Strategies for Building Semantic Classification Dataset with LLM and Active Learning

Stanford CS224N **Default** Project

Jerry Chan
Department of Computer Science
Stanford University
jerryyc@stanford.edu

Abstract

In this study, we explore methods for collecting and augmenting training data using Large Language Models (LLMs) and active learning. We analyzed the effect of incorporating LLM-generated synthetic data into the training set of fine-grained semantic classification on Stanford Semantic Treebank-5 Socher et al. (2013). We observed that randomly generating samples from LLM leads to only a slight improvement and even hurts the performance when the proportion of synthetic data increases. This motivates us to further investigate strategies for collecting and labeling training data. Our experiments show the challenges of using LLM-generated synthetic data on fine-grain semantic classification problems: inaccurate labels and differences in data distribution. We further show that LLM-generated data can still be useful if a more accurate label is available. We also proposed two methods to efficiently label unlabeled data based on active learning from model uncertainty and query by committee, with model uncertainty outperforming the baseline.

1 Key Information to include

- Mentor: Sonia Hangjie Chu
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction and Related Work

High-quality, annotated data can be hard to obtain, but with the advent of Large Language Models (LLMs), one can leverage their few-shot learning ability to create large amounts of synthetic data. LLM data generation is much cheaper than human-created datasets or those from data brokers and can be generated more quickly with an easily accessible language interface. This motivates us to investigate if we can apply the method to fine-grain sentiment classification tasks and potential ways of improving the LLM data generation process.

Researches show LLM data augmentation successful in specific tasks with limited labeled data or unbalanced datasets (Chintagunta et al. (2021), Li et al. (2022), Wan et al. (2022)). However, works that compare different LLM data augmentation paradigms Bansal and Sharma (2023), Ding et al. (2024) show that randomly generated samples might lead to worse performance and generalization on certain tasks and waste unnecessary LLM query cost and compute.

In optimizing the data collection process, the field of active learning focuses on improving model performance by selectively choosing the most informative data points for labeling (Settles (2009)). Unlike traditional supervised learning, which relies on large, randomly sampled datasets, active learning aims to reduce labeling costs and improve efficiency by querying only the most uncertain

or informative samples. Key strategies in active learning include uncertainty sampling (Lewis and Gale (1994), Cohn et al. (1994)), where the model selects data points with the highest prediction uncertainty, and Query by Committee (QBC), which involves multiple models and selects samples based on the level of disagreement among the models (Freund et al. (1997)). Some more recent works like Sener and Savarese (2017) and Kirsch et al. (2019) demonstrated success in applying active learning strategies to modern deep learning methods.

SQBC by Wagner et al. (2024) combines LLM data synthesis with active learning and looks into ways to minimize data labeling costs while improving performance by utilizing active learning strategies. SQBC inspired us to investigate its applicability and similar active learning approaches on SST-5.

We started with naive approaches to LLM data augmentation and examined the effectiveness of fine-tuning the model on those data. We then tried to relabel the LLM-generated data with more accurate labels. On active learning, we applied a model uncertainty approach and SQBC on SST-5 and compared them with the random sampling baseline.

3 Approach

The baseline model with the provided dataset alone already performs very well on the paraphrase detection and semantic textual similarity tasks while the accuracy on the sentiment analysis task remains low. Furthermore, we observed a significant overfitting problem while training the baseline model on sentiment analysis compared to the other tasks despite regularization measures, indicating great potential improvement with data augmentation. Thus, we decided to focus the data augmentation methods and experiment design on improving the performance of the sentiment analysis task. The baseline approach of the paraphrase detection and similarity detection will still be detailed in the baseline section.

3.1 Baseline - Default Project

The baseline model was built on top of the 12-layer BERT model provided with the default project. For each task, a different decoder output head is added on top of the pooler output of the base BERT model, containing a dropout layer with a dropout rate of 0.5, one linear layer with a size corresponding to the dimension of the desired prediction, and a non-linear layer (Softmax, Sigmoid) to transform real number output to the space of the label if needed.

For semantic analysis, the tokenized input sentence is directly passed to the model without any preprocessing. For paraphrase detection and semantic textual similarity, the input data contains two sentences. We concatenate the two input sentences, separated by *[SEP]* token, and merge the attention mask correspondingly. This allows the model to attend to tokens of both sequences in every layer. In our experiments, concatenating the input at a sentence level outperforms passing the two sentences through Bert separately and combining the embeddings before the decoder head, especially on similarity detection. Notice that the downside of this approach is the doubling of the input sequence length. Considering the transformer's quadratic complexity growth rate with respect to input size, the performance-complexity trade-off might not be justifiable with longer input sequences.

3.2 LLM Data Augmentation

We prompted the LLM with minimal guidance, a simple task description, and 3 randomly selected examples from each class and asked it to generate a fixed number of labeled test data. The prompt is included as following:

> system: Your task is to generate training data for a movie review sentiment classification model. Format your output as a JSON list of tuples with a sentence and its sentiment as an integer. 0 is negative, 1 is somewhat negative, 2 is neutral, 3 is somewhat positive, and 4 is positive.
> user: Here are some examples: EXAMPLES_IN_JSON_FORMAT
> user: Generate NUMBER_OF_SAMPLE samples

Considering the cost of LLM inference, we chose to use OpenAI's *gpt-3.5-turbo* as the data augmentation engine. We prompt the language model to generate 100 samples in JSON format and repeat the process until we obtain sufficient synthetic data. We noticed that *gpt-3.5-turbo* doesn't follow the instructions completely and often fails to generate any data or tries to generate more data, resulting in an incomplete response due to the context window size limit. Some parsing heuristics are applied to the response to ensure the quality of the generated dataset.

3.3 Relabel LLM Generated Data

When doing exploratory error analysis on the performance of our baseline model, we found that the model struggles the most with classifying between "somewhat positive" and "positive" and between "somewhat negative" and "negative". This highlights the importance of accurate labeling of the synthetic data. Therefore, we are curious about the performance gain on LLM augmented data if they have ground truth labels, such as those annotated by humans, which are not accessible in the scope of this project. We approximated human labels by labeling the data with a model on all training data and validated it on the development set.

3.4 Data Selection Problem Setup

From the experiment with LLM-generated data in SECTION, we learned that multiple challenges in LLM-generated data can jeopardize the effectiveness of incorporating synthetic data. We want to further study the question of data selection by filtering out the effect of the difference in data distribution.

In this and the following sections, we assume access to a limited labeled dataset $D_{labeled}$, a validation dataset D_{val}, and a larger unlabeled dataset $D_{unlabeled}$. The goal is to maximize the performance gain on D_{val} by selecting data from $D_{unlabeled}$ to be labeled and trained on while minimizing the labeling cost.

In our experiments, we use a portion of labeled data as $D_{unlabeled}$. When selecting the data from $D_{unlabeled}$, we didn't use any information on the label of $D_{unlabeled}$. The label is only accessed when it's selected to be used in training. This mimics collecting additional data from the true data source and querying the same labeling method used to label D_{val}.

3.5 Data Selection - Uncertainty Sampling

Uncertainty Sampling is a popular strategy in active learning. The method selects the samples that the model is least certain about for labeling. This approach aims to maximize the information gained from each labeled example, theoretically improving model performance more efficiently compared to random sampling. We trained a model on the $D_{labeled}$ and ran prediction on $D_{unlabeled}$ before computing the entropy $H(\theta; X)$ of the model's output with

$$H(\theta; X) = \sum_{c \in C} q(\theta; X, c) \log(q(\theta; X, c))$$

where $q(\theta; x, c)$ is the model's prediction of input X belonging to category c. We ranked $D_{unlabel}$ with $H(\theta; X)$ and prioritized labeling the samples with higher entropy.

3.6 Data Selection - Query by Committee

Another popular method is Query by Committee, which involves maintaining a diverse set of models (the committee) and selecting samples for which the committee members disagree the most. This disagreement indicates uncertainty, and labeling these samples can provide valuable information to improve the overall model performance. We adopt the method of Synthetic Data-driven Query By Committee proposed in CITE. For each sentence in $D_{unlabeled}$, we searched for the K most similar sentences in $D_{labeled}$ and computed the standard deviation of the labels on those sentences. The standard deviation serves as an indicator of the disagreement between the labels.

In practice, we first embedded sentences from $D_{labeled}$ and $D_{unlabeled}$ to a high dimensional latent space with a BERT fine-tuned on $D_{labeled}$. We then fit a K-nearest neighbor classifier with $D_{labeled}$.

We use the KNN classifier to find the label of nearest neighbors for data in $D_{unlabeled}$ to calculate the standard deviation.

4 Experimental Setups

4.1 Data

We use the SST-5 dataset provided with the default project. The dataset contains single-sentence movie reviews as input and five sentiment categories as output: "negative", "somewhat negative", "neutral", "somewhat positive", and "positive".

We split the dataset randomly into $D_{labeled}$ and $D_{unlabeled}$ with a one to four ratio, the $D_{labeled}$ contains approximately 1700 samples and $D_{unlabeled}$ contains approximately 6800 samples. For the validation set D_{val}, we use the labeled dev split provided with the default project with about 1000 samples.

4.2 Evaluation method

Our experiments look at how effectively the increment in the amount of training data contributes to performance gain on D_{val}. For each approach, we add different amounts of additional training data, LLM-generated or selected from $D_{unlabeled}$, to the training set with $D_{labeled}$, train three models on it with different random seeds, evaluate the models' accuracy on D_{val}.

4.3 Baseline

A baseline we compare our approaches to is randomly sampling data from $D_{unlabeled}$. This is equivalent to randomly collecting an unlabeled dataset from the true data source and labeling the data with the same labeling method as D_{val}.

4.4 Training Configurations

We followed the model configuration described in Section 3.1. We use Adam optimizer with βs of (0.9, 0.999), a learning rate of 0.001, and a learning rate scheduler that decreases the learning rate by 0.5 every 20 steps. We train the model for 50 epochs with different fine-tuning methods:

- **Last-Linear**: finetune the last linear layer that maps BERT pooler output to the multiclass categorization label.
- **Pooler**: finetune the last linear layer and BERT's linear layer right before pooler output.
- **Last-Attention**: finetune all the last linear layer, pooler linear layer, and the last attention module.

We chose not to experiment with tuning more than one attention module to make the result less noisy. We observed that the model already overfits completely to the entire training split with **Last-Attention** method even with heavy regularization measures. Tuning more parameters makes it even easier to fit the training data, and under such a situation, it's hard to measure the effectiveness of adding new training data as the model can just memorize all of them and not generalize to the evaluation. Fine-tuning with $Last - Linear$ setting also results in less representable results as the model doesn't have enough complexity to fit to the training data. In our experiments, Fine-tuning with **Last-Linear** is only able to achieve around 0.5 accuracy on $D_{labeled}$.

5 Experiments

5.1 Default Project Test Leaderboard Results

5.2 LLM Generation

In this experiment, we want to analyze the effectiveness of incorporating LLM-generated data into the training data and the performance difference between training on LLM-generated labels and the label generated by the trained model.

Task	Sentiment Analysis	Paraphrase Detection	Similarity	Overall
Score on Dev Split	0.509	0.904	0.867	0.782

Table 1: Baseline Performance on Test Split.

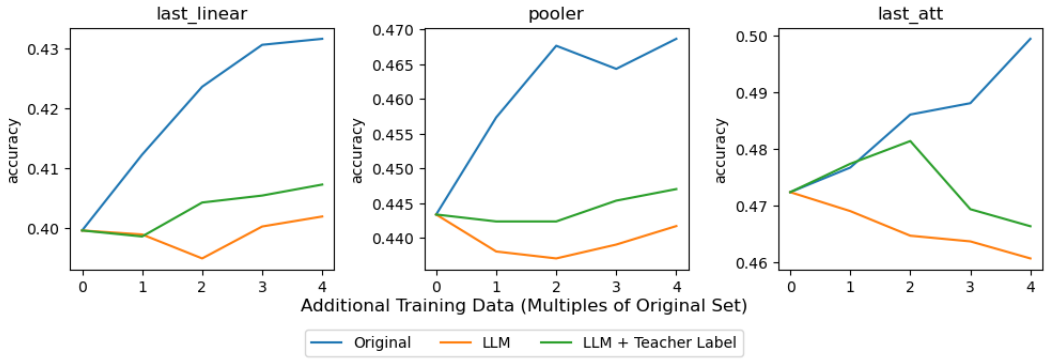

Figure 2: **Comparison Between Training with LLM Augmented Dataset and Original Dataset Across Different Finetuning Methods.**: The x-axis represents the amount of training data in addition to $D_{labeled}$. x = 0 means the model is only trained on $D_{labeled}$. x = 4 means the model is trained on $D_{labeled}$ and additional data 4 times $|D_{labeled}|$. The blue line: sampling additional data from the hold out set of the original data. The orange line: sampling additional data from D_{llm}. The green line: sampling additional data from D_{llm_t}. Each data point is average among 3 runs.

We generated 8000 rows of labeled data D_{llm} with the method described in Section 3.2. We relabel the data with a teacher model from Section 3.3 and note the relabeled dataset D_{llm_t}. The two labels only agree on **61.3%** of the data, showing potentially high discrepancy between the LLM-generated label and the true label. A confusing matrix between the LLM-generated label and teacher model-generated label is shown in Figure 1. We can see that the labels highly disagree on labels between two neighboring sentiment categories. This shows that GPT3.5 is struggling to generate accurate labels on fine-grain sentiments with only few-shot prompting

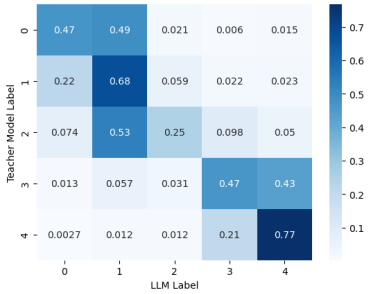

Figure 1: Confusion Matrix Between LLM Generated Label and Teacher Model's Label

Both datasets are randomly split into 4 subsets. We mix the different amounts of split with $D_{labeled}$ to train models and observe the performance changes in the amount of splits mixed in. The result is shown in Figure 2.

From the result, we can see that directly using LLM-generated data in this task harms the model's performance. The model trained with additional LLM-generated data performs worse than the model trained on much less data drawn from the original data distribution. We also observed that relabeling the data with a teach model that approximates the true label generally improves the effectiveness of the data augmentation process. The model shows performance gain trained on relabeled LLM-generated data. With the fine-tuning method of Last-Attention, we even see the same performance gain as the model fine-tuned on additional data drawn from the original data distribution. We also notice the performance drop when the ratio of simulated data surpasses 1/3. Additional experiments are

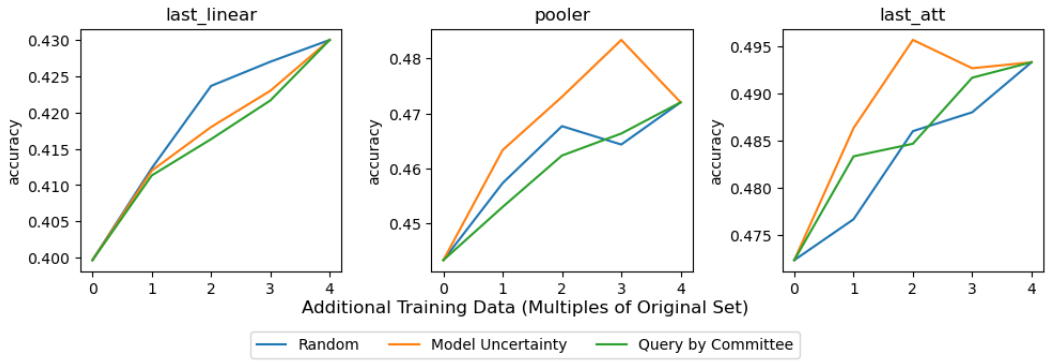

Figure 3: **Data Selection Method Comparison Across Different Finetuning Methods.**: The x-axis represents the amount of training data in addition to $D_{labeled}$. x = 0 means the model is only trained on $D_{labeled}$. x = 4 means the model is trained on $D_{labeled}$ and additional data 4 times $|D_{labeled}|$ drew from $D_{unlabeled}$ with the corresponding data selection method. The blue line: Randomly sample data from $D_{unlabeled}$. The orange line: Select data from $D_{unlabeled}$ by the Model Uncertainty method. The green line: Select data from $D_{unlabeled}$ by the Query by Committee method. Each data point is average among 3 runs.

needed to pinpoint the cause of this phenomenon. More discussion on possible reasons is addressed in Section 6.2.

Notice that the label from the teacher model is far from perfect. As we learned from training the default project baseline, the model trained on this task typically only has 50 55% accuracy. It is very likely that if provided with the true label, LLM-generated data can be much more effective.

5.3 Data Selection Strategies

This experiment compares the different data selection methods from Section 3.5 and Section 3.6 across different fine-tuning methods. We calculate the entropy and disagreement (standard deviation of labels of 5 nearest neighbors) of each instance in $D_{unlabeled}$. The correlation between entropy and disagreement is only **0.161**, indicating the two metrics are very different.

We separately ranked $D_{unlabeled}$ using the model uncertainty approach and the query by committee approach. We sorted $D_{unlabeled}$ by the ranking and split them into 4 equally sized subsets. The first split contains samples ranked the most effective by the selection method while the last split contains the least. We trained models on $D_{labeled}$ plus different amounts of splits from $D_{unlabeled}$, splits marked with higher effectiveness always got mixed in first. We repeated this experiment with different fine-tuning methods. Results are presented in Figure 3.

As mentioned in Section 4.4, when fine-tuning only the last linear linear, the model doesn't have enough complexity to fit to the training dataset. Across all methods and all training data configurations, the model fails to achieve even 60% accuracy on the training set, therefore explaining the seemingly random result shown with the Last-Linear finetuning method.

With finetuning methods that can fit the training data, the Query by Committee method performs similarly to random sampling. The Model Uncertainty method outperforms Query by Committee and random sampling as the model trained on data selected by Model Uncertainty achieves higher performance gain.

Notice that all methods have the same accuracy when trained on additional data 4 times the original set. This is because $|D_{unlabeled}| = 4|D_{labeled}|$, so no matter the selection method, selecting $4|D_{labeled}|$ samples from $|D_{unlabeled}|$ will exhaust the dataset and lead to the same training data.

6 Conclusion and Analysis

6.1 Reflection on Experiment Design

During this project, we noticed that overfitting poses a severe challenge to studying the relationship between data and performance. The training set only contains 8500 samples while a 12 layers BERT has 110M parameters. The last 2 layers of the classifier: the linear layer and pooler dense layer, have about 768 * 5 parameters and 768 * 768 parameters respectively. Fine-tuning only the last linear layer doesn't have enough complexity to learn the task. Fine-tuning more layers is too complex and overfitting the training data. Ideally, we will want to redo these experiments with other fine-tuning methods like LoRA Hu et al. (2021) or conduct the experiments on tasks less prone to overfitting.

6.2 LLM Data Augmentation

Upon further analysis, we found that aside from the inaccurate labeling, LLM-generated data also has a very different distribution than the original data. In Figure 4, we embedded the sentences from both datasets with a pretrained BERT model and reduced the embedding to 2 dimension space. We noticed the LLM-generated data is less diverse and doesn't cover the distribution of the original data. In Figure 5, we compare the labels (sentiment) of the two datasets. The original data has a bimodal-like distribution centered at "somewhat positive" and "somewhat negative" while LLM-generated data concentrates on "positive" and "negative". It's unclear why GPT3.5 tends to generate "positive" samples while examples are provided equally. Together with inaccurate labeling, the distribution difference might explain the ineffectiveness of training on LLM-generated Data.

However, relabeling the LLM-generated data with an inaccurate teacher model still shows performance gain, showing the potential of using LLM-generated Data with human labeling or other accurate labeling methods.

In the future, it'll be interesting to look into data generation methods other than in-context learning or different prompting strategies that promote more accurate data distribution.

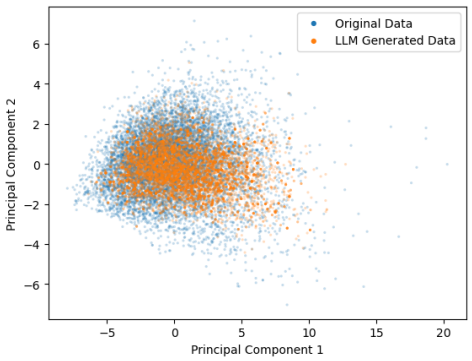

Figure 4: **Distribution Different Between Original Data and LLM Generated Data**: Original data and LLM generated data embedded to BERT latent feature space and projected to 2 dimensions by PCA for visualization

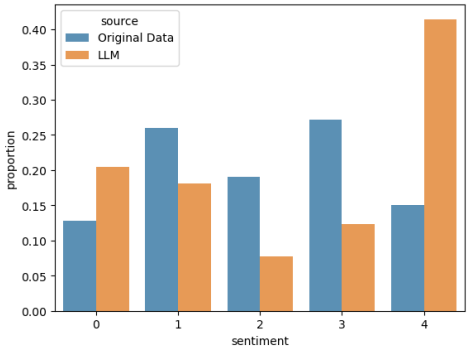

Figure 5: **Label Distribution of Original Data and LLM Generated Data**:

6.3 Data Selection

We show that Model Uncertainty can be a good indicator of the effectiveness of training on a data sample. This selection method doesn't require the additional data to be labeled, making it useful when iteratively developing an ML system to effectively prioritize labeling efforts. This approach can also be used to more efficiently label LLM-generated data, compensating for LLM's inability to generate fine-grain labels from just the examples.

7 Ethics Statement

The method poses two ethical concerns: (1) The LLM-generated data may inherit the bias of the LLM and the downstream model trained on those data will consequentially inherit the bias as well. (2) LLM might perform worse in the domain where data are underrepresented, leading to lower quality and less diverse data on underrepresented subjects and worse downstream performance on those subjects. The two concerns might lead to a biased model that perpetuates and potentially exacerbates existing inequalities and discrimination if it's being used in sensitive applications, such as loan applications and hiring processes.

Some practical strategies to mitigate those risks are evaluating the model with fairness metrics at every step of the process. For example, we might want to have additional labels for the sensitive attributes of the data points and make sure the model output and the model's performance are independent of those attributes. Furthermore, we also want to make sure the synthetic data generated by LLM gives equal coverage among those sensitive attributes.

References

Parikshit Bansal and Amit Sharma. 2023. Large language models as annotators: Enhancing generalization of nlp models at minimal cost. *arXiv preprint arXiv:2306.15766*.

Bharath Chintagunta, Namit Katariya, Xavier Amatriain, and Anitha Kannan. 2021. Medically aware gpt-3 as a data generator for medical dialogue summarization. *Proceedings of the Second Workshop on Natural Language Processing for Medical Conversations*.

David A. Cohn, Les E. Atlas, and Richard E. Ladner. 1994. Improving generalization with active learning. *Machine Learning*, 15:201–221.

Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia, Junjie Hu, Anh Tuan Luu, and Shafiq Joty. 2024. Data augmentation using llms: Data perspectives, learning paradigms and challenges. *arXiv preprint arXiv:2403.02990*.

Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine learning*, 28:133–168.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. 2019. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers.

Zekun Li, Wenhu Chen, Shiyang Li, Hong Wang, Jing Qian, and Xifeng Yan. 2022. Controllable dialogue simulation with in-context learning. *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach.

Burr Settles. 2009. Active learning literature survey.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Stefan Sylvius Wagner, Maike Behrendt, Marc Ziegele, and Stefan Harmeling. 2024. Sqbc: Active learning using llm-generated synthetic data for stance detection in online political discussions. *arXiv preprint arXiv:2404.08078*.

Dazhen Wan, Zheng Zhang, Qi Zhu, Lizi Liao, and Minlie Huang. 2022. A unified dialogue user simulator for few-shot data augmentation. *Findings of the Association for Computational Linguistics: EMNLP*.