UNDER REVIEW

RESCIENCE C

Replication / ML Reproducibility Challenge 2022

A Reproduction of Automatic Multi-Label Prompting: Simple and Interpretable Few-Shot Classification

Anonymous

Edited by
(Editor)

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
–

Published
–

DOI
–

A replication of H. Wang, C. Xu, and J. McAuley. Automatic Multi-Label Prompting: Simple and Interpretable Few-Shot Classification. 2022.

Reproducibility Summary

Scope of Reproducibility — We reproduce the results in the paper Automatic Multi-Label Prompting: Simple and Interpretable Few-Shot Classification by Wang et al. We reproduce their results on 3 GLUE datasets and extend experiments to 2 new non-GLUE ones.

Methodology — We used the author's code with minimal additions to accommodate our new datasets. We ran all experiments using a single combination of hyperparameters (author's repository's defaults) over 5 seeds, mainly using RTX8000 for 125 hours.

Results — We reproduced the original paper's metrics to within the standard deviation reported, verifying the paper's claims that its proposed method achieves competitive results and beats some baselines under different settings (whether or not models are fine-tuned). Our extended experiments show the method's potential in being used on real-world data, but uncovers new considerations regarding prompt template, language model choice, and seed choice to achieve satisfactory performance.

What was easy — The complete code implementation was available publicly with step by step instructions on how to get AMuLaP running with GLUE datasets. This made it easy to begin reproducing the work. Finding flagged and non-flagged tweets from Donald Trump to create our dataset was easy thanks to thetrumparchive.com which compiled this.

What was difficult — The original code is lengthy and lacks information on implementation dependencies, making it time-consuming to understand. It is made only for GLUE tasks and RoBERTa models, needing modification to work on new experiments. Some language models were not trained with a mask token, thus unsuitable for AMuLaP, and required us to extensively search for solutions. Finally, as label engineering is also tied to prompt engineering, we find expanding the work through the template lens challenging without prior experience in manual prompt engineering.

Communication with original authors — We did not feel the need to reach out to the authors as the method was well explained and simple to understand with basic probability knowledge after going through the code. We will reach out to them for feedback.

Copyright © 2022 , released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to ()
The authors have declared that no competing interests exists.
Code is available at .

1 Introduction

Prompt-based learning with pre-trained large language models (PLM) such as GPT-3 [1] have recently gained popularity and much work has been done in searching for prompting templates (i.e. prompt-engineering) [2] that best allow the PLM to treat tasks as masked language modeling problems. But when we want the PLM to fill in masks from a finite set of label words, finding good label word sets is just as important (i.e. label engineering). The search for label words usually requires task expertise, knowledge of the internal PLM, or enough computation for a brute-forced search of the PLM's vocabulary.

To mitigate the need for this manual work and expense, Wang, Xu, and McAuley [3] proposed a novel technique for automatic multi-label map search through prompting. Their proposed solution, AMuLaP, can be described as using prompting on a set of same-class samples to find a group of label representatives for the class without needing human effort. Using a group of labels reduces the noise brought by single labels when making predictions. On a high level, the authors claim competitive accuracy results on the GLUE dataset without fine-tuning nor access to model weights against majority voting and manual label zero-shot baselines. Its best performance is reached with fine-tuning, outperforming direct few-shot fine tuning on all GLUE datasets except for CoLA.

Automatic label search using prompt-based learning is relatively new, with only two works parallel to AMuLaP preceding it, called AutoL [4] and PETAL [5]. In this work, we aim to verify their main claims by reproducing their results on 3 GLUE datasets with and without fine-tuning and test their ability to generalize to real-world tasks and other models.

2 Scope of Reproducibility

2.1 Context and Original Claims

AMuLaP alleviates the need for manual label word search to optimize prompt-based learning with PLMs. Given a prompt template, AMuLaP uses a statistics-based selection method to choose multiple words to map as labels for a single class. As a result, the method's way of finding label mappings brings the advantage of being parameter-free and simple. The authors claim AMuLaP's ability to achieve competitive performance on GLUE tasks without accessing model weights, beating majority-vote and manual label zero-shot baselines across all GLUE datasets. Further, it reaches its best results with a fine-tuned model, outperforming direct few-shot fine-tuning on all GLUE datasets except for CoLA. AMuLaP's works as its one-to-many label mapping per class reduce the noise brought by limited few shot examples and PLM complexities during automatic label selection and inference. Finally, they suggest that the increase of supervision pairs when selecting multiple labels during fine-tuning resembles working with augmented data, possibly being another reason to why multi-labels work.

2.2 Experiment Scope

Our work verifies the following claims of the paper and expands on it by exploring its generalization to real-world tasks and models. We use the original code, with modifications to accommodate two new non-GLUE datasets and models, and outline our experiments below:

- We run AMuLaP under 2 settings (fine-tuned model and not) for MNLI, MNLI-mm, SST-2 and CoLA to verify claims of competitive performance on GLUE datasets.

- We run AMuLaP on 2 new datasets (flagged and non-flagged Trump tweets and Reddit comment-reply pairs). This is done for 3 goals: (1) to test the method's robustness on unclean real-world data. (2) Verify the author's claims related to multi-labels and noise by examining generated multi-labels. (3) explore the possible use case of AMuLaP's multi-labels as a way to use PLMs to uncover insights about why datasets are classed a certain way. Particularly, we are interested in the multi labels as additional context behind why groups of Trump tweets were flagged, and as a clearer reason behind why a Reddit comment-reply pair falls in the "unsure" relation class.
- We try AMuLaP with PLMs other than its RoBERTa-Large backbone and new manual templates to test the claim that the method works due to its use of multi-labels. In particular, we want to see whether the method's success relies heavily on the choice of PLM/template rather than the idea of one-to-many label mappings.
- The original implementation builds upon AutoL's Gao, Fisch, and Chen [4] codebase, a related work preceding it. To improve its usability, we modified the codebase to be more modular and added information on how to use it on non-GLUE datasets.

The rest of our report proceeds as follows. Section 3 outlines our methodology, and we summarize how AMuLaP works in section 3.1. Section 4 shows our results, section 5 a discussion of our insights from reproducing the work, and we end in section **??** with information on our members' contributions.

3 Methodology

We use the author's code with some modifications to handle non-BERT-based models and our two non-GLUE datasets. After these modifications, we follow the instructions in the README file to run experiments. We also use code from Gao, Fisch, and Chen [4] to run an alternative automatic label search technique called AutoL on the Trump tweets and Debagreement. AutoL is an alternative to AMuLaP used by the authors as a comparison, which automatically finds single labels using prompts.

3.1 Model descriptions

AMuLaP is a method that uses the whole vocabulary of the language model to find the best labels. Using n examples per class, we sum the predictions of the mask token over the whole vocabulary. Then, for each word, we assign it to the class with the highest probability. Finally, we select k labels with the highest probability for each class. When testing, we sum the prediction on the mask token of these multi-labels for each class to get the prediction class with the highest sum. Refer to the original paper and Figure 1 for a formal description of the method [3].

The main model used is RoBERTa Large (355M parameters) [6] with pretrained weights modified to do prompt fine-tuning. In Table 4, we also used DerBERTa-V2 xlarge (900M) and xxlarge (1.5B) [7], RoBERTa Base (125M) [6], and BERT Large Uncased (340M) [8] for comparison. Similarly, the models were modified for prompt fine-tuning, and pretrained weights were used. From the original paper of each model, the results on the GLUE benchmark [9] should be better the more parameters it has, apart from RoBERTta Base, which performs similar to and even outperforms BERT Large on certain tasks. A major limitation when choosing a model is that it has to have been trained with a mask token; otherwise we would need to fine-tune it, which might be resource expensive. To use a generative model, we would need to modify the work extensively, and there is no say in whether the method would work as well.

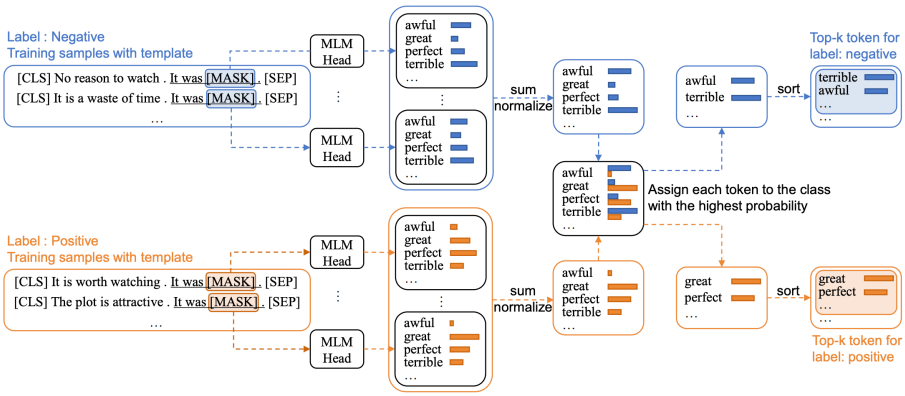

Figure 1. Illustration from the original paper of implementing AMuLaP on a binary sentiment classification task (SST-2). Each training sample with the task-specific template (the underlined text) is fed into a pre-trained language model L to get its own probability distribution over the vocabulary V. The obtained probability distributions are summed by class and normalized to get the probability distribution of each class. Then each token in V is assigned to the class with the highest probability (e.g., the token terrible is assigned to the class negative, and the token great to the class positive). Finally, we choose the top-k tokens as label words for each class [3]

.

3.2 Datasets

We use 4 GLUE datasets (MNLI, MNLI-mm, SST-2, and CoLA), a set of Donald Trump tweets from The Trump Tweet Archive (https://www.thetrumparchive.com) and Reddit comment reply pairs from Debagreement [10]. All of them were processed into train, dev, and test splits using the author's code to create k-shot data (generate_k_shot.py). The code creates train and test files with k samples for each class, and we used $k = 16$ in our experiments. The test set is the dev set of an original GLUE dataset and the new dev set is the same length as the new train set. Below are details of each dataset:

- **MNLI**: MNLI is a dataset composed of sentence pairs annotated with textual entailment information (Entailment, contradiction, neutral) with each class represented approximately equally [11]. Here is an example:

  Premise: Your gift is appreciated by each and every student who will benefit from your generosity.
  Hypothesis: Hundreds of students will benefit from your generosity.
  Label: neutral

- **SST-2**: The Stanford Sentiment Treebank regroups sentences labeled with a positive or negative sentiment. There are about the same amount of examples for each class [12]. Here is an example:

  Positive example: a smile on your face
  Negative example: the action is stilted

- **CoLA**: The Corpus of Linguistic Acceptability is a set of 10,657 English sentences labeled as grammatical or ungrammatical (acceptable or unacceptable) represented with 70.5% acceptable [13].

  Example of an acceptable sentence: They made him president.
  Example of an unacceptable sentence: The car honked down the road.

- **Trump Tweets**: Trump's tweets are either flagged or not flagged by Twitter; these are our classes. Since there were only 304 flagged tweets out of 50,000 tweets, we

created a dataset using randomly chosen 694 ok tweets to get 1000 data points with an average of 38 tokens per tweet. We split the training with a ratio of 0.3, and the dev set is 0.1 of the test set.

Here is an example of a flagged tweet: RIGGED ELECTION. WE WILL WIN!
Here is an example of an ok tweet: So great to watch this! https://t.co/pYoiLjM0pz.

- **Debagreement**: It is a dataset composed 42,894 comment-reply pairs extracted from Reddit that are either agreeing, disagreeing, neutral or unsure [10]. On average, each pair consists of 89 tokens (using RoBERTa-Large tokenizer).

  Example of a disagreement: Comment: I feel like the pictures should be reversed… Only one side is throwing a fit.
  Reply: We've had months of leftists burning and looting, you think the right is throwing a fit?

3.3 Hyperparameters

We replicated the use of hyperparameters in the original paper as much as possible; we only tried one learning rate and one batch size instead of 3. For all GLUE experiments, we tried five different seeds (13, 21, 42, 87, and 100) to generate the training split. For Trump and Debagreement, we only used a single set of training split. We used the same seeds for all experiments to get the standard deviation and average. We report results from the highest top-k based on the accuracy of the dev set. We used top-k's of 1, 2, 4, 8, and 16 for fine-tuning, and 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, and 1024 for no fine-tuning. We used a batch size of 8 for all experiments and a learning rate of 1e-5 when fine-tuning. Figure 2 shows the impact of the top-k on SST-2 and MNLI-mm. We found that depending on the template and the dataset, the best top-k can be large, but most of the time, there are no specific patterns to determine the best top-k. Larger top-k reduces the standard error on the results and with a good template, it should perform well compared to a small one.

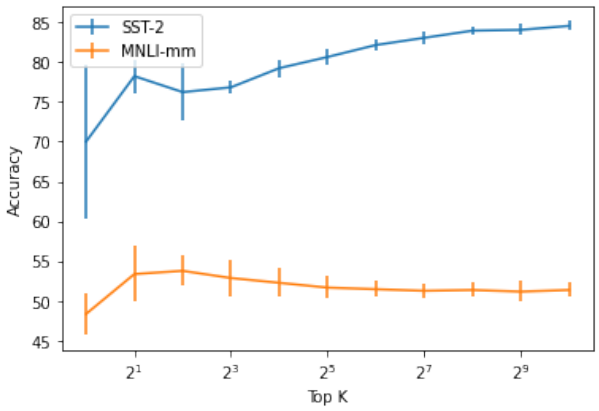

Figure 2. Accuracy on SST-2 and MNLI-mm using AMuLaP with different top K's with no fine-tuning

3.4 Experimental setup and code

The experiment can be run using a python file with arguments. We ran each dataset with the hyperparameters described in Section 3.3 to report the best result based on the dev set. A file named prompt.py has a class containing multi-label prompting methods. New models that support mask modeling can easily be added in model.py by replicating the

code used for the other classes. Every dataset was evaluated using accuracy, but CoLA uses Matthews Correlation. The code is available at [hidden]. Refer to the README.md file for more information to run the code.

3.5 Computational requirements

We primarily used RTX8000 for AMuLaP runs, which took 2 minutes per run without fine-tuning, and 10 minutes per run with fine-tuning on average. We took an average of 250 minutes for fine-tuning experiments and 110 minutes non fine-tuned experiments. When working with DeBERTa-V2-xxlarge, we used A100. AutoL experiments were run on standard Google Colab and Kaggle GPU, with single runs taking 2 and 10 hours for the Trump and Debagreement datasets, respectively. In total, with the ability to run over some experiments in parallel, our main AMuLaP experiments took a total of 125 hours.

4 Results

Our results generally support the claim made by the original authors, of AMuLaP's competitive results on the GLUE datasets and the reduction of noise by using multiple labels. Further, our experiments on Trump tweets and Debagreement showed AMuLaP's potential in generalizing to real-world data. However, we found a large decline in performance when using non-default PLMs and templates.

4.1 Results reproducing original paper

We ran the experiments on SST-2, MNLI, MNLI-mm, and CoLA to reproduce the results from the original paper (first point in section 2.2), the two of which are compared in Table 1. The average results we got for AMuLaP without fine-tuning is within the standard deviation of the original work and slightly worse when fine-tuning. Overall, the results correspond to what was shown in the original paper.

	MNLI (acc)	MNLI-mm (acc)	SST-2 (acc)	CoLA (Matt.)
Setting 2: Dtrain + Ddev ; No parameter update.				
Original AMuLaP	50.8 ± 2.1	52.3 ± 1.8	86.9 ± 1.6	2.3 ± 1.4
Reproduced AMuLaP	51.3 ± 1.9	52.9 ± 2.3	86.9 ± 1.4	2.2 ± 2.9
Setting 3: Dtrain + Ddev ; Prompt-based fine-tuning.				
Original AMuLaP FT	70.6 ± 2.7	72.5 ± 2.4	93.2 ± 0.7	18.3 ± 9.4
Reproduced AMuLaP FT	67.0 ± 2.7	69.1 ± 3.2	92.6 ± 1.4	17.0 ± 11.1

Table 1. Reproduced and original experiment results under two settings on some GLUE dataset using Roberta. For few-shot settings, n is set to 16 per class. For reproduction, we give the average and standard deviation over 5 runs (different seeds) for all experiments.

We got slightly worse results when fine-tuning, which might be because we only tried a single learning rate. Trying multiple learning rates would have been too long and would have use much more computation resources.

4.2 Results beyond original paper

The first two subsections below refer to the second point in section 2.2.

Default AMuLaP on Trump tweets and Debagreement — Table 2 shows the results of running AMuLaP with RoBERTa-Large for the Trump and Debagreement datasets. We also ran AutoL to use as a comparison, similar to the original paper. We find relatively high accuracies for Trump: 77.3% without fine-tuning and 86.7% with, but unsatisfactory accuracies for Debagreement: 30.5% without fine-tuning and 40.5% with.

	Trump (acc)	Debagreement (acc)
Setting 2: Dtrain + Ddev ; No parameter update.		
AMuLaP	77.3	30.5
Setting 3: Dtrain + Ddev ; Prompt-based fine-tuning.		
AMuLaP FT	86.7 ± 2.7	40.5 ± 1.3
Auto-L FT	89.1 ± 1.1	36.4

Table 2. Experiment results under two settings for the Trump and Debagreement datasets using AMuLaP (multi-label) and AutoL (single label) with Roberta Large. For few-shot settings, k is set to 16 per class. We show the average and standard deviation over 5 runs with different seeds for all experiments apart from Auto-L on Debagreement, done on a single run.

AMuLaP for Trump tweets and Debagreement: Label and Template — Table 3 shows the multi-labels found using AMuLaP for the Trump and Debagreement datasets along with the fixed manual templates used to find them:

	Generated Multi-Labels
Trump Template 1: "Sentence 1. It was [MASK]."	
Flagged tweets	rigged, awesome, close, disgusting, terrible, huge, over, shocking, illegal, ridiculous
Ok tweets	fun, true, amazing, inevitable, not, great, beautiful, funny, good, incredible
Trump Template 2: "Sentence 1. This is a [MASK] tweet."	
Flagged tweets	false, TRUE, warning, lawful, FALSE, fraud, rigged, harassing, felony, panicked
Ok tweets	recent, real, new, live, deleted, related, verified, true, great, sponsored
Debagreement Template 1: "Sentence 1. [MASK]. Sentence 2."	
Disagreeing pairs	Or, And, </s>, Mattis, Something, Like, *, Even, Wait, -
Agreeing pairs	But, The, If, and, Now, New, Assuming, US, Yes, Heck
Neutral pairs	So, It, Well, Maybe, Just, ., Also, <, Obviously, ...
Unsure pairs	Oh, Why, I, Because, As, but, Ok, However, Not, OK

Table 3. Multi labels found for each dataset class using a specific fixed manual template with AMuLaP and RoBERTa-Large

The table shows a higher amount of noise (non-word labels) chosen as multi labels for the Debagreement dataset compared to Trump tweets, which in fact, presents none.

AMuLaP with Alternative PLMs — Following point 3 in section 2.2, we test AMuLaP with PLMs other than its default RoBERTa-Large backbone and report results in Table 4. In this experiment, we take the results using the backbone RoBERTa-Large as a baseline. We observe that the baseline model performs best for the Debagreement dataset in both settings and SST-2 in the fine-tuned setting. It is outperformed in the non-fine-tuned setting only by DeBERTa for the SST-2 and Trump datasets and in the fine-tuned setting by RoBERTa-Base for the Trump dataset.

	SST-2 (acc)	Trump (acc)	Debagreement (acc)
Setting 2: Dtrain + Ddev ; No parameter update.			
AMuLaP RoBERTa-Large (Baseline)	86.9 ± 1.6	77.3	30.5
AMuLaP Deberta-V2-xxlarge	89.3 ± 0.7	78.9	27.5
AMuLaP Deberta-V2-xlarge	81.9 ± 2.4	79.4	28.7
AMuLaP Roberta-Base	84.5 ± 0.6	63.0	25.0
AMuLaP Bert-Large-Uncased	73.8 ± 2.0	63.0	26.6
Setting 3: Dtrain + Ddev ; Prompt-based fine-tuning.			
AMuLaP RoBERTa-Large (Baseline)	93.2 ± 0.7	86.7 ± 2.7	40.5 ± 1.3
AMuLaP Roberta-Base	89.5 ± 0.4	76.7 ± 1.7	39.0 ± 0.6
AMuLaP Bert-Large-Uncased	84.7 ± 1.6	82.9 ± 2.4	30.6 ± 0.5
Auto-L Roberta-Base	-	80.3 ± 2.1	31.1

Table 4. Experimental results under two settings. Reproduction of results from using different models on SST-2, Trump, and Debagreement datasets for AMuLaP. For few-shot settings, n is set to 16 per class. We show the average and standard deviation over 5 runs for all experiments apart from Auto-L on Debagreement, which was done on a single run.

5 Discussion

Our work verifies the authors' claims about AMuLaP on the GLUE dataset, as our reproduced results fall within the standard deviation of the original report. Testing AutoL on Trump and Debagreement also supports the author's motivation for using multiple labels. Instead of multi-labels, Auto-L searches for good combinations of single labels per class and consistently achieves results lower than AMuLaP with the same PLM.

Our extended experiments uncover the potential use of AMuLaP and the factors needed for its success. AMuLaP's multi-labels can be used to find possible reasons behind classification, as common interpretations of the Trump multi-labels make sense as the reasoning behind why tweets were flagged (labeled "false" or "rigged") or not (labeled "real" or "true"). Then, examining our poor Debagreement results uncovered a need to explore effective templates to support automatic label search. Table 4 shows PLM choices impact performance. Also, the 0.48 average standard deviation increase in our reproduction of GLUE dataset results in table 1 shows the significance of seeds in AMuLaP's performance. We elaborate on our observations regarding template significance and PLM choice below.

In the classification task lens, AMuLaP's high accuracy on the Trump dataset is proof of its potential on real-world data, while its poor results on Debagreement raise new considerations. A closer look at the generated labels shows AMuLaP's fixed template aspect as a weakness. The generated labels (shown in table 3) contain noise, and 90% of them are conjunction words. The selection of these words makes sense as the template places the mask token between the comment and reply pair. But, the occurrences of similar purpose conjunctions (ex: "Even", "But", "Heck" and "However" across three different classes) suggest that they may not be informative enough for accurate classification.

Informed by this observation, we tested other manual templates that resemble successful Trump ones by putting the mask token at the prompt end. However, we could not find a template that reaches a higher accuracy. We attempted using the automatic template search functionalities of AutoL [4], but needed more resources to complete it. Joint automatic template and label search remain a future task for us.

Using different PLMs, table 4 verify the authors' claim that AMuLaP achieves its best re-

sults with access to the model weights. All experiments under setting 3 outperform their equivalent in setting 2. We observe that larger PLMs do not consistently achieve better results, as shown by the baseline RoBERTa-Large model exceeding DeBERTa-V2-xlarge for SST-2 in setting 2. We are interested in how well AMuLaP can use larger models; as we have seen, DeBERTa-V2 achieved better results than RoBERTa while still being relatively small compared to OPT models of GPT-3. We wanted to use GPT-3, but it is currently impossible to get the predictions over the whole vocabulary as AMuLaP requires. This feature may be available in the future, or we will need to contact OpenAI.

Other aspects of our work to be improved are finding an appropriate baseline to measure our findings on Trump tweets and Debagreement and running our experiments testing different PLMs on multiple seeds. We also still need to complete our planned human evaluation, where literature professors and student judge whether the automatically searched multiple labels are good representations of a class. We needed to give more human annotator volunteers more time to complete annotating.

5.1 What was easy

The complete code implementation was available publicly with step-by-step instructions on how to get AMuLaP running with GLUE datasets. This made it easy to begin the reproduction of the work. Finding flagged and not flagged tweets from Donald Trump was also easier than we initially thought. We were able to easily create our dataset thanks to thetrumparchive.com which compiled the tweets and labeled each of them.

5.2 What was difficult

We needed to understand the code before fully understanding the explanation of AMuLaP in the original paper. However, the code was built upon another paper's work and contained lengthy files, making it time-consuming to understand. Further, the implementation works only with GLUE datasets and BERT-based PLMs, without adequate documentation on what to change to work with new data and non-BERT PLMs. Most models were not trained with a mask token and, thus, were unsuitable for a prompt-based method with a mask, such as how AMuLaP was implemented. As we wanted to try a larger model than RoBERTa, we had to extensively search to find a way to find one. We used DeBERTa-V2 for mask modeling, but the most recent version of HuggingFace transformers had an issue loading the weight of the model head. Fortunately, the issue was being solved concurrently with our research, and we could use a branch directory. Because of time constraints and GPU memory limitations, we could not fine-tune DeBERTa-V2. Finally, the label engineering problem AMuLaP tries to solve is tied to prompt engineering, so we find template experimentation challenging without prior experience.

5.3 Communication with original authors

We did not need to reach out to the authors as the method was well explained and simple to understand with basic probability knowledge after going through the code. We will reach out to them for feedback.

References

1. T. B. Brown et al. **Language Models are Few-Shot Learners**. 2020.
2. P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. **Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing**. 2021.

3. H. Wang, C. Xu, and J. McAuley. **Automatic Multi-Label Prompting: Simple and Interpretable Few-Shot Classification**. 2022.
4. T. Gao, A. Fisch, and D. Chen. "Making Pre-trained Language Models Better Few-shot Learners." 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)**. Online: Association for Computational Linguistics, Aug. 2021, pp. 3816–3830.
5. T. Schick, H. Schmid, and H. Schütze. "Automatically Identifying Words That Can Serve as Labels for Few-Shot Text Classification." In: **Proceedings of the 28th International Conference on Computational Linguistics**. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 5569–5578.
6. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. **RoBERTa: A Robustly Optimized BERT Pretraining Approach**. 2019.
7. P. He, X. Liu, J. Gao, and W. Chen. "DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION." In: **International Conference on Learning Representations**. 2021.
8. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186.
9. A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding." In: **Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP**. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355.
10. J. Pougué-Biyong, V. Semenova, A. Matton, R. Han, A. Kim, R. Lambiotte, and D. Farmer. "DEBAGREEMENT: A comment-reply dataset for (dis) agreement detection in online debates." In: **Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)**. 2021.
11. A. Williams, N. Nangia, and S. Bowman. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference." In: **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)**. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1112–1122.
12. R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank." In: **Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing**. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642.
13. A. Warstadt, A. Singh, and S. R. Bowman. "Neural Network Acceptability Judgments." In: **Transactions of the Association for Computational Linguistics** 7 (2019), pp. 625–641.