Semi-Supervised Learning for Prompt Classification in ChatGPT

Alexandru-Gabriel Ilie
University Politehnica of Bucharest
313 Splaiul Independentei, Bucharest, Romania
alexandru.ilie2108@stud.acs.upb.ro

Traian Rebedea
University Politehnica of Bucharest
313 Splaiul Independentei, Bucharest, Romania
traian.rebedea@upb.ro

ABSTRACT
Chat Generative Pre-trained Transformer (ChatGPT) is a large language model-based chatbot that can interact with people and hold interesting and interactive conversations. Individuals have the ability to engage in dialogues with the model by submitting input sentences or prompts of their choosing. Over the past months, ChatGPT has been continuously growing in popularity, reaching over one million users in a matter of days and surpassing the one billion visits in less than 5 months. It is clear that ChatGPT has become an important aid for numerous people, as there are various tasks it is used into, such as generation, question answering, rewriting or simple chatting. Such tasks are represented by certain instructions that are encapsulated in the user input sent to the model. Having access to the most common types of user's instructions could help Machine Learning engineers improve current datasets and models and adapt them to better suit human needs. However, obtaining a large amount of annotated data is expensive and time-consuming. In order to address the aforementioned issues, we investigate the usage of semi-supervised learning techniques. In this paper we describe the creation process of a new multi-label classification dataset for instruction classification in ChatGPT using user-shared conversations and employ various semi-supervised learning approaches in order to boost our model's performances. The unlabeled data used for semi-supervised learning methods is extracted from the same source as our labeled dataset. This approach increased the weighted F1 score of the model by 3.5%.

Author Keywords
Natural Language Processing; Instruction Classification; Multi-label Classification; Semi-supervised Learning; ChatGPT.

ACM Classification Keywords
I.2.7 Natural Language Processing: Text analysis.

DOI: 10.37789/rochi.2023.1.1.9

INTRODUCTION

The user's interactions with ChatGPT is composed of a dialogue with alternative replies on each side. Users provide an input, typically containing instructions to guide the model in assisting them with their task, for example "Could you please help me rewrite the following paragraph in a more formal tone?". In this paper, we will use user input, instruction, and prompt interchangeably, referring to the sentence sent by the user.

Being able to classify the instructions provides an opening to many opportunities, such as improving current models and state-of-the-art Machine Learning training and deploying techniques.

To accomplish this objective, our initial steps involve data extraction and labeling to create a dataset, followed by the training of a baseline model. Subsequently, we enhance its performance through the application of contemporary semi-supervised learning techniques.

In the subsequent sections, we explore deeper into each component introduced earlier.

Benefits

The extensive utilization of the Generative Pretrained Transformer (GPT) [24] model has created large amounts of data that could be explored in order to further improve the development of the generative models, by understanding which are the most common requirements of the average users. This may improve the quality of the data given to the generative model during training, as it would reveal the actual distribution of tasks commanded by the users. Hence, the developers of the model could in term focus on providing more and with an increased quality examples in that specific category.

Additionally, this could benefit the Text-to-Text models described by Raffel et al. [22], as we could add the same prefix for the input as the one utilized in training or fine-tuning, which should increase the quality of the model output.

In addition to that, there is an increased trend in fine-tuning large language models for specific tasks, without altering many of their parameters through prompt tuning, as presented by Lester et al. [23]. Given that we can correctly identify the type of instruction sent by the user, we can forward the input to the model trained on the downstream specific task, hence achieving an enhanced performance.

We would like to build a text classifier that could label the prompts sent by the users to ChatGPT in order to be able to have a broader image of the actual distribution of the categories in real-world data. Therefore, this could be a starting point for improving the ChatGPT training data.

Understanding the type of interaction users have had with GPT models, such as ChatGPT, is very important to understand how people use this new and very powerful technology. As far as we know, this is one of the first papers that proposes to better understand a large dataset of shared interactions (called

instructions) users have had with the ChatGPT model in the first months since its release. This study should also be very informative in training InstructGPT-like models [1] such as ChatGPT to better align with user needs.

Moreover, we are interested in understanding the semisupervised learning improvements that may be obtained following different modern approaches to the problem and the impact of the selection of the hyperparameters and augmentation data on the performance of each method. We aim to discover how reliant and reusable the methods are and how much depend on the specific task or model employed in the classification.

Approach

Given the benefits mentioned above, we strive to create a new dataset based on real world data. However, manually annotating an enormous amount of data is hardly achievable due to time and cost limitations.

To overcome this limitation, we created a new dataset containing 502 labeled entries, divided into 64%: 16%: 20% between training, validation and test. In order to fully exploit the data sources, we augmented 40000 prompts given by the users utilizing two major augmentation techniques with different variations. The prompts were extracted from real-world conversations among users and ChatGPT shared across the Internet through ShareGPT, a Google Chrome extension.

We employed a BERT-base model for our newly created dataset which obtained respectable results given the low amounts of labeled entries in the given dataset, achieving 0.535 accuracy and 0.624 weighted F1-score.

We further increased the performance of the model utilizing a similar setup in terms of architecture and supervised learning parameters while employing numerous variations of semi-supervised learning methods, more specifically FixMatch, in order to determine which one best suits our model and how stable they are in comparison to slight variations of their hyperparameters.

The best model obtained with semi-supervised learning has shown an increase in the weighted F1 score by 3.5% and in accuracy by 8%, utilizing FixMatch with Backtranslation for strongly augmented prompts and Easy Data Augmentation for softly augmented ones.

RELATED WORK

Training of GPT models for instruction inputs

ChatGPT is used as an assistant, therefore, most of the human input given to it is represented by instructions. An important aspect of the performance of the model, especially what transforms them into a human usable resource, is the quality of the training data.

Ouyang et al. [1] present a list of categories used for GPT models during training, created from ideas generated by the labelers.

As presented in Table 1, there were 10 classes used for classification and for each category an example is attached to describe the idea of that particular class.

The table 2 presents the class distribution used to train generative models.

In our use case, these classes represented the backbone of our labeling. We started with them and added a few others while removing the ones that appeared less frequently.

One main disadvantage of the approach presented in the paper is that only a small portion of the training data was obtained directly from customers, whereas most of it is generated by the persons labeling the data, which introduces a high bias in the dataset. One thing that we observed is that the distribution of classes presented in Table 2 was different from the ones presented in the paper, but this aspect might be a result of labeling fewer examples.

In this dataset, a high number of instructions were generative ones. While in our research generative instructions were the highest class as well, the question-answering and brainstorming classes represent more than in the paper discussed. This observation proves that the data obtained from real-world situations is different from the one imagined for training, thus proving the necessity of automating the labeling process.

FixText

FixText [3] is an extension of FixMatch [2] which operates only on textual data.

While the overall behavior is similar, as pseudo-labels created by a pre-trained model on labeled data and consistency regularization which attempts to guide the model to predict the same classes for softly augmented and strongly augmented inputs are utilized to boost the model performance, the augmentation techniques are different.

These new pseudo-labeled examples create a new loss that is added to the one computed on the labeled examples:

$$ł = l_s + \lambda_u * l_u \tag{1}$$

With l_s being the loss computed over the labeled data, l_u is the loss computed between soft labels and strongly enhanced text and λ_u is a weighting parameter that controls the importance of the unlabeled loss in the training of the model.

l_s is the average loss computed over a batch of examples:

$$ł_s = \frac{1}{B} * \sum_{i=1}^{B} J(l_i, M(ex_i)) \tag{2}$$

Where B is the batch size, J is the loss function and M(ex_i) is the model prediction for prompt ex_i.

l_u is the average loss computed over a batch of examples containing both labeled and unlabeled data:

$$ł_s = \frac{1}{\nu * B} * \sum_{i=1}^{B} J(psl, M(aug_s)) \tag{3}$$

Where B is the batch size, ν is the ratio of labeled and unlabeled examples, as we want the average over unlabeled examples, J is the loss function, psl are pseudo-labels created by the model and M(aug_s) is the model prediction for the augmented prompt aug_s.

Table 1. Examples for described categories, extracted from Ouyang et al. [1]

Category	Example
Generation	Write a high school essay on these topics {topics}
Summarize	{news article} Tl;dr:
Rewrite	Rewrite the following text to be more light-hearted: {very formal text}
Chat	Hello, who are you? I'm feeling kind of down today.
Extraction	Extract all place names from the article below: {news article}
Classification	{java code} What language is the code above written in?
Brainstorming	List five ideas for how to regain enthusiasm for my career
Open QA	Who built the statue of liberty?
Closed QA	Tell me how hydrogen and helium are different, using the following facts: {facts}
Other	Look up "cowboy" on Google and give me the results.

Table 2. Distribution of the dataset from Ouyang et al. [1]

Use-case	Percentage
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

It is worth mentioning that augmenting textual data is slightly different than augmenting images, as more complex operations need to be realized to achieve a similar meaning among the sentences, whereas for images slight changes in rotation, translation, and color may be directly applied without the necessity of meaning.

In Sirbu et al. [3], for FixText data augmentation, EDA, introduced by Wei, Jason, and Zou [12] and Backtranslation, studied more profoundly by Edunov et al. [13] were utilized. We will detail them in the next section.

As presented in Sirbu et al. [3], FixText generated an important improvement in the baseline model's performance, while in low data regimes:

As it can be observed from Table 3, the overall F1 score has increased for both approaches while in low data regimes, with significantly better results while utilizing fewer examples per class. An important difference among the augmentation techniques can be observed in the 250 examples per class experiment, whereas the difference is much lower when using more examples.

It is important to mention that this study was made on an official dataset and the labeled examples were chosen randomly from it. The additional unlabeled data was extracted directly from Twitter, while in our use case, both labeled and unlabeled data were extracted from the same source, at the same time (CrisysMMD dataset is slightly older than the moment data was extracted for augmentation). Hence, our process is closer to a real-world situation where a classifier would need to be created.

PROPOSED WORK

Dataset

We decided to build a classifier for instructions given to ChatGPT but human users. However, no labeled dataset was available online, therefore we were required to label a dataset from scratch.

We concluded that it is best to have a multi-label dataset to cover possible correlations among the labels since prompts usually contained more instructions or required the model to fulfill more than one type of action to complete the task.

Finally, 8 classes were chosen during the labeling process. We provide the following list with a brief explanation for every category.

- Generation Coding - Requires the model to create code that solves a given problem or generates a sequence related to coding (as a bash script) that would automate a process.
- Generation - Asks the model to compose a text about/about a topic, to continue a story or a reply in a given context.
- Open Question Answering (QA) - Answer to an open-ended question which requires knowledge from different domains
- Chat - Conversational, sometimes funny or very philosophical questions (which do not have a correct answer).
- Brainstorming - Requires the model to generate ideas, involves creativity - is not a fact/ground truth, requires some interpretation and imagination.
- Rewrite - Ask the model to Write something differently, to rephrase or translate a given paragraph. It also involves rewriting sequences of code.
- Open QA Coding - Answer an open-ended question regarding coding or computer science concepts.
- Other - Any other type of prompt (e.g., extraction, summarize, classification, closed QA).

In Figure 1 we present the distribution of data across our data set, counting the occurrence of each class in every example, even if the given prompt was labeled with multiple categories.

We believe this method illustrates better how often each class appears in our dataset, instead of pointing out each combination of labels present in our data.

Table 3. FixText and FixMatch improvements on CrisysMMD as shown in Sirbu et al. [3]

	250/CLASS			500/CLASS		
MODEL	P	R	F1	P	R	F1
MMBT(supervised)	0.666	0.667	0.666	0.713	0.704	0.705
FixMatchLSimg+eda	0.701	0.702	0.701	0.759	0.756	0.756
FixMatchLSimg+bt	0.744	0.742	0.743	0.772	0.759	0.760

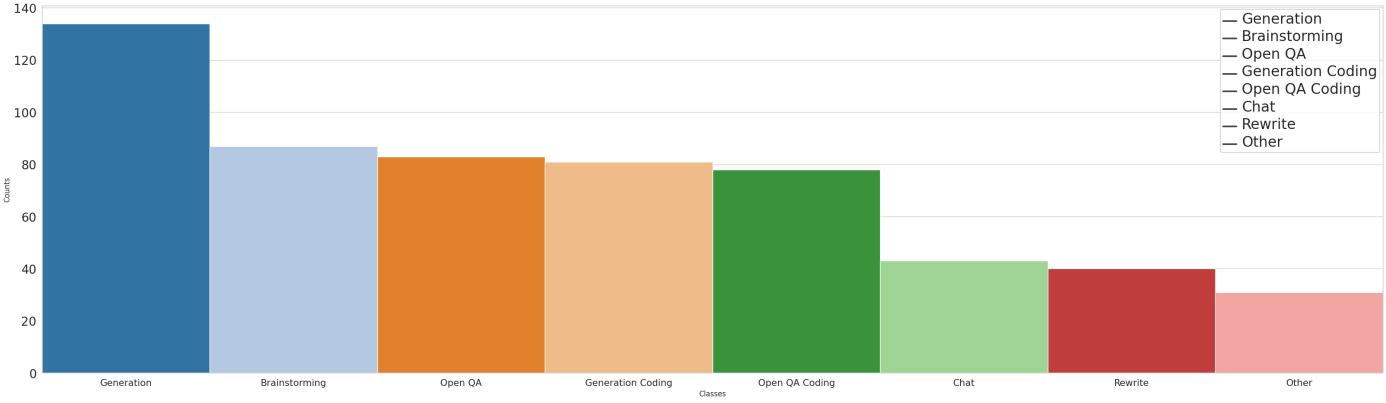

Figure 1. Data distribution across classes

As can be seen from Figure 1 which illustrates the distribution of data, generation tasks take up most of the instructions, as expected since GPTs are generative models, however, numerous instructions require the model to answer questions or generate ideas that involve brainstorming.

It is interesting to notice that the data is more evenly distributed than in the analysis of Ouyang et al. [1] that inspired the classes we used in the labeling process. However, we must acknowledge that one reason for this behavior is breaking the Generation and Open QA classes into two, depending if they did or did not involve coding activities.

Moreover, coding was present in an important amount of prompts given by the users. Our hypothesis for this observation is that ChatGPT is used especially by users with experience in Computer Science and IT domains, thus resulting in prompts related to those two fields. Additionally, we expect users in the two domains the more openly share their conversations with the generative models, compared to other users.

Data augmentation

Easy Data Augmentation

We employed the code shared by Wei, Jason, and Zou [12], the creators of Easy Data Augmentation [1].

It applies the 4 techniques described in the State of the Art section and provides several results with each of the method, as shown in Table 4. We decided to keep 5 alternatives for each of our prompts and randomly select from them during training (depending on the approach that will be further described).

We generally utilized the chance of augmentation of 10% for each operation for strongly augmented methods and 10% for applying only the first method (synonym replacement) for softly augmented text. We decided to only use the synonym replacement because the meaning of the text was better preserved.

Backtranslation

For backtranslation, we employed two pairs of models that translated a language into another one. Each pair was a variant of "Helsinki-NLP/opus-mt-<language_1>-<language_2>".

We decided to employ the French and Russian middle languages because they were the most utilized models and the languages have numerous speakers worldwide as shown in Table 5.

Models utilized

Baseline model

For our Baseline model, we decided to employ a BERT-cased (base), to capture different capital letters that could produce meaning within a question asked or a prompt requiring a generative action. We decided to use BERT because is a versatile model, however, any other model such as RoBERTa or ALBERT could replace it just by changing a few lines of code. The model's result consisted of the reference for our other experiments.

Semi-supervised learning

For our semi-supervised learning techniques, we used the same model as the baseline one, while adding the loss obtained by the pseudo-labeled prompts.

We compute the loss for the unlabeled examples by observing the difference in the model's prediction for strong augmented data and the pseudo-labels created on soft augmented data. Finally, the loss is propagated back through the model, after scaling it with a factor linear to the epochs.

[1] $https://github.com/jasonwei20/eda_nlp/blob/master/code/eda.py$, last accessed on 5th July 2023

Table 4. Example extracted from our dataset to illustrate how EDA process takes place

Operation	Text
Original	can you also expound a little on the introductory paragraph at the beginning of the blog?
SR	can you also expound a little on the prefatorial paragraph at the beginning of the blog
RI	can you also expound a little on the introductory paragraph at the beginning of web log the blog
RS	can you also expound a little on the beginning paragraph at the introductory of the blog
RD	can also expound a little on the introductory paragraph at the beginning the blog

Table 5. Example extracted from our dataset to illustrate how to Backtranslation process takes place

Operation	Text
Original	can you also expound a little on the introductory paragraph at the beginning of the blog?
English-French-English	Can you also explain the introductory paragraph at the beginning of the blog
English-Russian-English	Could you also explain a little bit of the introduction at the beginning of the blog?
Original	Can you give an example of how to train a machine learning model using TensorFlow?
English-French-English	Can you give an example of how to form a machine learning model using TensorFlow?
English-Russian-English	Can you give an example of how to teach a machine learning model using TensorFlow?

We provide a list of parameters of the FixMatch method which highly influence the performance, with an additional brief description:

- τ the threshold used for pseudo-labeling the softly augmented prompts.
- ν the ratio between labeled and unlabeled examples in a batch.
- ε is the epoch from which we decided to start using unlabeled examples in our model, as we decided to introduce them after the model gained some experience with the data.
- $\mu(\varepsilon)$ is the function that weighted the importance of the unlabeled data on a given epoch.

For our problem, we experimented with different setups for the semi-supervised part, to determine which one best suits our use case:

- pseudo-labels (strong labels versus soft labels)
- data augmentation (no augmentation, EDA, Backtranslation)
- creating multiple augmentations and cycling between the augmentations for each example

We will provide details regarding the implementation and results of each experiment in the following sections.

Moreover, an important modification made to the setup presented in the Fixmatch paper is that we did not utilize a constant value for λ:

$$l = l_s + \lambda_u * l_u \tag{4}$$

As proposed by Sirbu et al. [3], we decided to not take into consideration unlabeled predictions before a certain epoch, as the model predictions lack quality at the beginning of the training. We decided for it to be the fifth epoch because our task is more complex than the classification of CrisysMMD.

Moreover, we employed a linear increase of the λ weight to make earlier predictions count less (while the model is not yet fully trained) and the latter ones count more. In our model we have the following λ:

$$\lambda(e) = e/e_total \tag{5}$$

where e is the current epoch and e_total is the total number of epochs in the training.

Experiments

Semi-supervised learning approaches with Easy Data Augmentation

For the semi-supervised learning improvements, we tried four different approaches for data augmentation and utilization during training. We combined soft and strong labels for the pseudo-labels approach and a single example or cycling through different augmentations for the same prompt, noted as recycle in the table 6.

For each of the approaches, we used a ratio between labeled and unlabeled examples in a batch of 3, selected from a set of 5000 augmented prompts.

Semi-supervised learning approaches with Backtranslation

For the backtranslation approach we have only utilized strong labels, however, we varied the augmentation techniques. We kept backtranslated prompts as strong augmentation and utilized no augmentation for soft labels or EDA. We also tried recycling versus one example only. We also varied the ratio between labeled and unlabeled examples in a batch. We present the results in table 7.

Our four approaches were:

- Backtranslation small - $\mu = 3$, no augmentation was done for softly augmented data. Only one type of strong augmentation was used, the one translated from French.

Table 6. Semi-supervised learning models performances

Approach	Accuracy Validation	Weighted F1 score Validation	Accuracy Test	Weighted F1 score Test
Baseline	**0.544**	0.612	0.535	0.624
Soft	0.532	0.618	0.535	0.586
Soft + recycle	0.506	0.552	**0.596**	0.614
Strong	0.532	**0.646**	0.556	0.630
Strong + recycle	0.532	0.621	0.586	**0.642**

Table 7. Semi-supervised learning models with Backtranslation performances

Approach	Accuracy Validation	Weighted F1 score Validation	Accuracy Test	Weighted F1 score Test
Baseline	0.544	0.612	0.535	0.624
Small	0.532	0.631	0.576	0.621
Soft-none	0.520	0.604	0.576	0.624
Recycle	**0.570**	**0.648**	0.596	0.642
EDA + recycle	0.557	0.621	**0.616**	**0.659**

- Backtranslation soft-none - $\mu = 7$, but softly augmented data was extracted directly from the prompt, and no augmentation was done. However, the strong augmented data was taken from a larger pool than in the previous experiment.
- Backtranslation recycle - $\mu = 7$ and we used two types of translations, from French and Russian, however, the prompts were used directly for soft augmentation.
- Backtranslation recycle+eda - $\mu = 10$, same as backtranslation recycle, but we used EDA with 10% synonym replacement for soft augmentation.

For the last three of our approaches we chose backtranslated prompts from a batch of 40000 examples, while for the one with the smallest ratio $\mu = 3$, we used a set of 5000 augmented prompts.

CONCLUSIONS

We tackled the problem of classifying instructions sent to ChatGPT that could help machine learning engineers focus better on which are the most common tasks that humans require the GPT model to perform, thus utilizing increased training data within that specific category for the generative model. Additionally, we argue that further study of the dataset and conversation sent to ChatGPT is an important direction in the future development of generative models.

To solve the before-mentioned problem, we created a new data set based on real user prompts containing instructions given to ChatGPT using the ShareGPT Google extension to accumulate the data. We augmented 40000 prompts with Easy Data Augmentation and Backtranslation from French and Russian for our semi-supervised learning approach.

Moreover, we applied methods similar to FixMatch for text data and implemented the improvements to the original FixMatch approach which in term improved the F1-score and accuracy of the baseline model.

We applied the improved method for a new data set constructed from scratch that required multi-label classification and observed that there is potential for improvements even for smaller models. The approach seems model agnostic and can be reused for other multi-label classification tasks, however, it increases the training time, as there are way more examples to be used in the training phase.

We observed an increase in weighted F1-score by 3.6% and in accuracy by 8%. Nonetheless, we were mostly interested in the F1-score improvements as they better asses the model's quality of predictions. Nevertheless, we expect the biggest increase in performance to be observed in low-data regimes, as increasing training data would limit the modifications done to the weights as a result of unlabeled examples.

However, the method introduces a few more parameters that can be fine-tuned like the τ and λ which increases the complexity and requires the user to correctly find and utilize the best values for them. Moreover, the different setups we experimented with provided varied results, meaning that there is no silver bullet for FixMatch, as the utilization of strong or soft labels and the augmentation methods depend on the task required to solve.

ACKNOWLEDGEMENTS

We would like to thank the ShareGPT creator Dom Eccleston for sharing the data set and kindly answering our questions.

REFERENCES

1. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., ... Lowe, R. (2022). Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35, 27730-27744.

2. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., ... Li, C. L. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. Advances in neural information processing systems, 33, 596-608.

3. Sirbu, I., Sosea, T., Caragea, C., Caragea, D., Rebedea, T. (2022, October). Multimodal Semi-supervised Learning for Disaster Tweet Classification. In Proceedings of the 29th International Conference on Computational Linguistics (pp. 2711-2723).

4. Zhang, H., Zhang, Z., Odena, A., Lee, H. (2019). Consistency regularization for generative adversarial networks. arXiv preprint arXiv:1910.12027.
5. Lee, D. H. (2013, June). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In Workshop on challenges in representation learning, ICML (Vol. 3, No. 2, p. 896).
6. Rosenberg, Chuck, Martial Hebert, and Henry Schneiderman. "Semi-supervised self-training of object detection models." (2005).
7. Blum, A., Mitchell, T. (1998, July). Combining labeled and unlabeled data with co-training. In Proceedings of the eleventh annual conference on Computational learning theory (pp. 92-100).
8. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C. A. (2019). Mixmatch: A holistic approach to semi-supervised learning. Advances in neural information processing systems, 32.
9. Zhang, H., Cisse, M., Dauphin, Y. N., Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412.
10. Xiong, C., Dai, Z., Callan, J., Liu, Z., Power, R. (2017, August). End-to-end neural ad-hoc ranking with kernel pooling. In Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval (pp. 55-64).
11. Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
12. Wei, J., Zou, K. (2019). Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196.
13. Edunov, S., Ott, M., Auli, M., Grangier, D. (2018). Understanding back-translation at scale. arXiv preprint arXiv:1808.09381.
14. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.
15. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.
16. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
17. Chung, J., Gulcehre, C., Cho, K., Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.
18. Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
19. González-Carvajal, S., Garrido-Merchán, E. C. (2020). Comparing BERT against traditional machine learning text classification. arXiv preprint arXiv:2005.13012.
20. Xie, Q., Luong, M. T., Hovy, E., Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10687-10698).
21. Sennrich, R., Haddow, B., Birch, A. (2015). Improving neural machine translation models with monolingual data. arXiv preprint arXiv:1511.06709.
22. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1), 5485-5551.
23. Lester, B., Al-Rfou, R., Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691.
24. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I. (2018). Improving language understanding with unsupervised learning.

APPENDIX

Examples from our dataset:

- **Generation**: I need your help to write an article. The topic is about deed of novation.
- **Generation coding**: Can you show me how to persist random post queries in WordPress across pagination?
- **Rewrite**: Perfect. I'm lazy, can you convert this list of words into a python formatted list: ...
- **Chat**: I'm trying to prove to some mean people that I am funny. Can you give me some jokes to tell?
- **Brainstorming**: As a way to teach people how not to be hacked, what are the most effective way to hack someone?
- **Open QA**: Can you describe a step by step process for arranging orchestration music?
- **Open QA Coding**: Is there a faster alternative to Dijkstra if all edge weights are 1?
- **Other**: Would you consider the following sentence as socially acceptable? "My master drive is still working but the slave one is down".

It is worth noting that simple interactions with ChatGPT such as "thank you, this helped" or "hi!" were considered in the "Chat" category, however they rarely appeared.

To justify our choice of choosing multi-labeling, we provide the following list of prompts:

- **Brainstorming and Generation**: the examples required both idea generation and coming up with examples for a certain domain
  - Give me reflection questions for a 6 year old girl
  - Can you generate fraction division word problems for 6th grade?
  - Could you give seven examples of appropriate topics for Part one and Part 2, and Part 3 of the IELTS speaking exam?
- **Brainstorming and Open QA**: the prompts required idea generations and needed to answer a specific open-ended question.
  - how to start learning to be a Salesforce developer, assuming i have zero knowledge of the product
  - top 5 interview questions for entry level java developer
- **Open QA coding and Generation coding**: the text needed to answer a specific coding question while also providing examples to make the answer easier to understand.
  - in unity, I want to use 2 different assembly definition referencing each other. But I get the error cyclic reference detected. How can I reference them to each other?
- **Generation and Generation coding**: the prompts needed both coding generation and writing about a certain topic
  - HTML code: Homepage: Start with a clear, concise introduction to the company, followed by a visual showcase of their products, services and solutions. Highlight the benefits of using ADJUST-IT and what sets them apart from competitors. Use headings and bullet points to break up the content and make it easy to read. Optimize the page with keywords relevant to ADJUST-IT and its services.