Improved Methods for Solving Diverse Winograd Schemas

Stanford CS224N Custom Project

Rohan Cherivirala
Department of Computer Science
Stanford University
rohanc12@stanford.edu

Max Vandervelden
Department of Symbolic Systems
Stanford University
mvdvldn@stanford.edu

Abstract

In this project we develop an improved RoBERTa model for solving Winograd schema problems, specifically those in the robust WINOGRANDE dataset developed by Sakaguchi et. al [1]. Winograd schemas, first conceptualized in the Winograd Schema Challenge (WSC) [2], test anaphora resolution [3], a challenging area of NLP for language models. We use the finetuned RoBERTa model (**RoBERTa-WinoGrande-ft**) developed in the aforementioned paper as a target performance comparison. The additions we make to **RoBERTa-WinoGrande-ft** are (1) new tested loss functions, (2) a **novel** randomized probabilistic problem modeling, and (3) relative word embeddings. These changes resulted in a model that achieved a 75.6% accuracy on the WINOGRANDE dataset and lies 3.5% below that which was reported in Sakaguchi et. al [1]. Although no improvement in accuracy was found, these changes could prove fruitful in cracking the Winograd challenge with future work.

1 Key Information to include

- Mentor: Christopher Cross
- External Collaborators: None
- Sharing project: No

2 Introduction

In 2011, Levesque, Davis, and Morgenstern [2] developed the Winograd Schema Challenge (WSC), a commonsense reasoning test involving pronoun anaphora resolutions. In linguistics, anaphoras are expressions that are defined by the words, phrases, and general context surrounding them; without these contextualizing primers, their meaning becomes ambiguous [3]. For pronouns, specifically, situations like these appear frequently because pronouns are inherently referential to the noun that they are describing. For example, in the sentence:

"John moved the couch from the garage to the backyard to create space. **It** is too small."

Here, an example choice for the pronoun **It** may be "garage" and "backyard". Both of these nouns syntactically satisfy the blank in this sentence; however, the contextual sentence, "John moved the couch from the garage to the backyard to create space," and the determiner sentence, "**It** is too small," must be understood in tandem when determining the correct input. Based on this, we know "garage" is the more likely here because the couch is being moved out of the said garage because of its size. Because Winograd schemas test linguistic understanding of the meaning of the sentence itself both also have commonsense reasoning skills about the nouns involved, they are often used as a benchmark for the logic capabilities of new language models and are considered an improvement to the Turing

Test [1]. Models tested on Winograd schemas are generally fed some contextualizing sentence or clause and a series of potential noun options for a blank or corresponding pronoun. Frequently, these pronouns are presented as blanks ("_") to the models instead of the actual pronouns themselves [1]. This replacement significantly increases difficulty due to the lack of gender and number information conveyed by the existence of these pronouns. Accuracy on these tests is measured by how frequently the models choose the right noun for the corresponding pronoun to match the meaning of the context clause.

On average, humans perform very well on Winograd schemas, achieving around 94% accuracy [4]. However, most language models struggle, usually doing no better than random chance. This is generally caused by models lacking the contextual understanding and complex reasoning skills needed to solve the schemas [2]. In the small challenge dataset developed by Levesque, numerous neural language models have achieved around 90% accuracy [2]. However, because of WSC's small size (273 entries), it is not clear how robust these models actually are and whether they would perform well against newly generated and diverse Winograd schemas [1]. In response, Sakaguchi et. al [1] developed a more robust dataset of Winograd schemas known as WINOGRANDE.

We make three conceptual changes to the RoBERTa finetuned model developed by Sakaguchi et. al. Our first addition is testing new loss variants of Binary Cross-Entropy (BCE), namely BCE Dice Loss and BCE Loss with Logits. We included these new metrics in an attempt to efficiently increase accuracy within our model. Our second addition is the most significant. We change the problem's modeling, introducing more variability in how the input context and determiner sequences are fed into the model's pretrained RoBERTA base to provide more generalizability. Our third addition is changing the model's embeddings from absolute to relative key queries. Because pronouns often describe the words or nouns most proximate to them, we seek to better represent this with relative embeddings.

3 Related Work

There are a number of different methods which have been developed to test commonsense reasoning skills for NLP models. These testing methods are known as Natural Language Inference (NLI) tasks. These types of problems involve some context, and, based on this context, a hypothesis; the model must determine the validity of this hypothesis [5]. Winograd schemas are a great example of NLI problems because they involve a contextualizing sentence and a "hypothesis" wherein the model must compare the validity of two possible noun inputs [2]. Some other developed NLI metrics problems include SNLI [6], MULTINLI [7], QNLI [8], and RTE [9]. Of these, the latter three are included in the multi-task GLUE benchmark which is frequently used to measure the general commonsense reasoning skills of natural language models [10].

Our work primarily stems from that of Sakaguchi et. al [1]. WINOGRANDE is a dataset of more than 44,000 Winograd schemas to improve this robustness in comparison to WSC, focusing explicitly on encouraging creativity within and removing algorithmic bias from these examples. To foster creativity within the dataset, authors randomly selected anchor words from articles from which workers would develop related example schemas. This process is known as "creativity from constraints" and encourages new sentence topics and structures [11]. To reduce spurious dataset bias, the authors developed a new algorithm known as AFLITE, an adversarial filtering algorithm. Through this algorithm, the authors systematically removed dataset-specific bias created between different words in the context and determiner clauses which might prime the model to assign certain nouns in place of pronouns based on sentiment. The authors note how the WINOGRANDE dataset contains significantly more challenging and complex questions than the WSC and other existing variants and therefore accuracy on it suggests high levels of linguistic understanding and reasoning.

After testing the dataset on various current model types, the authors then developed **RoBERTa-WinoGrande-ft** from a pretrained RoBERTa base which was finetuned on the WINOGRANDE training and dev sets. With this model, the authors achieved higher than 85% accuracy on six similar Winograd schema experimental tasks (including the WSC), much higher than the 65% to 80% accuracy demonstrated by the other models. When compared to models trained on the Definite Pronoun Resolution Dataset (DPR), another NLI dataset [5], the model performs consistently better on all WSC variant tasks [1].

In terms of modifying these types of pretrained transformers, small changes, like changing embed scaling, relative and absolute embeddings, and early stopping, have shown to have a significant impact on model performance. Small changes in how the problem's modeling and input can drastically improve performance on large datasets like PCFG and COGS, sometimes by more than 30% [12]. Our approaches include similar ideas, making small modifications to the model's architecture in an attempt to discern what may make significant impacts.

4 Approach

4.1 Architecture Changes

These changes are fundamentally intended to improve the framework used on top of the RoBERTa pretrained base developed by Sakaguchi et. al, available at **github.com/allenai/winogrande** [1]. As such, we used their basic framework but made original changes for all of our approaches. We refer to the final model with these changes in our **Results** and **Conclusion** sections as **RoBERTa-Winogrande-Modified**.

Our first approach deals with testing new loss functions. The default loss function used in NLP models for binary classification is Cross-Entropy Loss. For binary comparisons like what we are doing between two noun options, this loss is known as Binary Cross-Entropy or BCE [13]. We test two new loss functions which are variations on BCE Loss – BCE Dice Loss and BCE Logits Loss – to measure how these functions influence final model accuracy. BCE Dice Loss is a combination of BCE and the Dice coefficient for loss, measured as

$$Dice\ Loss = 1 - Dice\ Coefficient = 1 - \frac{y \cap y_{pred}}{y \cup y_{pred}} \tag{1}$$

where y are the actual answers to the WINOGRANDE dataset and y_{pred} are our model's predictions for the answers [13]. We theorized that providing the model with a new metric of loss could serve as a better representation of accuracy. We tested a second new loss function, BCE with logits, to provide more numerical stability within our model. This stability is created by combining the output of BCELoss and a sigmoid layer within one class. BCE Loss is somewhat numerically unstable in calculation because of its potential to produce very large or small values, leading to `Inf` or `NaN` calculations [14].

Our second approach is a **novel** approach to changing the problem encoding. Our change specifically manipulates the inputs into the RoBERTa pretrained base of the model. The multiple-choice (binary) version of RoBERTa is presented with two different choices. Each choice is presented as two segments of the input `sentence` (see **Evaluation method** for more detail). These two segments are formed by splitting `sentence` by an index, which Sakaguchi et. al chose to be the space before the "_" representing the possible noun inputs [1]. Therefore, the first segment was guaranteed to contain all of the primer context phrase or sentence. We introduce some randomization into this selection by subtracting the initial split index by $\frac{\texttt{len(sentence)}}{3}$ with a 15% probability. For the other 85% of the time, the splitting index remains the same as the blank ("_") index (see **Figure 1**). Our motivation for this change was to ensure that the model pays close attention to both the primer and determiner when making its predictions and by varying the location of the separation on specific examples. Moreover, this procedure helps convey the importance of both parts of the sentence in commonsense reasoning.

Our third approach was to use relative key query embeddings to represent word tokens instead of absolute embeddings. Potentially, this allows the model to include closeness to other words in the input sentence as a better representation of coreference probability. Additionaly, relative embeddings help capture the intricate complexity of commonsense reasoning problems much better than relative representations.

4.2 Baselines

We compared **RoBERTa-Winogrande-Modified** to a few main evaluation benchmarks and targets. The first was random chance. Given the two possible answer choices for all problems in the WINOGRANDE dataset, random chance performance would be equivalent to roughly 50%. Our second set of benchmarks come from Sakaguchi et. al, who measured the performance of various

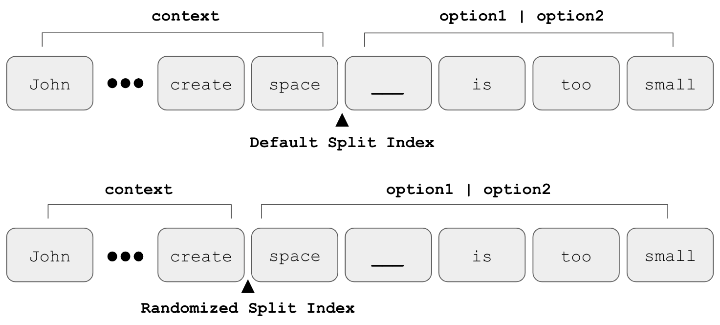

Figure 1: Comparison between default and potential randomized problem embedding

model types including Wino Knowledge Hunting (WKH), Ensemble Neural LMs, and BERT on the dataset. Our target performance was the RoBERTa model itself, which achieved a 79.3% dev and 79.1% test set accuracy. The performance of these models in comparison to our final results is given in Table 1.

5 Experiments

5.1 Data

The dataset we are using and evaluating on is the WinoGrande dataset, which is already split into training, dev, and test sets which we will also use [1]. Structurally, the authors also their data in comparison to the WSC dataset [2], only providing the expression itself and a binary option between two possible corresponding nouns [1]. The WinoGrande dataset contains 44k problems; each with a `sentence` string entry; two option choice strings, `option1` and `option2`; and an `answer` binary string [1]. `sentence` contains some sort of primer context phrase or clause and a determiner phrase or clause that involves a blank "_" representing a pronoun later within it. Here, the language model must either choose from the nouns `option1` or `option2` which best fit in this blank given the context of `sentence` [1].

5.2 Evaluation method

Since the problem we are looking at is solving a binary choice task, the evaluation metric we used was determining what percentage of the test set the model predicted correctly. More specifically, we evaluated a model based on the percentage of questions in the test set where it was able to choose the correct option out of the two possible answers.

5.3 Experimental details

Based on the random grid search hyperparameter optimization, we trained our model using a learning rate of $1e-5$. Likewise, we trained our eventual model using 100 epochs. Since each epoch took around 5.3 minutes, training took around 9 hours. We trained our models on the `train_large` WinoGrande dataset. Outisde of the changes made in our approaches, we followed the general model configurations established in Sakaguchi et. al, specifically that of finetuning RoBERTa on the `WinoGrade` dataset [1].

5.4 Results

Among our loss variants, we saw very little difference in performance and loss degradation. The regular BCE Loss achieved a loss of roughly 0.05 and a final accuracy of 73.8%. In fact, the multi-factor BCE Dice Loss showed a relatively worse accuracy and loss over time in comparison to the regular BCE Loss. Ultimately, Dice Loss experienced a 0.03 greater loss and a 0.016 lower accuracy (73.8% compared to 72.2%). BCE with Logits performed almost identically to regular BCE in terms of loss and was slightly less accurate (73.8% compared to 72.4%).

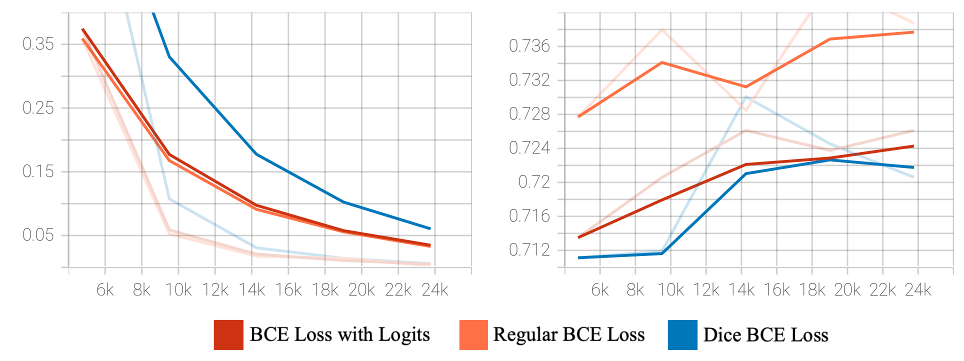

Figure 2: Loss (left) and Accuracy (right) over Training Iterations for Loss Variants ($Smoothing = 0.6$)

When modifying problem modeling, we saw a significant increase in the accuracy of the model. For the normal BCE Loss, there was a 0.2% improvement when including the modified problem approach over the same number of 40 epochs. However, we trained this modified approach for longer than our initial BCE Loss run. When left to train longer, this modified-approach model achieved a 0.8% improvement between the two final versions. In comparison to the BCE Logits loss variant, adding the new problem randomization significantly improved accuracy over 40 epochs by 1.1%.

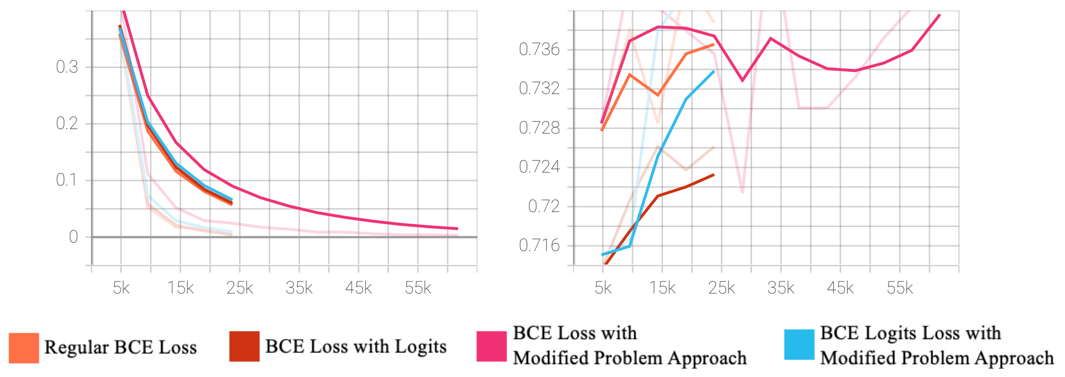

Figure 3: Loss (left) and Accuracy (right) over Training Iterations for Modified Problem Modeling ($Smoothing = 0.6$)

When modifying the embedding of the model from absolute to relative, we saw some interesting changes. We trained this relative embedding model for 70 epochs. Both the modified problem approach and relative embedding accuracies reach a large value of accuracy and decrease dramatically before rebounding. For the relative embedding model, it achieved an accuracy of above 75% at around 15k training iterations (25 epochs), significantly higher than the 73.8% accuracy achieved by the standard BCE Loss. After this point, the accuracy decreases dramatically to a local minimum accuracy of 72.7% at around 24k iterations (40 epohcs). Afterward, it increases quickly to a final accuracy of 74.2%, 0.4% larger than the default BCE Loss. Furthermore, based on the similar performance of the modified problem model, it seems as though the accuracy might still continue to increase if left to train past 70 epochs. We discuss this trend further in **Analysis**.

Ultimately, our final results for the model, including the default BCE Loss, modified problem modeling, and relative embeddings, were promising. After only training on the large WINOGRANDE dataset for 100 epochs, we achieved an accuracy of 75.6%, significantly larger than random chance and all of our model baselines. The final 79.1% accuracy calculated by Sakaguchi et. al was achieved on the largest training dataset (xl) over a much larger number of training iterations [1].

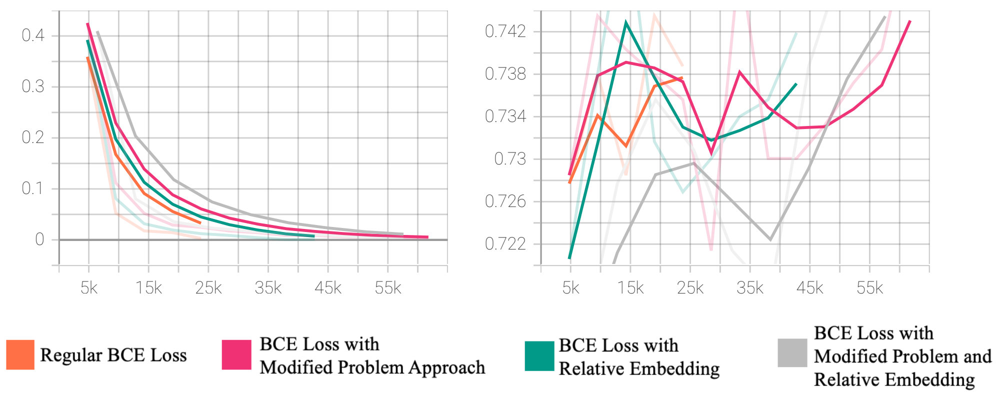

Figure 4: Loss (left) and Accuracy (right) over Training Iterations for Embedding Changes ($Smoothing = 0.6$)

Model	Test Accuracy (%)
Random Chance	50
Wino Knowledge Hunting	49.6
Ensemble Neural LMs	50.9
BERT	64.1
RoBERTa (**RoBERTa-WinoGrande-ft**)	79.1
RoBERTa-WinoGrande-Modified	75.6

Table 1: Final accuracy of benchmarks and our model, **RoBERTa-Winogrande-Modified**, on WINOGRANDE test set.

6 Analysis

As seen in our results section, each loss function resulted in a stable descending loss curve. This is mainly a result of the loss functions we chose being derivatives of the original Binary Cross-Entropy Loss. Although many of these changes initially seemed promising, we found they did **not make a substantial change** in the results and resulted in fairly similar model behavior. To highlight this, we can look at the loss curves in Figure 1. Moreover, we see that each loss curve follows an incredibly similar trajectory, with the BCE loss with logits being almost identical to the regular BCE loss.

Conversely, we found that the problem modeling change had a **significant impact** on the performance of the model. We believe that this is mainly due to the increased level of generality that is achieved by adding an element of randomization to the position of the separation token. This randomization helps convey the generality needed in commonsense reasoning and is employed in effective transformers like BERT.

By changing the model to use relative embeddings, we saw a **slight improvement** in model performance. This is likely a direct result of how the complex relationships between different parts of a Winograd sentence are better represented through a relative key-value query.

When examining the model's performance, we found a general pattern between the examples that were missed, specifically that they often required very specific knowledge about a specific attribute. Take the following sentence:

"Ben had to drain both water and oil down the sink, but the _ was too viscous."

Clearly, the correct answer in this scenario would be **oil**, since oil is generally more viscous than water. Thus, we believe that the model may have missed this question due to being unable to apply the specifics of liquid viscosity to commonsense reasoning. We believe that this is likely due to the model losing generality as it is finetuned on the training task.

This idea is also echoed in the accuracy we see on the validation set while training. Namely, there seems to be a spike in model accuracy at around 15-20k iterations followed by a sharp decline in model performance at around 25-30k iterations. After this, the model's performance slowly recovered. We found this result to be highly interesting and we believe this dip in accuracy is caused by an initial loss of generality that occurs from finetuning on the WinoGrande. On the same note, the general path of recovery that follows seems to stem from the model better understanding of the problem and hand. This view echoes the trends in the results that we saw, specifically with the fully trained model having poor performance on commonsense reasoning questions requiring specific outside knowledge.

7 Conclusion

7.1 Summary and Takeaways

We found that randomizing the model's input structure and introducing relative embeddings into the model significantly improved performance. We found little change from introducing new loss variants like BCE Dice Loss and BCE Loss with Logits. Conversely, we saw substantial and meaningful improvements when applying our novel modified problem approach and relative embedding schema. Although our final model's accuracy was 3.5% lower than that of Sakaguchi et. al, the changes made substantially improved model performance and outline a path for getting closer to solving the Winograd challenge [1].

Through this experience, we learned that small architectural changes have large compounding impacts on model performance. Additionally, we found that many changes that sound theoretically beneficial often have unexpected practical implications. Moreover, we found that the best way to make changes to a LLM was through repetitive testing and iteration.

7.2 Limitations

The main limitation we faced in our training process was a lack of computing power. Coupled with time constraints, we were unable to adequately tune our hyperparameters and were only able to carry out a relatively simple random grid search. This resulted in our model not being able to achieve the level of test accuracy we hoped for, especially in comparison to Sakaguchi et. al.

Another limitation that we faced is dealing with the unruly codebase that we used to build our model. Since the codebase was created 3 years ago, we found ourselves having to remove significant archaic code and fix many model compatibility issues. Moreover, we were unable to reproduce any of the results published in Sakaguchi et. al and instead started off with a model with about a 72% accuracy, 7.1% lower than the accuracy reported in their paper [1].

7.3 Future Work

One source for future work is how we choose the hyperparameters for our model. Grid search, our method for selecting hyperparameters, is a very simple and brute-force method. This method uniformly and randomly selects hyperparameters and therefore must test many ineffective options [15]. In comparison, Bayesian optimization is an informed search method, designed to finetune hyperparameters based on the success of previous trials. This process is based on Bayes theorem [16] and is applied to hyperparameter optimization via

$$p(Score \mid Hyperparameters) = \frac{p(Hyperparamaters \mid Score) * p(Score)}{p(Hyperparamaters)} \tag{2}$$

where $Score$ represents performance on some objective function. Here, the objective function we are trying to maximize is a representation for accuracy on the WINOGRANDE dataset. While Bayesian optimization reduces the number of trials needed to find adequate hyperparameters, it also requires more calculation for each individual iteration [16].

Another effective change to the model could be modifying the pretrained base. One pretrained architecture we came across in our research was DeBERTa, which performs better than RoBERTa on many Natural Language Inference (NLI) tasks similar to Winograd Schemas [17]. The base's disentangled attention mechanism and enhanced mask decoding could lead to significant improvement in our results with the model accuracy [17].

References

[1] Sakaguchi, Le Bras, Bhagavatule, and Choi. Winogrande: An adversarial winograd schema challenge at scale. 2019.

[2] Morgenstern, Davis, and Ortiz. Planning, executing, and evaluating the winograd schema challenge. 2016.

[3] Reuland, Everaert, and Volkova. Anaphora. 2011.

[4] Bender. Establishing a human baseline for the winograd schema challenge. 2015.

[5] PapersWithCode. Natural language inference, 2023.

[6] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference, 2015.

[7] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference, 2018.

[8] Dorottya Demszky, Kelvin Guu, and Percy Liang. Transforming question answering datasets into natural language inference datasets, 2018.

[9] Adam Poliak. A survey on recognizing textual entailment as an nlp evaluation, 2020.

[10] Wang, Singh, Hill, Levy, and Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. 2019.

[11] Stokes. Creativity from constraints: The psychology of breakthrough. 2005.

[12] Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. The devil is in the detail: Simple tricks improve systematic generalization of transformers, 2022.

[13] Opidi and Jha. Pytorch loss functions: The ultimate guide. 2023.

[14] PyTorch. Bcewithlogitsloss documentation, 2023.

[15] Koehrsen Will. A conceptual explanation of bayesian hyperparameter optimization for machine learning. 2018.

[16] Shekar, Bansode, and Salim. A comparative study of hyper-parameter optimization tools. 2022.

[17] He, Liu, Gao, and Chen. Deberta: Decoding-enhanced bert with disentangled attention. 2019.