SELF-TEACHING PROMPTING FOR MULTI-INTENT LEARNING WITH LIMITED SUPERVISION

Cheng Chen[1,2] **Ivor W Tsang**[1,2,3,4]
[1]*The Australian Artificial Intelligence Institute (AAII), University of Technology Sydney, Australia*
[2]*Centre for Frontier AI Research, A*STAR, Singapore.* [3]*Institute of High Performance Computing, A*STAR, Singapore*
[4]*School of Computer Science and Engineering, Nanyang Technological University, Singapore*
{Cheng.chen-16@student.uts.edu.au}, {ivor_tsang@cfar.a-star.edu.sg}

ABSTRACT

Multi-intent learning with limited supervision involves predicting multiple intentions of utterances using only a few annotated samples. The primary motivation for this task stems from the high costs and cumbersome processes associated with annotating large datasets. To mitigate this, we propose utilising Large Language Models (LLMs) for annotation assistance. Although LLMs show promise, they struggle with response randomness, and their previous prompts is static and do not learn from their outputs. To address this, we propose 'self-teaching prompting' (STP), a method that enables Large Language Models (LLMs) to iteratively learn from their consistent samples and refine their predictions over time. Our experiments with multi-intention datasets demonstrate that STP significantly enhances response accuracy.

1 INTRODUCTION

Spoken Language Understanding (SLU) (Young et al., 2013), a key function in spoken dialogue systems, is primarily aimed at accurate intent classification (Tur & De Mori, 2011). The success of multi-intent detection, as explored in various studies (Gangadharaiah & Narayanaswamy, 2019; Kim et al., 2017; Qin et al., 2021; Xing & Tsang, 2022; Veličković et al., 2017), has relied on traditional methods of obtaining clean annotations, such as crowdsourcing and manual annotation. Nonetheless, these methods are becoming increasingly inadequate due to the exponential growth of data samples. To mitigate this, we propose using Large Language Models (LLMs) such as ChatGPT (Devlin et al., 2018; Touvron et al., 2023; OpenAI, 2023) for annotating unlabelled samples. However, they often produce random and inconsistent outputs. As a result, prompt-based learning has emerged (Brown et al., 2020; Wei et al., 2021; Yao et al., 2022; Diao et al., 2023; Liu et al., 2023). Nonetheless, these methods are limited to 'static prompting', which does not allow LLMs to learn from their responses, resulting in sub-optimal response accuracy. This issue is particularly detrimental in fields requiring high-quality annotations, such as recommendation system, medicine and legal text mining. Subsequently, we propose a *Self-Teaching Prompting*(STP) approach, enabling LLMs to iteratively improve the accuracy and consistency of responses.

2 PROBLEM STATEMENT

We define an utterance space vector as $x \in X$, and the feature space is $X \subseteq \mathbb{R}^d$, where d denotes the utterance length. We denote the label space as $\mathcal{Y} = [k]$, where $[k] = \{1, 2, 3, \dots, k\}$, and $k > 2$ represents the size of the intents class. In our problem setting, an unsupervised distribution $D_X = \{x_1, x_2, \dots, x_n\}$ over the input space is given. We use ChatGPT, denoted as G_r, for the prompting task, where r, a temperature parameter, controls the level of randomness in generating label predictions for each input sample x. We denote a clean utterance space vector as $a \in X$, and its corresponding label as $t \in T$, where the label space T is defined as $T = [k]$, with $[k]$ being the set of class labels $1, 2, 3, \dots, k$. Additionally, we consider a small, clean dataset $D_{AT} = \{(a_1, t_1), (a_2, t_2), \dots, (a_s, t_s)\}$ over the instance and label spaces, where s represents the total number of instances in the clean dataset. In the MixATIS dataset, $s = 70$, and in the MixSNIPS dataset, $s = 400$. The true label set T, defined as $T = [k]$, where $[k]$ is the set of class labels $\{1, 2, 3, \dots, k\}$.

3 Self-Teaching Prompting

Self-teaching prompting is designed to solve issues of response randomness, the static nature of previous prompts, and lack of learning from outputs.

Hints: Hints are used to solve the issue of response randomness. Initially, we select an example with the highest contextual similarity score to utterance x from a small set of clean samples as a hint to help reduce the response randomness in $ChatGPT3.5$.

Consistent-Sample as Additional Hints: Additional hints enable LLMs to learn more reliable knowledge from consistent sample. G_r generates the predicted label sets $\vec{Y}_r$, where r is the temperature parameter controlling the randomness level in label prediction for each input sample $x \in X$. Specifically, for each x_i, $G_r(x_i, a_{top}, t_{top}) = \vec{Y}_{t_i}$, where a_{top} and t_{top} are the example and its true intent with the highest embedding score with x_i chose from the hint. This holds for all $i \in [N] = \{1, 2, 3, \ldots, N\}$, where N is the total number of training sample and $\forall r \in \{0.1, 0.3, 0.5, 0.7\}$. Consequently, four predicted sample distributions with varying temperature parameters r are obtained, denoted as $D_r = \{(x_i, \vec{Y}_{r,i}) | x_i \in X\}$. The selection criterion for the consistent distribution defined as $D_c = \{(x_i, \vec{Y}_{c_i}) | x_i \in X$ and $\vec{Y}_{r=0.1,i} = \vec{Y}_{r=0.3,i} = \vec{Y}_{r=0.5,i} = \vec{Y}_{r=0.7,i}\}$, including only instances with matching prediction label sets across all four temperature-based configurations G_r.

Self-Teaching Prompting: Self-Teaching Prompt solves the static nature of the Prompts. For the first epoch, consistent-sample process requires four repetitions and each repetition is using a different randomness parameter r of the ChatGPT G_r. Subsequently, a consistent sample D_c is chosen from those with identical prediction sets. The selected consistent sample is then merged with the initial hints (a small set of clean samples), which is then exploited in the next prediction round. This procedure is repeated for a total of three epochs. The algorithm table A.0.1 is presented in appendix.

4 Experiments

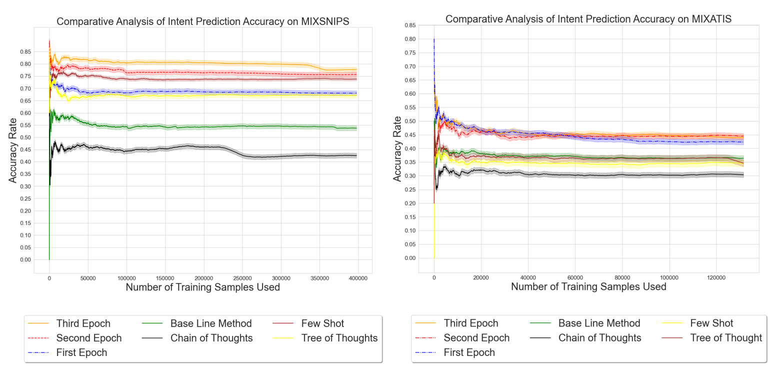

Figure 1: Multi-Intent Accuracy/Matching Comparison of our Self-Teaching and Chain of Thought Prompt, Random-CoT Prompt (Our Baseline Method), Tree of Thought, Few Shots. Accuracy/Matching Rate is defined as (Correct Predictions/Number of Samples Used).

In Table 1 (A.3) of the Appendix, we have shown a comparison of our methods with the Chain of Thought Prompt and Random-CoT Prompt (Our Baseline Method) to verify the effectiveness of the method. The left-hand side line chart in Figure 1 demonstrates the intent accuracy rate for MIXSNIP, while the right-hand side shows the intent accuracy rate for MIXATIS.

5 Conclusion

This paper presents a new paradigm for iteratively updating prompts by using their own generated responses in a large language model for multi-intent learning with limited supervision. Our results demonstrate that learning how to self-correct prompts can be valuable and significantly improve performance in intent detection tasks.

URM Statement

The authors acknowledge that at least one key author meets the URM criteria for ICLR 2024 Tiny Papers Track.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, 2018.

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

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. Active prompting with chain-of-thought for large language models, 2023.

Rashmi Gangadharaiah and Balakrishnan Narayanaswamy. Joint multiple intent detection and slot labeling for goal-oriented dialog. 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)*, pp. 564–569, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1055. URL https://aclanthology.org/N19-1055.

Charles T Hemphill, John J Godfrey, and George R Doddington. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.

Byeongchang Kim, Seonghan Ryu, and Gary Geunbae Lee. Two-stage multi-intent detection for spoken language understanding. *Multimedia Tools and Applications*, 76:11377–11390, 2017.

Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, 2023.

Jieyi Long. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Ankan Mullick, Abhilash Nandy, Manav Nitin Kapadnis, Sohan Patnaik, and R Raghav. Fine-grained intent classification in the legal domain. *arXiv preprint arXiv:2205.03509*, 2022a.

Ankan Mullick, Abhilash Nandy, Manav Nitin Kapadnis, Sohan Patnaik, R Raghav, and Roshni Kar. An evaluation framework for legal document summarization. *arXiv preprint arXiv:2205.08478*, 2022b.

Ankan Mullick, Ishani Mondal, Sourjyadip Ray, R Raghav, G Sai Chaitanya, and Pawan Goyal. Intent identification and entity extraction for healthcare queries in indic languages. *arXiv preprint arXiv:2302.09685*, 2023.

OpenAI. Gpt-4 technical report, 2023.

Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. Agif: An adaptive graph-interactive framework for joint multiple intent detection and slot filling. *arXiv preprint arXiv:2004.10087*, 2020.

Libo Qin, Fuxuan Wei, Tianbao Xie, Xiao Xu, Wanxiang Che, and Ting Liu. GL-GIN: Fast and accurate non-autoregressive model for joint multiple intent detection and slot filling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 178–188, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.15. URL `https://aclanthology.org/2021.acl-long.15`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Gokhan Tur and Renato De Mori. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons, 2011.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

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

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

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

Bowen Xing and Ivor W Tsang. Co-guiding net: Achieving mutual guidances between multiple intent detection and slot filling via heterogeneous semantics-label graphs. *arXiv preprint arXiv:2210.10375*, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

A APPENDIX

A.0.1 ALGORITHM TABLE

Algorithm 1: Self-Teaching Prompt

Data: Initial small set of clean samples $D_{e,A,T}$ (Initial Hints), where $e = 0$. Training distribution of D_X
Learning Objective: Refined Prediction label set of samples from ChatGPT
Initialisation: Loading Initial small set of clean samples $D_{e,A,T}$, where $e = 0$;
Randomness level: $R = \{0.1, 0.3, 0.5, 0.7\}$;
for $e \in \{0, 1, 2\}$ **do**
  for $r \in R$ **do**
    for *Each x in Training Distribution of $D(X)$* **do**
      Compute cosine similarity 1 between x and each a in the hints using lower embedding extracted using $Word2vec$ model;
      Choose example $(a_{\text{TOP}}, t_{\text{TOP}})$[1]from $D_e(A, T)$, based on highest cosine similarity score with x to be fed into $G_r(x, a, t)$;
      Select consistent samples D_{ec}, where e is the e-th epoch, with the same prediction set across $G_r(x, a, t)$, $\forall r \in R$;
    end
  end
  `/* Update the hints for the next epoch */`
  $D_{e+1,c}(A, T, X_c, \vec{Y_c}) = D_{e,A,T}(A, T) \cup D_{ec}(X_c, \vec{Y_c})$ `/* The updated hints now consist of initial small set of clean sample and new consistent samples of first epoch */`
end

A.0.2 SIMILARITY SCORE

$$\text{Similarity} = \frac{W(\text{a}) \cdot W(\text{x})}{\|W(\text{a})\| \|W(\text{x})\|} \tag{1}$$

where W stands for the word2vec model (Mikolov et al., 2013). It produces the lower embedding of the example sentence a and input question x. It has been first trained on whole sentences datasets X. The cosine similarity score is used to evaluate the word embedding similarity between the Question x and Sample a. The positive example is determined as following:

$$a_{\text{top}}, t_{\text{top}} = \underset{(a_i, t_i) \in D_{\text{e},(A,T)}}{\arg\max} \quad \text{Similarity}(a_i, x) \tag{2}$$

A.0.3 EVALUATION METRICS FOR PROMPTING

The matching/accuracy ratio is designed to measure the exact matching rate between the predicted label set and the tru label set. It is denoted as:

$$\text{Matching/Accuracy Ratio} = \frac{\text{Number of correctly predicted labels}}{\text{Total number of samples}} \tag{3}$$

The Subset Ratio evaluate the ratio of the predicted label set that includes the true label set. The subset ratio is only used in dealing with multi-intents classification task. It is defined as:

$$\text{Subset Ratio} = \frac{\text{Number of predicted label sets that includes true labels}}{\text{Total number of samples}} \tag{4}$$

These metrics is used to measure the performance of our proposed prompting method in terms of intent prediction accuracy and the subset intent prediction.

[1]In the implementation, the cosine similarity score is used to select top positive samples, with a single top sample chosen in the first epoch and top-ranked samples chosen in subsequent epochs for datasets MixATIS and MixSNIPS.

A.0.4 EXPERIMENT

A.1 DATASETS

The experiments are implemented on two open source multi-intents datasets. One of the datasets is MixATIS (Hemphill et al., 1990; Qin et al., 2020), which consists of **13162** utterances for training, 756 utterances for validation and 828 utterances for testing. The other one is MixSNIPS (Coucke et al., 2018; Qin et al., 2020), with **39776**, 2198 and 2, 199 utterances for training, validation and testing dataset.

A.2 ADDITIONAL DATASETS FOR GENERALIZABILITY OF SELF-TEACHING PROMPTING (STP)

In our study, we have enhanced the scope of testing the generalizability of our Self-Teaching Prompting (STP) method by incorporating multi-lingual and multi-domain additional datasets. Specifically, we included the legal dataset, as detailed in Mullick et al. (2022a;b), along with the medical dataset from Mullick et al. (2023). The legal dataset is comprehensive, comprising 1,385 training samples across four distinct intents. Furthermore, we expanded our experiment to add Healthcare datasets, namely the Indian Healthcare Query Intent-WebMD and 1mg (IHQID-WebMD). The dataset provide an authentic representation of Indian hospital query data, spanning several Indic languages such as Hindi, Bengali, Tamil, Telugu, Marathi, and Gujarati. However, for the purposes of our evaluation, we will only focus on the English and Gujarati versions of these datasets. Each of these language versions contains 306 training samples, collectively covering a total of four intents. This diversified dataset selection is integral to measuring the adaptability and effectiveness of our STP approach in real-world, multilingual contexts.

A.3 BASELINES

- Chain of Thought Prompt (Wei et al., 2022): This method provides a step-by-step, ground truth explanation to ChatGPT, allowing it to follow a logical sequence of thoughts.
- Self-Consistency Prompt (Wang et al., 2022): This method aims to enhance the consistency of generated answers by sampling multiple and diverse paths using a few-shot chain of thought approach.
- Random-CoT Prompt: This serves as the baseline for Self-Teaching Prompting. Instead of using our proposed iterative framework, it randomly selects an answer from our consistent sample distribution without adapting to our self-taught scheme.
- Tree of Thoughts Prompt (ToT): (Long, 2023; Yao et al., 2023) designed the Tree of Thoughts (ToT) prompt. It endows Large Language Models (LLMs) with the ability to engage in intermediate step exploration and correction by 'communicating to themselves.'
- Few Shots Prompt: (Brown et al., 2020) introduces the concept of few-shot prompts, which use a few examples as demonstrations in the prompt to help the model improve its performance.

A.3.1 EXPERIMENTAL RESULTS

The Table 1 presents a comparison of accuracy between the Self-Teaching Prompting Method and other methods. Table 2 displays the intent matching/accuracy for the additional baseline methods, Tree of Thoughts and Few Shots.

A.4 COST AND BENEFIT TRADE OFF ANALYSIS

The iterative process of STP, which requires multiple epochs and temperature configurations, inevitably increases computational resource usage compared to methods that do not need iterative updates. However, the increased complexity and resource usage are justified by the improved accuracy. In addition, it should be noted that even our initial learning phase outperforms other prompting methods. Additionally, in resource-constrained environments, STP can be adjusted by decreasing the number of epochs and number of temperatures, given the available resources. This adaptability

Table 1: ChatGPT 3.5 Generated Prediction Label Set: Matching Ratios and Subset Ratios for two whole datasets (MixATIS and MIXSNIPS) at different confidence levels (0.1, 0.3, 0.5, and 0.7). Random-CoT Prompting uses randomly selected example from the positive sample pool for each question.

Level of Randomness	0.1	0.3	0.5	0.7	Average	Diff from Third Epoch (MixATIS)
MixATIS - Chain of Thought Prompting						
Matching Ratio	0.3073	0.3000	0.3045	0.286	0.2995	0.1182
Subset Ratio	0.5164	0.5153	0.5185	0.498	0.5120	0.0196
MixATIS - Random-CoT Prompt (Baseline)						
Matching Ratio	0.3403	0.3382	0.3386	0.3286	0.3364	0.0813
Subset Ratio	0.5106	0.5102	0.5152	0.5064	0.5106	0.0210
MixATIS - **Self-Teaching Prompting - First Epoch**						
Matching Ratio	0.3733	0.3767	0.3761	0.3448	0.3677	0.05
Subset Ratio	0.5705	0.5764	0.5784	0.5516	0.5692	-0.0376
MixATIS - **Self-Teaching Prompting - Second Epoch**						
Matching Ratio	0.4322	0.4231	0.4366	0.4054	0.4243	-0.0066
Subset Ratio	0.5423	0.5381	0.5583	0.5316	0.5426	-0.011
MixATIS - **Self-Teaching Prompting - Third Epoch**						
Matching Ratio	0.39832	0.4302	0.4208	0.4215	0.4177	-
Subset Ratio	0.5141	0.5401	0.5351	0.5372	0.5316	-
MIXSNIPS - Chain of Thought Prompting						Diff from Third Epoch (MIXSNIPS)
Matching Ratio	0.3962	0.4147	0.4352	0.4567	0.4257	0.3555
Subset Ratio	0.4459	0.4644	0.4814	0.5004	0.4730	0.3325
MIXSNIPS - Random-CoT Prompt (Baseline)						
Matching Ratio	0.568	0.602	0.586	0.568	0.581	0.2002
Subset Ratio	0.612	0.63	0.622	0.63	0.623	0.1825
MIXSNIPS - Self-Teaching Prompting - First Epoch						
Matching Ratio	0.6613	0.6765	0.6832	0.6869	0.6770	0.1042
Subset Ratio	0.7455	0.7602	0.7680	0.7715	0.7613	0.0442
MIXSNIPS - **Self-Teaching Prompting - Second Epoch**						
Matching Ratio	**0.7925**	0.7557	0.7548	0.7589	0.7655	0.0157
Subset Ratio	0.8238	0.7913	0.7904	0.7960	0.8004	0.0051
MIXSNIPS - **Self-Teaching Prompting - Third Epoch**						
Matching Ratio	0.7688	**0.7783**	**0.7951**	**0.7827**	0.7812	-
Subset Ratio	0.7936	0.8018	0.81867773	0.8077	0.8055	-

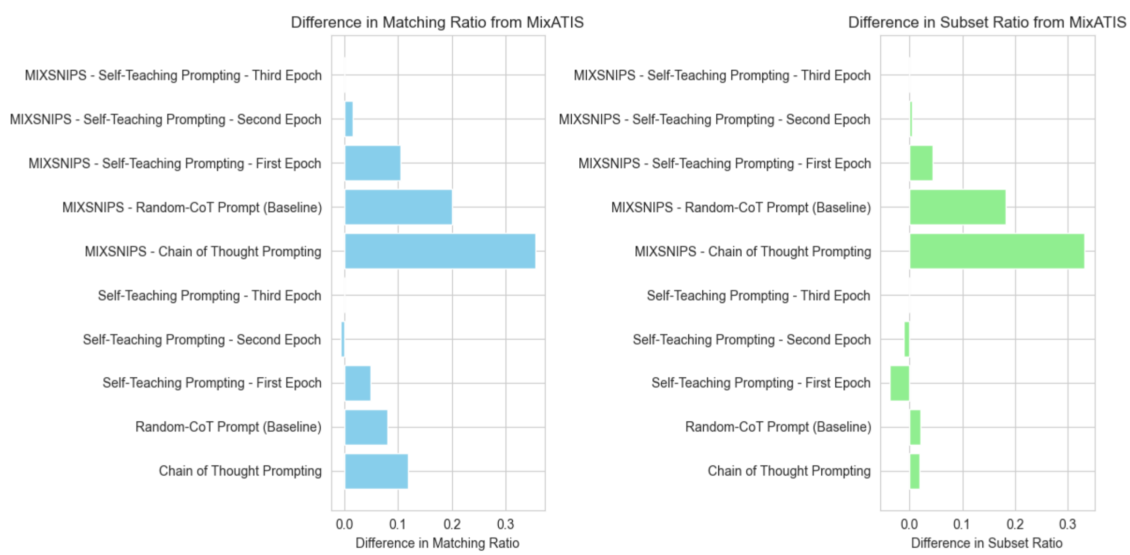

Figure 2: Intent Accuracy and Subset Ratio Comparison of our Self-Teaching and Chain of Thought Prompt, Random-CoT Promp (Our Baseline Method).

endows STP with the flexibility to operate in various scenarios, making it a feasible solution even when computational resources are scarce. The results shown in Tables 5, 6, and 7 illustrate a drastic improvement in intent accuracy through the Self-Teaching Prompting (STP) approach. More specifically, we observe continuous improvement in intent accuracy across each epoch, indicating the efficacy of the STP in improving model performance and reducing randomness. For instance, in

Dataset	Few Shots Prompting		Tree of Thought Prompting	
	Subset Ratio	Matching/Accuracy Ratio	Subset Ratio	Matching/Accuracy Ratio
Mixatis	0.5371	0.3452	0.5600	0.3600
Mixsnips	0.7871	0.7316	0.7425	0.6729

Table 2: Comparison of Subset and Accuracy Ratios for Mixatis and Mixsnips on ToT and Few Shots Prompting. The above results are conducted using temperature at 0.7.

Method	Total Estimated Time	Total Estimated Cost (Dollars)
Our Method	92 hours, 27 minutes, and 45 seconds	59.779152
ToT	14 hours, 7 minutes, 22 seconds	11.724276
Few Shots	10 hours, 17 minutes, 33 seconds	30.905524

Table 3: Total Estimated Time and Cost for Different Methods on MIXSNIPS.

the Medical Dataset (English) as shown in Table 5, the STP method demonstrates a consistent increase in matching ratio/accuracy across multiple epochs. Starting with an intent accuracy of 0.8385 in Epoch 1, the method shows gradual improvement. This trend indicates the STP's ability to endow the model with self-teaching capabilities and progress over each epoch. In addition, compared to other methods such as Tree of Thought and Few Shots Prompting, the STP method exhibits superior intent accuracy and lower standard deviation, as shown in the Legal Dataset (English) in Table 7. The STP method not only outperforms in terms of intent accuracy but also demonstrates high consistency, as indicated by the lower standard deviations. Lastly, Table 4 shows that the Self-Teaching Prompting (STP) method is a cost-effective approach, especially when dealing with challenging tasks like multi-intent classification. It costs only 8 dollars, yet achieves a 7.91% higher intent accuracy compared to the Few Shot method and a 6.43% improvement in intent accuracy against the ToT method. Therefore, the total cost incurred remains significantly low.

A.5 Self-Teaching Pipeline

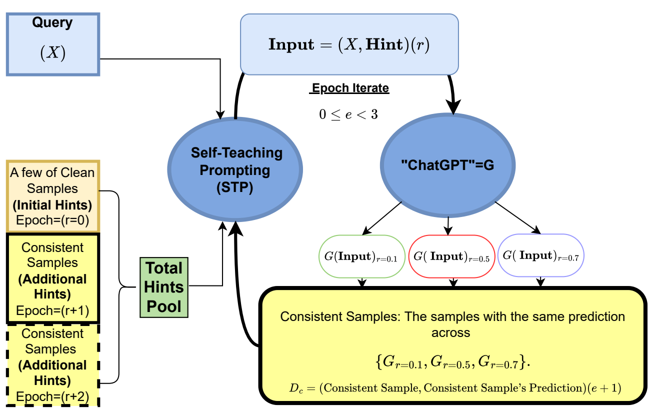

Figure 3: Self-Teaching Prompting

A.6 Experiment on Additional Datasets

We have conducted experiments on additional datasets, including a medical dataset in both English and Gujarati (one of the languages of India), as well as a legal dataset. These are shown in **Table 2** (Medical Dataset in English), **Table 3** (Medical Dataset in Gujarati), and **Table 4** (Legal Dataset). We compared our **Self-Teaching Prompting (STP)** method with **Chain of Thought (CoT)**, **Self-Consistency**, **Tree of Thought**, and **Few Shot Prompting**. Our method consistently outperforms the other methods. We conducted experiments at four different temperatures to demonstrate the consistency of our work. The total cost (US dollars) and estimated time for each dataset are also included, corresponding to all methods.

Dataset	Total Estimated Time	Total Estimated Cost (Dollars)
Our Method	8 hours, 46 minutes, 28 seconds	19.56491814
Few Shots	1 hours, 52 minutes, 6 seconds	11.104942

Table 4: Total Estimated Time and Cost for Our Method and other method on MIXATIS

Method	Matching Ratio/Accuracy	Std	Estimate Time	Total Cost
Our (STP)				
STP Epoch 1	0.8385	0.004259	0h 15m 52s	0.27418
STP Epoch 2	0.8352	0.001419	0h 11m 58s	0.27352
STP Epoch 3	0.8401	0.001412	0h 11m 52s	0.27352
STP Epoch 4	0.8418	0.002718	0h 11m 54s	0.27354
Self-Consistency				
Self-Consistency	0.7984	0.00882	0h 12m 2s	0.28546
Chain of Thought				
Chain of Thought	0.7672	0.01331	0h 11m 59s	0.27550
Tree of Thought				
Tree of Thought	0.7459	0.0314	0h 15m 46s	0.37620
Few Shots				
Few Shots	0.8082	0.008518	0h 11m 57s	0.39214

Table 5: Summary of Metrics Medical Dataset (Single Intent Classification) in English. The Std denotes standard deviation

Method	Metric	Std	Estimate Time	Total Cost
Our (STP)				
STP Epoch 1	0.740164	0.002719	0h 12m 7s	0.936448
STP Epoch 2	0.740164	0.003573	0h 11m 59s	0.936008
STP Epoch 3	0.741803	0.008479	0h 12m 6s	0.937416
STP Epoch 4	0.745082	0.002719	0h 12m 0s	0.936544
Self-Consistency				
Self-Consistency	0.711475	0.004016	0h 12m 8s	1.019896
Few Shots Prompting				
Few Shots Prompting	0.690164	0.008828	0h 11m 58s	1.430464
Chain of Thought				
Chain of Thought	0.646721	0.019723	0h 12m 8s	1.019896
Tree of Thought				
Tree of Thought	0.726230	0.033796	0h 12m 9s	0.837304

Table 6: Summary of Metrics Medical Dataset (Single Intent Classification) in Gujarati. The Std denotes standard deviation

Metric	Matching Ratio/Accuracy	Std	Estimate Time	Total Cost
Our (STP)				
STP Epoch 1	0.577978	0.007047	0h 14m 38s	6.86064
STP Epoch 2	0.594224	0.005306	0h 14m 6s	
STP Epoch 3	0.598014	0.003477	0h 14m 14s	
Self-Consistency				
Self-Consistency	0.576173	0.076412	0h 41m 54 s	1.78551
Tree of Thought Prompting				
Tree of Thought	0.522563	0.012551	0h 13m 52s	2.02092
Few Shots Prompting				
Few Shots	0.581408	0.004128	0h 14m 55s	4.17343
Chain of Thought Prompting				
Chain of Thought	0.575632	0.004772	0h 13m 8s	2.28816

Table 7: Summary of Metrics for Legal Dataset (Single Intent Classification) in English. The Std denotes standard deviation.

A.7 EXPERIMENTAL DETAILS

In Self-Teaching Prompting (STP), the group prompting format, where each query consists of multiple utterances, has been utilised on the MixATIS and MixSNIPS datasets to obtain a prediction for each utterance from ChatGPT. This formatting is more economic for large scale multi-intent classification tasks. The single prompting format, where each query consists of a single utterance, has been applied to Legal and Medical datasets (in both English and Gujarati).

A.8 STP PROMPTING OVERALL STRUCTURE

In the following we have provided overall structure of our Self-Teaching Prompting.

```python
import openai
import numpy as np
import pandas as pd
import json
from gensim.models import Word2Vec
import time

# Set your OpenAI API key
api_key = "YOUR_API_KEY"
openai.api_key = api_key
temperatures=[0.1,0.3,0.5,0.7]
# [Other initializations and configurations]
# examples=[Clean Samples]
# Function to calculate similarity between sentence and examples
def calculate_similarity(sentence, examples, model):
    sentence_vector = np.mean([model.wv[word] for word in sentence.lower().split() if word in model.wv], axis
        ↪ =0)
    similarities = {}
    for example, vector in examples.items():
        if np.isnan(vector).any() or np.isnan(sentence_vector).any():
            continue
        try:
            similarity = np.dot(vector, sentence_vector) / (np.linalg.norm(vector) * np.linalg.norm(
                ↪ sentence_vector))
            if not np.isnan(similarity):
                similarities[example] = similarity
        except Exception as e:
            print(f"Error␣calculating␣similarity␣for␣'{example}':␣{e}")
    return similarities

# Main processing loop
for epoch in range(3):
    for temperature in temperatures:
        # [Processing code for each epoch and temperature]
        # ...

        # Calculate similarities and select top similar example
        example_vectors = {example: model.wv[word] for word in example.split() if word in model.wv}
        for sentence in dataset:
            similarities = calculate_similarity(sentence, example_vectors, model)
            top_similar_example = max(similarities, key=similarities.get)

            # [Further processing with the selected example]
```