process programming languages, often requiring tracking and understanding entities like variables, functions, or objects across codebases.

Leveraging Math in Pretraining and Finetuning: Mathematical problems often require tracking how actions and relations change over time (Amini et al., 2019), which is a fundamental skill for entity tracking. Mathematical reasoning, encompassing linguistic, visual, practical, logical, numerical, and symbolic processes (Faldu et al., 2021), forms the bedrock of this comprehension. Previous studies have highlighted the efficacy of training models on question-answer pairs from mathematical problems, demonstrating strong performance and adaptability to novel mathematical tasks (Saxton et al., 2019). Despite the unexplored application of mathematical principles in entity tracking, ongoing research underscores the potential value of incorporating math in pretraining for such tasks.

3 Methodology

Our methodology is rooted in the framework established by researchers who evaluated the effect of pre-training on code to track the evolving states of entities in text (Kim and Schuster, 2023). To control for differences in model architecture and pre-training procedure, we fine-tuned a pre-trained T5 model (Raffel et al., 2020) on five different datasets and observed variations in performance on an entity-tracking task. The framework of the current study is outlined in Figure 1.

3.1 Datasets

To ensure consistency in our results regardless of the datasets' size, we deliberately selected the smallest dataset size (Coding Question-Answer Pairs) and maintained this uniform number across all three data categories by randomly selecting such a number of samples from the other two datasets. Consequently, our baseline question-answer pairs in factual information, coding and mathematics each comprise 1,887 observations for training (80%) and 472 for testing purposes (20%). The following five datasets were used:

Factual Question-Answer Pairs: The dataset originally comprises over 200,000 question-answer pairs sourced from the Jeopardy television game show. These pairs encompass questions spanning diverse topics like science, history, and pop culture, accompanied by their corresponding answers, typically one-word responses. An example of a question can be found in Figure 4. These pairs offer factual information without any accompanying reasoning.

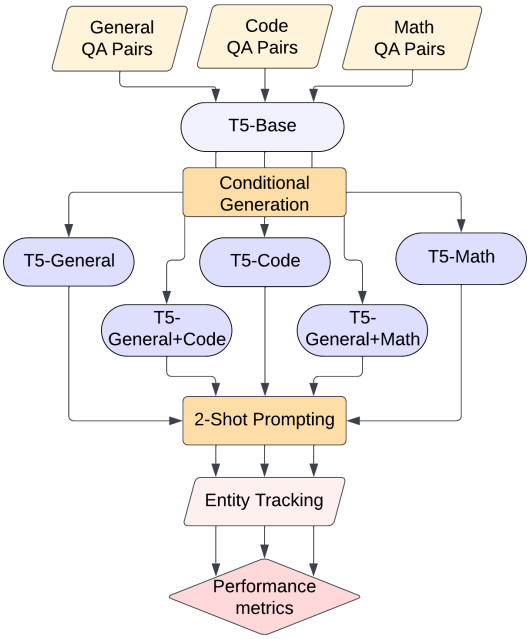

Figure 1: Diagram of our experiment procedure.

Coding Question-Answer Pairs: We employed a dataset featuring coding questions from LeetCode paired with their corresponding answers in Python. These answers encompass either pure code solutions or code accompanied by supplementary explanations in natural language. An example of a question can be found in Figure 5.

Mathematical Question-Answer Pairs: We utilized the Mathematics Aptitude Test of Heuristics dataset (Hendrycks et al., 2021). The dataset has detailed solutions to mathematical questions utilizing natural language explanations, mathematical facts, and mathematical expressions using LaTex. An example of a question can be found in Figure 6. The original dataset contained 7,500 train and 5,004 test samples.

To delve deeper into the implications of using different data sources, two novel datasets were derived from the original ones:

General+Code: We created a mix dataset of 50% Factual Question-Answer Pairs and 50% of Coding Question-Answer Pairs.

General+Math: We created a mix dataset of 50% Factual Question-Answer Pairs and 50% of Mathematical Question-Answer Pairs.

3.2 Models

We used a pre-trained T5-base model, a variant of the Transformer architecture (Raffel et al., 2023), as our baseline model. The T5-Base model, developed by Google Research, is known for its versatility in NLP tasks. The decision to use T5-Base is based on its success in capturing complex language patterns, its adaptability to various downstream tasks, and its small nature (Raffel et al., 2023).

Using the T5-base model as baseline, we fine-tuned five models on a conditional sequence generation task using the five datasets described in Section 3.1. The models are named after the dataset they were fine-tuned on.

3.3 Experiments

In addition to comparing the performance of models mentioned in Seciton 3.2, we experimented with different learning rates and dataset sizes.

3.3.1 Learning Rate

For each model, we experimented with three learning rate settings: 3e-3, 3e-4, 3e-5. By exploring multiple learning rates, we aim to decipher the ideal rate that not only expedites model convergence during training but also gives best performance across diverse datasets. This comprehensive exploration serves as a part of hyperparameter tuning, allowing us to identify the learning rate that optimally balances the trade-off between fast convergence and stable model behavior, thereby contributing to the model's adaptability in real-world scenarios.

3.3.2 Dataset Size

After observing the initial results, we conducted a follow-up experiment to test the effect of dataset sizes on performance and evaluate the extent to which our relatively small dataset sizes limit our study. Specifically, we additionally fine-tuned the T5-General, T5-Code, and T5-Math models using 500 training examples only, and compared their performance with those trained with all 1,887 training examples.

3.4 Evaluation

In the original work by Kim and Schuster (Kim and Schuster, 2023), the authors employed a 2-shot training method with the following prompt for T5 models:

> Given the description after "Description:", write a true statement about a box and the contents of this box according to the description after "Statement:".
>
> **Description:** Box 0 contains the car, Box 1 contains the cross, Box 2 contains the bag and the machine, Box 3 contains the paper and the string, Box 4 contains the bill, Box 5 contains the apple and the cash and the glass, Box 6 contains the bottle and the map.
> **Statement:** Box 0 contains the car, Box 1 contains the cross, Box 2 contains the bag and the machine, Box 3 contains the paper and the string, Box 4 contains the bill, Box 5 contains the apple and the cash and the glass, Box 6 contains the bottle and the map.
>
> **Description:** Box 0 contains the car, Box 1 contains the cross, Box 2 contains the bag and the machine, Box 3 contains the paper and the string, Box 4 contains the bill, Box 5 contains the apple and the cash and the glass, Box 6 contains the bottle and the map. Remove the car from Box 0. Remove the paper and the string from Box 3. Put the plane into Box 0. Move the map from Box 6 to Box 2. Remove the bill from Box 4. Put the coat into Box 3.
> **Statement:** Box 0 contains the plane, Box 1 contains the cross, Box 2 contains the bag and the machine and the map, Box 3 contains the coat, Box 4 contains nothing, Box 5 contains the apple and the cash and the glass, Box 6 contains the bottle.
>
> **Description:** `{description}`
> **Statement:** Box `{boxnum}` contains

In our approach, inspired by the mentioned work, we adopted a similar 2-shot prompting strategy. However, instead of passing in the full prompt as a long string, we used the following shorter prompt for each sample:

> Given the description after "Description:", complete the last sentence with a true statement about the contents of the specified box according to the description.
>
> **Description:** `{description}`
> **Statement:** Box `{boxnum}` contains

and we used the two examples from the original prompt to train the models under supervision before evaluating them on the test set. We adopted this approach to ensure that the models fully grasp the task requirement, and fairly evaluate their entity tracking ability while minimizing the influence from their ability to transfer between tasks or understand long sentences.

4 Implementation

4.1 Fine-tuning Procedure

4.1.1 Tokenization

The datasets, including those for math, general knowledge, and code, were divided into training and testing sets. For tokenization, we leveraged the pre-trained tokenizer provided by the "t5-base" model, which converts the sequence of text into a sequence of tokens that the model can take.

4.1.2 Preprocessing

To prepare the data for fine-tuning, we created a preprocessing function. This function added a prefix such that the sentences are of the same size, tokenized the text using the T5-Base tokenizer, and set the labels accordingly. The tokenized answers were crucial input features for the downstream task, ensuring compatibility with the T5-Base model's text-to-text format.

4.1.3 Fine-tuning Details

We fine-tuned the T5 model on the previously mentioned math and coding datasets for the sequence generation task and a cross-entropy loss function. Unless otherwise noted, we employed the following specifications: a learning rate of 3×10^{-5}, a batch size of 8, an evaluation batch size of 4, a weight decay of 0.01, and a total of 3 training epochs.

4.2 Evaluation

4.2.1 Details for Two-Shot Training

We trained each of the models on a conditional generation task using two supervised examples as mentioned in section 3 before evaluating them on the test set. We trained them for 3 epochs, with a batch size of 8 and a weight decay of 0.01. We used a cross-entropy loss between the generated sequence and the gold answer, and we used an Adam optimizer with parameters of a learning rate of 10^{-4} and epsilon of 10^{-8}.

4.2.2 Metrics

For each dataset (general knowledge, code, and math), the model's performance was assessed using the standard NLP evaluation metrics of accuracy, precision, recall, and F1 score. Note that when computing accuracy, we treated a box as a unit of observation and we count a prediction as correct if and only if all items in the reference solution matches all items in the predictions, regardless of item order. When computing precision, recall, and F1 score, we treated an item as a unit of observation. An item is counted as a true positive if it appears in both the reference solution and the prediction, a false positive if it appears in the prediction only, and a false negative if it appears in the reference solution only.

5 Results and Analysis

The metrics across all evaluated models can be found in Table 1.

5.1 Main Results

The results table distinctly showcases the model's performances across various combinations. The code-based model exhibits superior entity tracking performance compared to the general knowledge-based model, showcasing enhancements in accuracy, recall, and F1 score while maintaining a comparable precision level. Conversely, the mathematical model displays a marginal decline in performance across all metrics. The model combining 50% mathematical reasoning with 50% general knowledge demonstrates performance superior to the mathematical model alone but falls short of the baseline. However, the model blending general knowledge and code presents a similar accuracy to the baseline but a decrease compared to the code-only model.

5.1.1 Interpretation of main results

The decline observed in the mathematical model's performance might be attributed to the presence of LaTeX symbols, unfamiliar during training and potentially interpreted as out-of-vocabulary words. In contrast, the code-only model demonstrates improved performance, leveraging its reasoning capabilities without a significant prevalence of out-of-vocabulary words, likely due to code characters being regular characters in its training data. The model blending 50% mathematical reasoning with 50% general knowledge performs better than the mathematical model alone but falls short of the baseline, indicating the impact of incorporating textual information without LaTeX symbols. This aligns with the hypothesis regarding the influence of out-of-vocabulary words. However, the model combining general knowledge and code exhibits similar accuracy to the baseline but decreases compared to the code-only model. This outcome might stem from the reduced emphasis on reasoning-heavy code-related questions, leading to a decline

Model	Learning Rate	Accuracy	Precision	Recall	F1 Score
General	**3e-3**	0.01	0.01	0.01	0.00
	3e-4	0.14	0.42	0.40	0.20
	3e-5	0.15	**0.43**	0.39	0.20
Code	**3e-3**	0.00	0.00	0.00	0.00
	3e-4	0.03	0.11	0.11	0.05
	3e-5	**0.18**	0.41	**0.44**	**0.21**
Math	**3e-3**	0.00	0.01	0.01	0.00
	3e-4	0.02	0.08	0.09	0.04
	3e-5	0.12	0.37	0.33	0.18
General+Code	**3e-3**	0.00	0.03	0.02	0.01
	3e-4	0.03	0.10	0.09	0.05
	3e-5	0.14	0.42	0.39	0.20
General+Math	**3e-3**	0.00	0.00	0.00	0.00
	3e-4	0.10	0.34	0.37	0.18
	3e-5	0.14	0.41	0.35	0.19

Table 1: Performance metrics of models with different learning rates. Accuracy is computed box-wise while other metrics are computed item-wise.

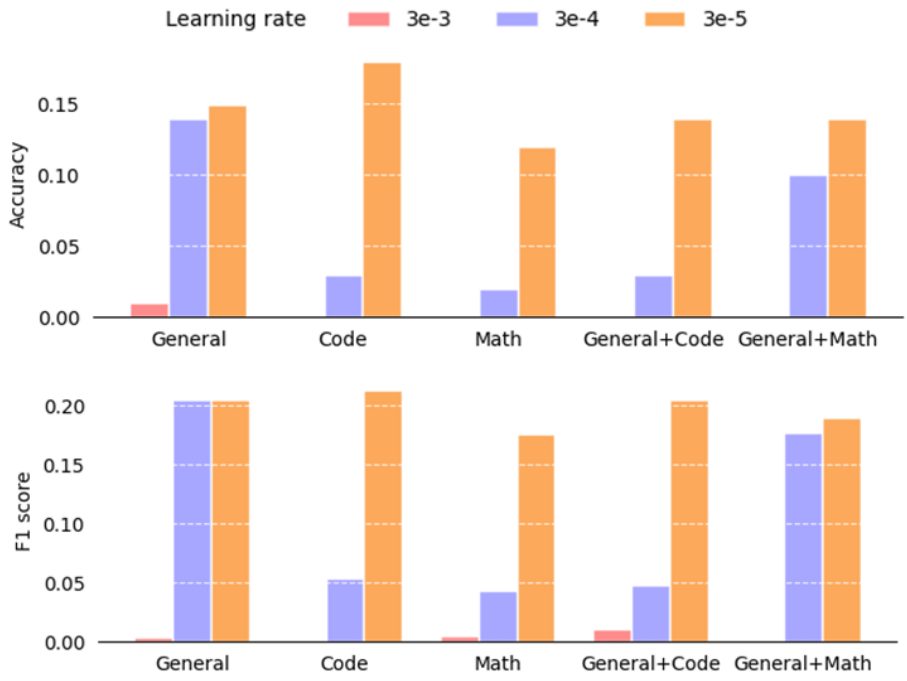

Figure 2: Accuracy and F1 score of models with different learning rates.

in reasoning capability compared to the code-only variant.

5.2 Results Across Different Learning Rates

In the experiment, employing a high learning rate of 3e-3 resulted in extremely poor performance across all datasets. Using 3e-4 and 3e-5 yielded improved and comparable results, particularly in the general and general+math scenarios. However, an overall learning rate of 3e-5 proved to be the most effective for the majority of models. This indicates that lower learning rates were advantageous for both code and math models.

5.2.1 Interpretation of the Results Across Different Learning Rates

The difference in performance across learning rates could be explained by the theory. The decline in performance observed with a high learning rate of 3e-3 might be attributed to the models overshooting optimal parameter values and failing to converge effectively during training. On the other hand, lower learning rates often facilitate more stable and incremental learning, allowing models to gradually fine-tune their parameters without overshooting optimal values or getting stuck in suboptimal local minima. This is especially crucial for code and mathematical reasoning tasks that might require intricate, nuanced learning patterns.

5.3 Results Across Finetuning Dataset Size

As seen in Figure 3, accuracy improves with a larger finetuning dataset for the general knowledge and the code-only models. However, accuracy decreases with a larger finetuning dataset for the math-only model.

5.3.1 Interpretation of Results Across Finetuning Dataset Size

For the general knowledge-only and code-only models, a larger dataset correlates with an improvement in accuracy. This trend aligns with the notion that a more extensive and diverse dataset often leads to better model performance, enabling models to capture a broader range of patterns and nuances. The observed decrease in accuracy for the math-only model with a larger finetuning dataset might be attributed to the presence of out-of-vocabulary LaTeX symbols. This decline suggests that the inclusion of additional data, particularly with LaTeX symbols unfamiliar during model training, could have introduced complexities or variations challenging the model's effective understanding of the mathematical reasoning domain. The OOV LaTeX symbols may have disrupted the model's ability to generalize effectively, contributing to the decline in accuracy despite the increased dataset size.

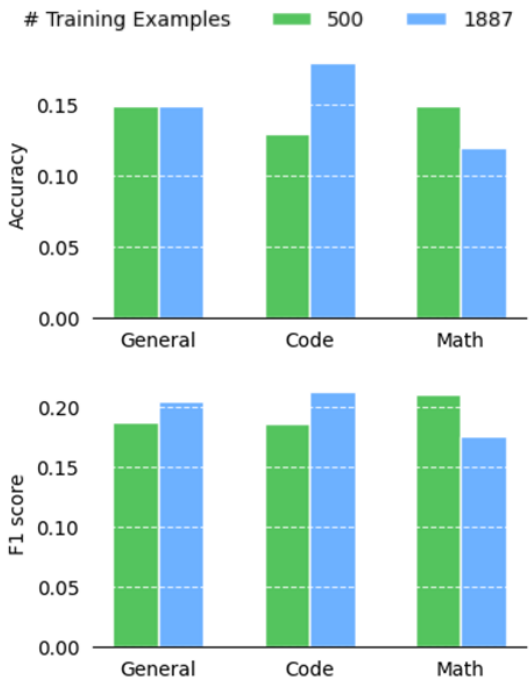

Figure 3: Accuracy and F1 score of models with different training dataset sizes.

5.4 Discussion

Overall, none of the models produce promising results - there are only a few item-level matches, and none of the full predictions match the correct answer. We attribute the current performance partially to the difficulty of the training task. Some of the code and math solutions are long and contain many out-of-vocabulary words such as special characters in code and latex commands in math, potentially preventing the models from generating sequences that are comparable to the correct answers and benefit from gradient updates.

Since our findings were replicated in our different dataset size experiment, the results of the study likely generalize to similar settings, and we expect the performance of T5-Base models fine-tuned on code to be better on entity tracking tasks than a baseline. However, the results are not generalizable to other model architectures, and additional research is needed to study the effect of fine-tuning with reasoning-heavy data on entity tracking.

Model	# Examples	Accuracy	Precision	Recall	F1 Score
General	**500**	0.15	0.39	0.37	0.19
	1887	0.15	0.43	0.39	0.20
Code	**500**	0.13	0.40	0.35	0.19
	1887	0.18	0.41	0.44	0.21
Math	**500**	0.15	0.42	0.42	0.21
	1887	0.12	0.27	0.33	0.18

Table 2: Performance metrics of models with different training dataset sizes. Accuracy is computed box-wise while other metrics are computed item-wise. All models use learning rate 3e-5 during training.

5.5 Limitations

Throughout this project, we had several noteworthy limitations, with one of the most prominent being the constraints imposed by the relatively modest size of our fine-tuning dataset. Specifically, we employed datasets comprised of only 1887 observations for training the model. This restriction, while necessary due to resource/time limitations, had implications for the model's performance.

One of the key areas where this limitation became apparent was in dealing with out-of-vocabulary words. These are words and terms that the model hadn't encountered during its training phase, and as a result, it struggled to generate accurate predictions or contextually relevant responses when such words were introduced. A larger training dataset would have provided a more extensive vocabulary for the model to learn from, potentially enabling it to better handle OOV words by drawing upon a wider range of contextual information.

Another aspect that could have benefited from a larger dataset is the understanding and interpretation of Latex syntax. Latex is a complex and specialized markup language commonly used for typesetting mathematical and scientific documents. Training the model to understand and generate Latex effectively requires exposure to a diverse set of Latex expressions and patterns. With a more extensive dataset, the model could have had access to a richer variety of Latex syntax examples, allowing it to improve its ability to work with mathematical notations, equations, and scientific text.

Yet another constraint we encountered pertained to the scarcity of computational resources at our disposal for running the models. Our model execution took place on Google Colab, where we had to work within the confines of the available GPUs.

Finally, our decision to use the T5 model, influenced by computational limitations, presented a notable trade-off. While T5's versatility in language tasks like summarization and reading comprehension is commendable, it does not reach the sophistication of newer models like Llama 2 (Touvron et al., 2023) or GPT-4 (OpenAI, 2023). T5's architecture is robust, yet it lacks the advanced techniques and scalability of Llama 2. Llama 2, notable for its efficiency and Ghost Attention technique, excels in dialogue control but does not match the creative prowess of GPT-4, which stands out in complex language generation. This compromise underlines our project's constraints, balancing the need for a fine-tunable model within our resources against the advanced capabilities of more recent models.

5.6 Future work

In the next phase of our research, we wish to enrich the training dataset by incorporating a broader range of data, specifically targeting out-of-vocabulary words frequently encountered in Latex and code. This enhancement is expected to improve the model's performance in handling specialized terminology and complex expressions.

In future endeavors, we aim to employ more sophisticated models than T5, since they likely have enhanced capabilities for comprehending the intricacies of Latex and code syntax.

6 Conclusion

Our project has made advancements in the field of Natural Language Processing, specifically in the domain of entity recognition. We discovered that incorporating computational reasoning into the process of entity tracking enhances the effectiveness of our models, while Latex-heavy mathematical reasoning does not. The T5-base model played a crucial role in setting the foundation for our research. However, our analysis also brought to light its limitations, particularly in its ability to process the complex syntax of Latex and coding languages. These insights serve as a stepping stone for future research, underlining the importance of employing more advanced models capable of comprehending the nuances of such specialized languages more effectively.

References

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Albert Bandura. 2008. Observational learning. *The international encyclopedia of communication*.

Jiaao Chen, Jianshu Chen, and Zhou Yu. 2019. Incorporating structured commonsense knowledge in story completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6244–6251.

Keyur Faldu, Amit Sheth, Prashant Kikani, Manas Gaur, and Aditi Avasthi. 2021. Towards tractable mathematical reasoning: Challenges, strategies, and opportunities for solving math word problems. *arXiv preprint arXiv:2111.05364*.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Luong Hoang, Sam Wiseman, and Alexander M Rush. 2018. Entity tracking improves cloze-style reading comprehension. *arXiv preprint arXiv:1810.02891*.

Hao Huang, Xiubo Geng, Pei Jian, Guodong Long, and Daxin Jiang. 2021. Reasoning over entity-action-location graph for procedural text understanding. In *ACL-IJCNLP 2021-59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*.

Daniel Jurafsky and James H Martin. 2020. Sequence labeling for parts of speech and named entities. *Speech and Language Processing*, 71.

Najoung Kim and Sebastian Schuster. 2023. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3835–3855, Toronto, Canada. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*.

Abhishek Sharma, Amrita, Sudeshna Chakraborty, and Shivam Kumar. 2022. Named entity recognition in natural language processing: A systematic review. In *Proceedings of Second Doctoral Symposium on Computational Intelligence: DoSCI 2021*, pages 817–828. Springer.

Janvijay Singh, Fan Bai, and Zhen Wang. 2023. Entity tracking via effective use of multi-task learning model and mention-guided decoding. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1247–1255.

Shubham Toshniwal. 2022. Efficient and interpretable neural models for entity tracking. *arXiv preprint arXiv:2208.14252*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Sergey Troshin and Nadezhda Chirkova. 2022. Probing pretrained models of source code. *arXiv preprint arXiv:2202.08975*.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Jiyang Zhang, Sheena Panthaplackel, Pengyu Nie, Junyi Jessy Li, and Milos Gligoric. 2022. Coditt5: Pretraining for source code and natural language editing. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12.

Impact Statement

Entity tracking spans various domains and holds immense potential to influence lives profoundly. Its application across various downstream tasks, such as machine translation, information extraction, text summarization, and question-answering, carries substantial implications for individuals and societies. Improvements in entity tracking can lead to better communication and less ambiguity, better information retrieval, more contextual understanding, better semantic analysis, and even better personalization and recommendation systems as better tracked entities can lead to better suggestions for users based on their preferences.

Within machine translation, for instance, entity tracking plays a pivotal role—it can help bridge barriers and enhance intercultural communication, but it has the potential to disrupt communication with severe repercussions if misinterpreted. Consider the sentence: *'I told the president of Valdovia that if his General of the Army does not reach an agreement with the president of Springfield, he will declare war.'* Inaccurate entity tracking might misinterpret the pronoun 'he' in 'he will declare a war,' attributing it wrongly to either president, leading to a catastrophic misunderstanding. A mistranslation in the language of Springfield could erroneously suggest to the Springfieldian president that the Valdovian president intends to declare war against him, triggering conflict due to a translation error guided by entity tracking mistakes. Our research, however, can help advance the field of entity tracking, especially by fine-tuning models with code only Q&A data, and make such critical misinterpretations less likely to occur.

While our models are not intended for direct application, our study may have positive societal impact by providing guidance for future development of models with better reasoning abilities. Specifically, as our study provided additional evidence to the conclusion that models trained on code tend to have better entity-tracking ability, one may tailor the training data used accordingly during model development if such reasoning ability is important in their applications. As entity-tracking is a critical component of advanced natural language understanding, we expect our study conclusion to contribute indirectly to a variety of application domains.

Even though enhanced entity tracking has a wide variety of positive applications, its potential misuse poses societal risks. Our research shows that even fine-tuning with code alone can enhance this task, making it an easily accessible technique, thus increasing the possibility of widespread application and, consequently, misuse. Enhancing entity tracking could enable more precise identification and profiling of individuals, potentially leading to increased surveillance or invasions of privacy. Moreover, more accurate tracking and profiling could be used in misinformation campaigns, where more accurate tracking might be exploited to create and spread false narratives or targeted propaganda, leading to societal unrest. Additionally, better entity tracking might inadvertently reinforce biases present in the data used for training, perpetuating existing societal inequalities or stereotypes.

In terms of the current study, we used datasets containing general facts or constructed toy problems. None of the datasets contain sensitive information by design, ensuring a commitment to data privacy and security. In terms of bias and fairness considerations, our approach involved fine-tuning a pre-trained language model, specifically T5, on the aforementioned datasets. While we did not implement specific measures targeting bias mitigation, we do not expect the datasets or tasks to introduce any additional bias or fairness issue to the models. Regarding accountability, we also do not expect our models to be used in critical applications as they were employed to validate academic hypotheses only.

In conclusion, the evolution of entity tracking presents a double-edged sword—holding immense potential to enhance communication and understanding across various domains while simultaneously creating risks of societal implications if misused. Our research strives to navigate this balance, contributing to the advancement of entity tracking techniques while advocating for responsible deployment.

Appendix

Question: Until the 18th century, European treaties were generally written in this language.
Answer: Latin

Figure 4: Example of a general knowledge question-answer pair.

Question: You are participating in an online chess tournament. There is a chess round that starts every '15' minutes. The first round of the day starts at '00:00', and after every '15' minutes, a new round starts.
For example, the second round starts at '00:15', the fourth round starts at '00:45', and the seventh round starts at '01:30' You are given two strings 'loginTime' and 'logoutTime' where:* 'loginTime' is the time you will login to the game, and* 'logoutTime' is the time you will logout from the game. If 'logoutTime' is **earlier** than 'loginTime', this means you have played from 'loginTime' to midnight and from midnight to 'logoutTime'.
Return the number of full chess rounds you have played in the tournament.
Note: All the given times follow the 24-hour clock. That means the first round of the day starts at '00:00' and the last round of the day starts at '23:45'.
Answer:

```
def second_largest_digit(s: str) -> int:
    largest = -1
    second_largest = -1
    for c in s:
        if c.isdigit():
            digit = int(c)
            if digit > largest:
                second_largest = largest
                largest = digit
            elif digit != largest and
                digit > second_largest:
                second_largest = digit
    return second_largest
```

Figure 5: Example of a code question-answer pair.

Question: What is the number of units in the distance between $(2, 5)$ and $(-6, -1)$?
Answer: We use the distance formula: $\sqrt{(-6-2)^2 + (-1-5)^2}$, so then we find that $\sqrt{64+36} = 10$.
- OR -
We note that the points $(2, 5)$, $(-6, -1)$, and $(2, -1)$ form a right triangle with legs of length 6 and 8. This is a Pythagorean triple, so the length of the hypotenuse must be 10.

Figure 6: Example of a math question-answer pair.