Winogrande: leveraging chatGPT generated synthetic data via pseudo-labelling

Rianur Rahman **Paul Cho** **Ansar Choudhury**

Abstract

In this project, we focus on the challenge of pronoun resolution, a vital aspect of Natural Language Processing, which involves the correct identification of a referent for pronouns within a given context. We study the impact of generating synthetic data using a large language model, ChatGPT, in a semi-supervised setting to tackle the WINOGRANDE (Sakaguchi et al., 2019) pronoun resolution task. Our approach investigates the effect of this synthetic data on model performance by comparing results across various ratios of organic and synthetic data. With this project, we aim to examine the effectiveness of synthetic data augmentation to improve model performance, and thus we intend to provide insights into alternative data collection techniques and contribute to enhancing the performance of NLP models.

(Dai et al., 2023) (Yang et al., 2020) (Pluščec and Šnajder, 2023) (Sakaguchi et al., 2019) (Vaswani et al., 2017) (Liu et al., 2019) (Devlin et al., 2019) (Lee, 2013) (Radford and Narasimhan, 2018) (Ye et al., 2023) (Levesque et al., 2012) (Bras et al., 2020) (Win) (Ren et al., 2021)

1 Credits

We could credit winogrande repo here.

2 Introduction

The problem of pronoun resolution has been a significant challenge in natural language processing (NLP) for many years, where it has been difficult for an NLP model to determine the correct referent for pronouns in each context. This task is essential for understanding the deeper meanings of texts. In this project, we investigate the task of pronoun resolution using the WINOGRANDE XL dataset (Sakaguchi et al., 2019), from the paper, "WINOGRANDE: An Adversarial Winograd Schema Challenge at Scale". This is a large-scale dataset specifically designed to test the capabilities of NLP models in resolving ambiguity in pronoun references, and more precisely the Winograd Scheme Challenge (WSC) (Levesque et al., 2012).

The WINOGRANDE XL dataset (training size of 40,938 examples) is a collection of twin sentence samples designed to examine the capabilities of pronoun resolution in machine learning models. There are several data points, each with a sentence with an ambiguous pronoun and a set of potential referents. An example from this dataset may look like this: "The dog chased the cat because it was afraid." In this sentence, we know from our understanding that the word "it" refers to the cat, yet the model's task is to identify whether the correct referent is the cat or the dog.

Our motivation for choosing this problem is twofold. First, the authors of the WINOGRANDE paper highlighted the difficulty in collecting data for this task, where the data collection process involved generating adversarial examples to challenge the models' reasoning abilities. However, this process was done manually, by employing workers to create new Winograd Scheme instances. We intend to address this challenge by using a large language model, ChatGPT, to synthetically generate training data. Second, we want to investigate the effectiveness of this alternative data collection method in a semi-supervised setting using Pseudo-Labels (Lee, 2013). Our goal is to determine if similar performance can be achieved by relying more on synthetic data and less on the original dataset. Ultimately, we would like to experiment with the possibility of mainly using synthetic data, due to the easier data collection process. Furthermore as shown in the WINOGRANDE paper, performance increased with training set size thus we'd like to investigate whether this holds true in the case of synthetic data.

We expect to see an increase in performance when supplementing the organic dataset with our synthetic data, compared to solely using the organic data. We will test the performance with different ratios of organic and synthetic data to determine an optimal balance for achieving the best performance on the WINOGRANDE challenge. By analytically studying the influence of synthetic data on the model's performance, we hope to gain insights into the viability of using large language models such as ChatGPT for data augmentation and semi-supervised learning in NLP tasks.

3 Related work

WINOGRANDE Paper

We are using the WINOGRANDE paper (Sakaguchi et al., 2019) and their code as a starting point and are adapting and building our project from there. WINOGRANDE is a large-scale dataset of 44K problems inspired by the original WSC (Levesque et al., 2012) design, yet is accustomed to improving the scale and hardness of the dataset.

The authors collected the WINOGRANDE dataset using an adversarial approach. They used Amazon Mechanical Turk (AMT) workers to generate new Winograd Schema instances, where the workers used a language model to produce sentences which would be challenging for the AI systems to solve. The adversarial approach was in the fact that the workers were encouraged to produce sentences that the language model could not answer correctly. This process confirmed that the generated dataset was more robust while targeting the weakness of the AI models. They used an AFLITE (Bras et al., 2020) algorithm to generate adversarial examples, which could fool the model to make incorrect predictions on WSC. This algorithm works by iteratively modifying the input sentences to amplify the differences between the predicted and correct output. The authors discovered that their models were susceptible to these adversarial examples as their performance was significantly worse compared to the original examples, suggesting that the models are not as robust as they wished for.

The RoBERTa (Robustly Optimised BERT) model (Liu et al., 2019) used in this paper is a variant of the BERT model (Devlin et al., 2019), designed to improve performance on natural language tasks, which was pre-trained on a large corpus of text using a masked language modelling task. RoBERTa uses a larger training corpus and is trained for longer than BERT, hence leading to improved performance on many downstream NLP tasks. RoBERTa uses a similar architecture to BERT, which is based on the Transformer (Vaswani et al., 2017) design, yet the differences are in the training methodology. For example, RoBERTa has increased training steps, a larger set size, and a no next-sentence prediction task. The authors in the WINOGRANDE paper fine-tuned RoBERTa, which reduces training duration significantly compared to training a model from scratch.

RoBERTa uses a sub-word tokenisation approach, which uses the Byte Pair Encoding (BPE) algorithm for tokenisation, allowing for efficiency in handling rarer words. In RoBERTa, the input embeddings are a combination of position, token, and segment embeddings. We have borrowed the training optimisers(AdamW optimiser) and hyperparameters from the WINOGRANDE paper. The authors achieved leading results on the WINOGRANDE challenge, showing that the RoBERTa model attained an accuracy of 79.1%, whereas BERT achieved 64.9%. However, RoBERTa performed worse with the WINOGRANDE dataset compared to the original WSC dataset, establishing how robust the WINOGRANDE dataset is, along with how much more difficult it is. Nevertheless, RoBERTa's performance ranged from 59% to 79% when the size of the training data varied from 800 to 41k instances, therefore showing that to achieve human-level performance (94% accuracy), there would need to be over 118k training instances.

For our project, we also are using the AdamW optimiser as well as their code as a starting point, however, we are also including an additional dataset as well as a Pseudo-Labelling (Lee, 2013) functionality to devise labels for the unlabelled data as the main pillar of our experiment. As for our approach, we are utilising a semi-supervised approach as we have unlabelled synthetic data. We are investigating how the addition of this synthetic data affects performance.

Dataset Augmentation

Data augmentation is used to increase the size and variety of the training dataset, which helps to improve the generalisation ability of machine learning and NLP models. Data augmentation is useful when there is limited or unbalanced labelled data. By generating additional examples, the models better learn the underlying patterns, therefore reducing

the risk of overfitting. In NLP,and mentioned in the paper "Data Augmentation for Neural NLP" (Pluščec and Šnajder, 2023), data augmentation can be performed using techniques such as:

- Text paraphrasing: Rewriting sentences while preserving the meaning, which can be done by synonym replacement, rule-based approaches, or even neural paraphrase generation.
- Back-translation: Translating sentences to another language and then translating back to the original language which usually uses alternate phrasing while maintaining the original meaning.
- Generative approaches: Using generative models such as GPT (Radford and Narasimhan, 2018) to create samples inspired by the original data.
- Insertion, deletion, or swapping words.
- Masking: Replacing words/tokens with a mask token.

In the paper "Generative Data Augmentation for Commonsense Reasoning" (Yang et al., 2020), generative models are used to create new training samples to improve the model's performance on the CommonsenseQA dataset. In this paper, they made use of the GPT-2 model to generate synthetic data for WINOGRANDE. In the paper "AugGPT: Leveraging ChatGPT for Text Data Augmentation" (Dai et al., 2023), AugGPT is based on ChatGPT, to rephrase each sentence in the training sample to multiple similar, but semantically different samples. This is like our approach in this project where we adapted their approach of using ChatGPT and relabelled our synthetic data during training.

Data augmentation can help improve the performance of NLP models by increasing the training data size, therefore can enhance performance with unseen examples by learning the underlying patterns better. Also, augmentation can help with class imbalance. However, some of the common pitfalls and problems with dataset augmentation are that the quality of the augmented data may affect the original meaning, hence being detrimental to the model's performance. Computational cost is also a concern, as techniques such as back-translation and generative approaches can be computationally expensive. Also, some augmentation techniques might not be suitable for certain NLP tasks, or they may require tuning to have a useful improvement to the dataset. Another potential issue is over-augmentation which can result in over-fitting. Finally, some of these techniques which work for one task, may not be transferable to other datasets or tasks. Therefore, to alleviate these issues, the techniques chosen to augment the dataset must be carefully chosen and tuned based on the specific dataset and task.

In the AugGPT paper (Dai et al., 2023), a limitation and incorrect augmentation results can be due to the lack of domain knowledge of ChatGPT, however, the authors are planning to adapt the general-domain LLMs such as ChatGPT to domain-specific data, such as medical texts, via model fine-tuning, prompt engineering, knowledge distillation etc. In (Yang et al., 2020), they have shown that combining multiple data augmentation transformations can further improve a model's performance, as opposed to using individual transformations. This is due to the increased diversity, hence improving the model's generalisation. Although it may be infeasible to try all combinations of augmentation methods, there is promise in the automatic combination of dataset augmentation methods, such as TextAutoAugment (Ren et al., 2021).

Transformers

Transformers are a type of neural network architecture introduced in the paper "Attention is All You Need" (Vaswani et al., 2017). They were designed for NLP tasks and are now the basis for many leading models. Transformers are known for their highly parallelisable and efficient architecture, facilitating them to process large amounts of data and long-range dependencies.

Transformers are built on the concept of self-attention mechanisms, where the core components are the encoder and decoder. The encoder processes the input data, and the decoder generates the output. The encoder consists of a stack of identical layers, each containing two sub-layers: a multi-head-self-attention mechanism and a position-wise fully connected feed-forward network. The input sequence is passed through the attention mechanism, where the importance of each word relative to the others is weighed. Then it is passed to the feed-forward network. The decoder also consists of a stack of identical layers, but with three sub-layers: a multi-head self-attention mechanism, an encoder-decoder attention layer, and a position-wise fully connected feed-forward network. The decoder sim-

ilarly processes the sequence to the encoder, but with the encoder-decoder attention layer which provides the encoder's output to the decoder.

Transformers are so dominant in this literature due to their scalability, long-range dependencies, and parallelisation capabilities. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks require sequential processing of input data which is unlike Transformers, therefore, allowing for faster training and increased efficiency with using hardware. The attention mechanism is a vital component in the Transformer architecture, allowing the model to weigh the importance of the input element relative to others. It computes a weighted sum of input values based on their significance to a particular task, and this mechanism, therefore, grants Transformers the ability to ascertain dependencies and context in the input data, hence improving performance within NLP objectives.

BERT & RoBERTa

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model introduced in 2018 in the paper, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." (Devlin et al., 2019) BERT was designed to capture the deep bidirectional context in text for NLP tasks, including sentiment analysis, question answering and named entity recognition. BERT is based on the Transformer architecture and is pre-trained on two unsupervised tasks: masked language modelling (MLM) and next sentence prediction (NSP). The MLM task incorporates masking a percentage of the input tokens and then training the model to predict the masked token based on their context. The NSP task trains the model to predict if a given sentence follows another sentence. These tasks assist BERT to learn deep contextual representations from a large unannotated text corpus. Once pre-trained, a task-specific output layer is added and is trained for a few more epochs, and so BERT will as a result be fine-tuned for specific NLP tasks. BERT is widely used due to its ground-breaking performance with NLP tasks. Its bidirectional context representation and pre-training on large-scale corpora, result in BERT obtaining praiseworthy semantic and syntactic information. The architecture of BERT is also easily adaptable, therefore is versatile using fine-tuning.

RoBERTa (Robustly Optimized BERT) is a variant of BERT in the paper "RoBERTa: A Robustly Optimized BERT Pretraining Approach" (Liu et al., 2019). RoBERTa builds upon BERT's architecture yet targets to improve performance via improvements in the pre-training methods and the hyperparameters. Firstly, RoBERTa uses a larger training dataset compared to BERT, and it is trained for more steps, so developing a more extensive pre-training phase. RoBERTa also uses a larger batch size during training, which helps improve the stability of the optimisation process. A different learning rate schedule is used by RoBERTa and compared to BERT, RoBERTa uses a larger peak learning rate. In addition, RoBERTa engages a dynamic masking strategy during the MLM task, as opposed to BERT using a static masking strategy. Finally, RoBERTa does not use the NSP task during pre-training as it was demonstrated that this technique provided limited impact on BERT's performance. Consequently, these changes achieved better performance on many NLP tasks, outperforming BERT.

GPT

Generative Pre-Training (GPT) is mentioned in the paper "Improving Language Understanding by Generative Pre-Training" (Radford and Narasimhan, 2018) and introduces the first version of the GPT model, where GPT models are a family of large-scale generative language models based on the Transformer architecture. The GPT model builds upon the Transformer architecture by focusing on the unsupervised pre-training of a language model followed by task-specific-fine-tuning. As a result, GPT can have a general understanding of a large corpus of text, which then familiarises this understanding to NLP tasks with minimal task supervision.

The GPT series model we have chosen to use for our project is GPT-3.5-turbo (Ye et al., 2023) which is currently used within ChatGPT. In the Winograd Scheme Challenge, various models were tested such as code-davinci-002, text-davinci-001, text davinci-002, text-davinci-003, and gpt-3.5-turbo, and in zero-shot scenarios, gpt-3.5-turbo consistently achieves the best performance. A zero-shot scenario is a situation in which a model is evaluated on a task that it hasn't been specifically trained on or fine-tuned for.

Pseudo-Labelling

In the paper "Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks" (Lee, 2013), the author sug-

gests a semi-supervised learning technique called "Pseudo-Labelling". This method used unlabelled data to improve the performance of deep neural networks when there were only limited available labelled data. Pseudo-Labelling uses the predictions of the model on the unlabelled data to generate 'Pseudo-Labels', which in turn are used as additional training data. We adapted this approach to relabel our synthetic data during training.

The outline of the Pseudo-Labelling approach:

1. Train the model on the available labelled data for a few epochs.
2. Use the trained model to predict labels for the unlabelled data and assign Pseudo-Labels to this data at the beginning of every epoch, based on the model's predictions.
3. An augmented training dataset is formed by combining the labelled and Pseudo-Labelled data.
4. Continue training the model with this dataset while updating the model weights and refining its performance.

By using Pseudo-Labelling with synthetic data, we were able to augment our training set, especially when labelled data are scarce, potentially improving the model's performance which we will investigate in our experiments.

4 Methods

4.1 Data generation and preprocessing

We generated synthetic data in the style of the winogrande data though we did not generate the data in twin pairs but rather single individual sentences. The synthetic data was generated by making calls to the openai API. More specifically, the "gpt-3.5-turbo" model was used to generate responses to the following prompt:

> " I'm going to give you a task, do not answer until you've fully read and understood the prompt. All requirements of the prompt must be followed. Requirement 1: Your task is to create a sentence to show the ambiguity of pronouns. Requirement 2: in the generated sentence the one and only one ambigous pronoun must be replaced by the underscore character _ and there must only be two possible resolution options. Requirement 3: Make it very difficult to resolve the pronoun. Requirement 4: Do not reuse the example from the format provided at the end of the prompt. Requirement 5: Make sure your whole answer is in the exact format below without anything else: {"sentence": "Ian volunteered to eat Dennis's menudo after already having a bowl because _ despised eating intestine.", "option1": "Ian", "option2": "Dennis", "answer": "2"} Before you write your answer make sure you've followed the instructions and that all 5 requirements are met."

The model then generated 128 responses to the prompt, each with a maximum number of 100 tokens. The model had a temperature parameter(range 0 to 2) that could be used to restrict the randomness of the generated responses, with a temperature of 0 being more focused and deterministic. We left it at the default value of 1 to let the model have some creativity. This is all done in the synthetic_data_generation.py script. Note that in the script we have 50 call periods. Each period makes 6 calls to the "gpt-3.5-turbo" model so we should have 128x50x6=38400 responses but in synthetic_data_by_chat_gpt.txt we had 44048 simply because we had a few extra responses saved from previous hand-runs. To run the script, it is required to paste a valid open ai key in paste_your_open_ai_key_here.txt. Charges apply for using the api, more details at: https://openai.com/pricing.

Once the data was generated, we cleaned it to make sure it was in a valid format. Since we needed the data in jsonl format, that was the first check that was made. We made sure that the data had 4 fields: sentence, option1, option2 and answer. We then printed any lines that did not fit in such format and manually fixed them. As an example of the desired response format we return: {"sentence": "Ian volunteered to eat Dennis's menudo after already having a bowl because _ despised eating intestine.", "option1": "Ian", "option2": "Dennis", "answer": "2"}. Common mistakes included missing commas between the fields, some field names had first letter capitalized, extra full stops after parentheses, new lines started in the middle of the response and most interestingly sometimes the model would just fail and apologise for being unable to complete the task. To see the full list of

errors, run synthetic_data_checkAndClean.py with line number 5 uncommented. Once the data was in a valid format we carried out some more cleaning and prepocessing; we deleted any repeated reponses, deleted any sentences that had more than one underscore mask or none at all (since we needed exactly one underscore mask) and deleted any data points that had anything other than "1" or "2" in the answer field and made sure that sentences ended with full stops. Lastly we checked our synthetic data against the original organic data to make sure the model just hadn't copied data from Winogrande and no redundancy was detected. We then added unique ID's called qID in the jsonl file to keep track of the synthetic data. At the end we were left with 38121 datapoints and the fully cleaned version of the data can be found in fully_cleaned_data_chatgpt.jsonl

4.2 Tokenizer

Due to our choice of RoBERTa as our predictive model, we ended up using the RoBERTa tokenizer. Worthy of note is that this tokenizer uses Byte-Pair encoding, which gives it the ability to deal with new/rare words. Furthermore, the data diversity it encounters in its pretraining phase allows it to efficiently process text data while capturing subtle language nuances.

4.3 Pseudo labelling implementation

The synthetic data generated by our designed scheme comes with very poor labelling. In fact, we compared the labels that are generated along with the synthetic data against the predictive model trained on the whole organic dataset and the predictions of the fully trained model only matched up about 60% of the synthetic labels. Though we cannot claim that the synthetic labels were correct only 60% of the time (as that would require us to assume the fully trained model we're comparing it against is correct all the time), we also refer to the github repo(cha) where chatGPT was actually made to solve the winogrande problems and only achieved 62.75% which indicates that chatGPT might not be too well versed at assigning labels. To take care of the poor labelling we made use of the pseudo-labelling scheme as described in (Yang et al., 2020). We modified the original winogrande code(Win) such that the model is trained on both the synthetic(unlabelled) and organic(labelled) set and such that the loss function on each set is calculated separately. Our loss function of choice was cross entropy loss(very standard in the field) which compares the logits produced by our predictive model against the ground truth labels. The loss we trained on was the labelled loss (evaluated on organic labelled set) added to the unlabelled loss (evaluated on unlabelled synthetic set) where the unlabelled loss is multiplied by a weighting parameter α before the addition. Pseudo-labelling works by initially training the model only on the organic labelled data and once the model has been trained to adequate performance, we slowly introduce the unlabelled loss. This is essentially done by setting the α value to 0 in the initial phase epochs(which in our code is passed as the argument pseudo_label_wait_epochs) and once the wait period is over the α value slowly increases as described in the paper(Yang et al., 2020)

$$\alpha(t) = \begin{cases} 0 & \text{if } t < T_1 \\ \dfrac{t - T_1}{T_2 - T_1}\alpha_f & \text{if } T_1 \leq t < T_2 \\ \alpha_f & \text{if } T_2 \leq t \end{cases}$$

Where our α_f is 1 and T_1 corresponds to pseudo_label_wait_epochs and T_2 to pseudo_label_wait_epochs + 1. The α_f being 1 essentially means the most we weigh our unlabelled loss is a weight factor of 1. We pass an argument do_pseudo to our code that lets it to know whether to perform pseudo-labelling during runs. After pseudo_label_wait_epochs have passed, at the beginning of each epoch the model trained up to that point is used to make predictions on the synthetic data and and the predictions are then used as pseudo-labels for the synthetic data points. Essentially, the labels for the synthetic set evolve along with the predictive model in the hope that as the predictive model gets better, it learns to assign better and better pseudo-labels. The amount of the data that we use for training from each of the synthetic and organic datasets is determined by two arguments chat_gpt_data_proportion and winogrande_data_proportion respectively. We pass these arguments and these values range from 0 to 1, representing what proportion of the total data from each dataset we sample. Once we have sampled the required proportions, we concatenate them and sample from them randomly via a random sampling dataloader. We make sure to keep track which dataset, each datapoint came from and also

we sample batches of size per_gpu_train_batch_size which is again passed as an argument to the code and in our case was restricted to 12 across all experiments due to GPU memory constraints. The training data is divided into batches of size per_gpu_train_batch_size and however many such batches we have is the number of iterations in each epoch(note that every epoch is defined as the time taken to go through the whole training data once). All experiments were carried out with a randomness seed=42 for reproducibility. Inherited from the original code(Win), are the optimizer (AdamW), learning rate(1e-5 and with scheduling) and other similar hyperparameters. Throughout the experiments, the model was trained between 3 and 6 epochs at a time(though sometimes a model trained for a few epochs in one experiment is used as the starting point for a different model trained in another experiment). Even though this might seem like a low number of epochs, we note that each epoch usually ends up having a few thousand iterations in them. While we train our model, we regularly (argument logging_steps which in our case was set to every 100 iterations) evaluate the model on the validation data and report the prediction accuracy. To see this evaluation accuracy along with the training loss during training a TensorBoard session can be launched.

5 Experiments

For each of the runs, we look at the prediction accuracy on the validation set which has size 1267 examples and the loss value evaluated on the training set. In the colab demo notebook, these 2 metrics are plotted for every 100 iterations from beginning of training to end. In our analysis we return the mean and standard deviation of these across their values for the final 1000 iterations of each run(which gives us 10 values to take mean and standard deviation over). We had a total of 7 experiments/runs that can be divided into 2 categories:

5.1 Runs without pseudo-labelling

1. Using all the original labelled data and all of the synthetic data. Here we do not use pseudo-labelling to update the synthetic labels thus we treat the synthetic set as a labelled set and just use the labels that were generated along the synthetic data. We trained this from scratch for 3 epochs, with 6544 iterations per epoch.

2. Using all the synthetic data by itself. Again the labels that were generated along the synthetic data are used. We trained this from scratch for 3 epochs, with 3177 iterations per epoch.

3. Using all the original labelled data by itself. Here pseudo-labelling doesn't even apply. We trained this from scratch for 4 epochs, with 3367 iterations per epoch.

4. Using half of the original labelled data by itself. Here again pseudo-labelling doesn't even apply. We trained this from scratch for 3 epochs, with 1684 iterations per epoch.

5.2 Runs with pseudo-labelling

5. Using half of the original labelled data and half of the synthetic data. We train this for 4 epochs with 3272 iterations per epoch, with the pseudo-labelling having a wait period of 2 epochs thus we train on labelled set by itself for the first 2 epochs.

6. Using all the original labelled data and all of the synthetic data. We start from the pre-trained model checkpoint given by run 3. We then train for 3 epochs with 6544 iterations per epoch. No waiting period thus pseudo-labelling comes into play from the very beginning.

7. Using half of the original labelled data and all of the synthetic data. We train this from scratch for 5 epochs 4860 iterations per epoch and with a pseudo-labelling wait period of 2 epochs thus we train on labelled set by itself for the first 2 epochs.

6 Results and Discussion

We return the results in table form for comparison. The colab demo showcases how these values were calculated, and group17/csv_Results/ contains the csv files with the validation prediction accuracy and training losses from beginning to end for each experiment. These csv files were simply downloaded from the TensorBoard sessions.

As expected experiment2 has the worst performance(almost as bad as just randomly guessing with a 50% chance) as we trained on the poor labels that were generated along the data. As we go across the rows and increase our labelled data we

see an increase in performance. This is expected as it is documented that models benefit from more labelled data. As we go down the columns and add more unlabelled data in a pseudo-labelling setting, we tend to see either a decrease or no change in performance usually. However between experiment4 and experiment7 we actually see a notable increase in performance by introducing the unlabelled data through pseudo-labelling. We believe that pseudo-labelling might actually have some positive effects on the performance however this is only visible in situations where the unlabelled data vastly outnumbers the labelled data. In experiment4, we had twice the unlabelled data a labelled data whereas in the other pseudo-labelling experiments the two datasets were closer in proportion. Important to note that increases in the labelled data size have much better performance increases than similar magnitude changes in unlabelled data size.

	no labelled	**half labelled**	**whole labelled**
no synthetic	n/a	0.726 ±0.006 (experiment4)	0.768 ±0.001 (experiment3)
half synthetic, non-pseudo	n/a	n/a	n/a
half synthetic, pseudo	n/a	0.715 ±0.001 (experiment5)	n/a
whole synthetic, non-pseudo	0.528 ±0.003 (experiment2)	n/a	0.720 ±0.002 (experiment1)
whole synthetic, pseudo	n/a	0.737 ±0.003 (experiment7)	0.761 ±0.001 (experiment6)

Table 1: Mean ± standard deviation for validation set accuracy

7 Conclusion

During our project, for the most part we noticed no increase in performance by using unsupervised data with pseudo-labels. However, in a situation where the unlabelled data significantly outnumbered the labelled data, we had a positive effect on performance. It would be interesting to investigate a wider range of mixing proportions between labelled and unlabelled data, specifically proportions that have many more unlabelled data points than labelled. Further work could also include repeating the experiments again with the same sampling proportions but generate more synthetic data points overall(we had about 40k, could investigate something like 400k).

	no labelled	**half labelled**	**whole labelled**
no synthetic	n/a	0.086 ±0.016 (experiment4)	0.032 ±0.010 (experiment3)
half synthetic, non-pseudo	n/a	n/a	n/a
half synthetic, pseudo	n/a	0.199 ±0.028 (experiment5)	n/a
whole synthetic, non-pseudo	0.304 ±0.022 (experiment2)	n/a	0.210 ±0.016 (experiment1)
whole synthetic, pseudo	n/a	0.232 ±0.092 (experiment7)	0.087 ±0.033 (experiment6)

Table 2: Mean ± standard deviation for training set loss

References

The evaluation of winogrande debiased validation set on chatgpt. https://github.com/ugorsahin/Winogrande_ChatGPT. Accessed: 2023-04-12.

Winogrande: An adversarial winograd schema challenge at scale. https://github.com/allenai/winogrande. Accessed: 2023-04-12.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E. Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases.

Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang

Shen, Tianming Liu, and Xiang Li. 2023. Auggpt: Leveraging chatgpt for text data augmentation.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Dong-Hyun Lee. 2013. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Domagoj Pluščec and Jan Šnajder. 2023. Data augmentation for neural nlp.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Shuhuai Ren, Jinchao Zhang, Lei Li, Xu Sun, and Jie Zhou. 2021. Text autoaugment: Learning compositional augmentation policy for text classification.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics.

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models.