Parts of MinBERT

Stanford CS224N Default Project

Monica Hicks
Department of Computer Science
Stanford University
mdhicks@stanford.edu

Megan Liu
Department of Computer Science
Stanford University
meganliu@stanford.edu

Abstract

Parts of speech is valuable information to understand what a sentence means, and distinguishing such could help a model disambiguate homographs. Our project aimed to extend the base functionality of the minBERT language model to improve performance on 3 evaluation tasks (sentiment classification, paraphrase detection, and sentence similarity) by implementing strategies that will make the parts of MinBERT more robust and holistic. These are (1) POS tag experiments inspired by prior work (Benamar et al. (2021)), (2) gradient surgery implementation (Yu et al. (2020)), and (3) modifications to the task-specific heads, including Siamese network implementation (Reimers and Gurevych (2019)).

Our experiments implementing POS tagging at different stages of training revealed limited improvements to paraphrase detection and a negative performance impact to sentiment classification and sentence similarity tasks. We found that gradient surgery made the most significant improvement to overall score. Finally, the addition of a head-specific sigmoid function to similarity prediction also improved performance, while a simple cosine similarity prediction and Siamese network reduced overall performance.

We discovered our initial more ambitious/ambiguous experiments were extremely time-expensive, as our initial implementation took half a day to run on just the last linear layer, and the later steps taken towards gradient surgery which reduced runtime were most effective in improving our model's performance from the our baseline.

1 Introduction

Building systems which understand language and perform tasks is central to our increasingly technological society. As we seek to capture human ability to communicate with language in the field of Natural Language Processing, the original Bidirectional Encoder Representations from Transformers (BERT) language model (Devlin et al. (2019)) was a breakthrough advancement utilizing context awareness and pre-training to effectively handle various NLP tasks. This foundational model is limited when simultaneously balancing on multiple unique tasks.

Our work explores various strategies to improve this performance, building upon a baseline minBERT model in order to improve its performance on three downstream tasks simultaneously: sentiment analysis, paraphrase detection, and semantic textual similarity. We implemented multitask fine-tuning heads for each of these tasks, and tested effects of extensions to functionality on performance through (1) experiments with Part of Speech (POS) tagging, (2) gradient surgery (Yu et al. (2020)), and (3) modifications to the task-specific heads, including Siamese network implementation (Reimers and Gurevych (2019)). We are particularly interested in seeing if POS tagging would improve performance because from a human perspective the POS each word is gives important context which can help disambiguate homonyms/homographs and give broader context to a word's meaning in a sentence. We were also interested in how it would affect each task. Next, we explored gradient surgery to directly address the limitations multiple task heads introduce in conflicting gradient directions,

which interferes with performance on each of the conflicting tasks as well as overall performance. We found that our architectural modifications in combining the task iterations was perhaps the most crucial step towards improvement as it vastly reduced the runtime and made full model runs which updates not only task specific head parameters but also BERT model parameters, including gradients. Finally, we tried a number of head-specific modifications including Siamese Network as described in Reimers and Gurevych (2019) and direct application of cosine similarity and sigmoid as activation functions. Throughout our implementation, we naturally tried a variety of combinations of these extensions, which are detailed in our results.

We repeatedly encountered challenges with runtime as well as memory usage while implementing these experiments, especially with POS and Siamese Networks, which may have contributed to their decrease in performance. However, we reason that incorporating POS-tags at the fine-tuning stage (as opposed to pre-training) and task interference were the major factors which prevented overall task performance from increasing for POS, Siamese Networks, and cosine similarity activation functions. While sigmoid activation improved semantic textual similarity detection, we ultimately found gradient surgery to be our most effective extension for improving overall task performance.

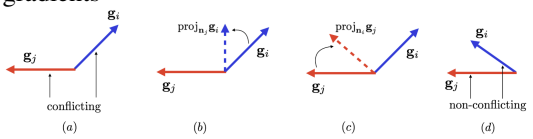
2 Related Work

While the BERT model's original approach is well defined by Devlin et al. (2019), we explored 3 other papers as inspiration for our extensions.

Parts of Speech Tagging Benamar et al.'s *Easy-to-use combination of POS and BERT model for domain-specific and misspelled terms* was our primary influence for methods of incorporating POS tags to our BERT model(2021). Benamar et al. chose to concatenate the POS tags directly to their corresponding term prior to passing to BERT, resulting in an interleaved representation in the sentence. One drawback of this approach is that interleaving tokens and POS tag representations can made it difficult for the model to understand the semantics of the words. Thus, we wanted to instead explore concatenating the entire sentence embedding with the POS tag embedding representation and observing the results. In our extension, the model still has the contiguous representation of the sentence, which should improve its ability to extract important semantic representations by allowing the attention heads to focus on the dependencies between components of the sentence.

Gradient Surgery When training a model on such distinctly varied tasks, it is extremely likely gradients are in different directions, and when two gradients are conflicting, the optimizer may end up making no progress at all. Yu's *Gradient Surgery for Multi-Task Learning* presents an approach to minimize this destructive interference from combining losses from tasks with conflicting gradients, illustrated in Figure 1 and calculated as $g_i = g_i - \frac{g_i \cdot g_j}{||g_j||^2} \cdot g_j$. By projecting the gradient of the i-th task g_i onto the normal plane of a conflicting task's gradient, g_j, interfering components are no longer being applied to the network. (Yu et al. (2020))

Figure 1: A diagram from Yu et al. illustrating how PCGrad handles tasks i and j with conflicting gradients



Siamese Networks The journal *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks* highlights the unsuitability of passing two sentences to a transformer to predict a target value, and outlines the implementation details for a Siamese BERT-Network (Reimers and Gurevych (2019)). A major drawback of the Siamese BERT-Network implementation is the memory overhead in maintaining two separate BERT models. We wanted to see whether we could improve performance in our downstream tasks using the Siamese BERT-Network concatenation strategy using a single BERT node.

3 Approach

The baseline model is a multi-task BERT model with shared BERT layers across all 3 tasks, applying a simple linear layer with dropout for each task. Tasks with sentence pairs are concatenated before being passed into the linear layer. During each epoch of training, the each task's dataset is considered

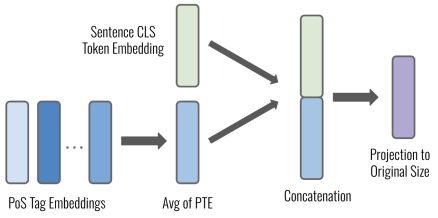
in its own separate, consecutive stage. This method utilizes the full extent of the provided data for each task (see section 5.1), but took approximately 7 hours to complete one run on last-linear-layer mode only, and would run out of memory by the 4th epoch on full-model-mode (at approximately 30 hours). However, when we later discovered we could evaluate on the 3rd epoch save of a full-model run, we discovered a significant boost to performance, which directed our later training architecture reorganization (see 4.3.1) and explorations. The losses are are calculated using cross entropy loss for sentiment classification, binary cross entropy loss for paraphrase detection, and mean squared error loss for semantic textual similarity.

3.1 Parts of Speech Tag Concatenation

Our first method of incorporating POS tags into our model was adding POS tag embeddings to our sentences in the `embed()` portion of our `bert.py` model. Due to limited success with this approach when evaluating on just sentiment classification and TA advice, we moved this code from the pre-training step to `multitask_classifier.py` to help improve our downstream tasks.

To implement this method, we took the following steps. Beginning with the `input_ids` passed to our function, we utilized the `convert_ids_to_tokens` function from `tokenizer.py` to retrieve the original components of the sentence. After retrieving these tokens, we used the imported library `nltk` to utilize the parts-of-speech-tagger to create a list of the corresponding parts of speech for each token.

Figure 2: A diagram illustrating our method of POS tag concatenation



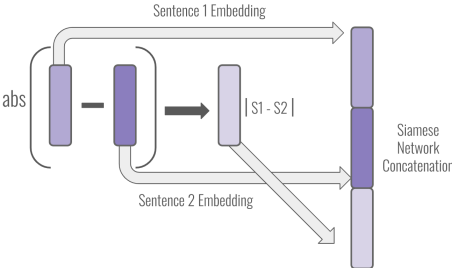
To concatenate this part-of-speech representation of the sentence with the representation of the original sentence, we took the average of the parts-of-speech embedding vectors and concatenated this to the `[CLS]` embedding returned from our `forward` function. The logic behind this is that, since BERT if Bi-Directional in nature, the `[CLS]` token contains the overall embedding for the sentence. Similarly, the averaged parts-of-speech tag presents a pseudo-representation of the overall components of the sentence. After concatenating these two embeddings together, we project them back to the original embedding size by passing them through the linear layer, and then the part-of-speech preprocessing portion is complete.

Our expectation was that this would have a pronounced effect in the downstream tasks of paraphrase detection and contextual similarity. We hypothesized that having the parts of speech information would be much more helpful in predicting the relatedness of two sentences versus the impact it would led to the task of sentiment analysis.

3.2 Siamese BERT-Network Implementation

We decided to implement a Siamese BERT-Network implementation of the downstream tasks with two sentences: paraphrase detection and semantic textual similarity. We implemented this network ontop of the POS tag embedding we outline in the prior sub-section.

Figure 3: A diagram of our Siamese BERT-Network implementation

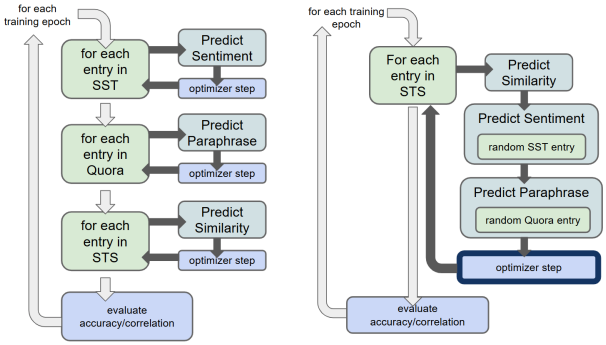


Beginning with the projected embeddings from the previous step, we concatenate sentence one and sentence two and then concatenate the absolute value of the difference between the two sentences to the end of the previous concatenation. We then pass this composite concatenation to the linear model which produces an output of the original embedding size.

3.3 Gradient Surgery Implementation

Our specific implementation for gradient surgery leverages the PCGrad implementation created by Yu et al. (2020) at the bolded "optimizer step" in our training method in Figure 4, projecting the 3 tasks' gradients onto each others' respective normal planes to reduce stalling from conflicting gradients.

Figure 4: (Left) The sequential training in the baseline model. (Right) The combined training architecture



4.3.1 Combining Training Iterations

Our initial, naive, consecutive approach to combining fine-tuning heads during training was not compatible with implementing gradient surgery. Thus we needed to adjust our training architecture in order to simultaneously access the 3 task gradients at once during the optimization step. Rather than optimizing for each task's entire dataset in order, we limited the batch size to be the same as the smallest dataset (STS) and only calculated losses for one data point from each dataset for each step in the batch. This allowed us to make optimization steps that account for all three tasks.

3.4 Semantic Textual Similarity Head Modifications

Finally, as we discovered that the STS task had a consistently lower correlation than the others' accuracies, we implemented a few simple experiments in modifying the architecture of the head. Following discussion during office hours, we tested = two strategies. First, we applied a Cosine Similarity to the STS head, combining the linear layers using the cosine similarity function rather than the concatenation, and deriving the logits as

$$\texttt{logit} = (\texttt{cosine\_similarity}(s1, s2) + 1) \cdot \frac{5}{2}$$

where $s1$ is the output from applying the linear layer to the first sentence's embedding, and $s2$ is the the output from applying the linear layer to the second sentence's embedding.

Next, we applied a sigmoid activation to our similarity prediction function as

$$\texttt{logit} = \sigma(l) * 5$$

where l is the output from the linear layer.

4 Experiments

4.1 Data

For **Sentiment Analysis**, we used the **Stanford Sentiment Treebank (SST)** dataset, and the CFIMDB dataset during early evaluation. The SST dataset contains 11,855 elements, reach representing a single sentence extracted from movie reviews. The 215,154 unique phrases in this dataset were annotated by 3 human judges. The sentiment outputs ranging from 0 to 4 (positive). The dataset was segmented into the following splits: train (8,544 examples), dev (1,101 examples), test (2,210 examples). The CFIMDB dataset contains 2,434 movie reviews. Each of these reviews contains a binary label of positive or negative. The dataset is divided into the following splits: train (1,701 examples), dev (245 examples), and test (488 examples).

For **Paraphrase Detection**, we used the **Quora** dataset. This dataset contains 404,298 pairs of questions that are labeled 1 (is a paraphrase) or 0 (is not a paraphrase). The data is segmented into the following splits: train (283,010 pairs), dev (40,429 pairs), and test (80,859 pairs).

For **Semantic Textual Similarity**, we used the **SemEval STS Benchmark** dataset which contains 8,628 sentence pairs. The pairs have similarity scores ranging from 0 (unrelated) to 5 (equivalent meaning). The dataset is divided into the following splits: train (6,040 examples), dev (863 examples), test (1,725 examples).

4.2 Evaluation method

To evaluate overall performance of our model, we calculate the performance as

$$\frac{\texttt{Sentiment Analysis} + \texttt{Paraphrase Detection} + \frac{\texttt{(Semantic Textual Similarity + 1)}}{2}}{3}$$

Where `Sentiment Analysis` and `Paraphrase Detection` is the accuracy between the predicted labels and true labels, and `Semantic Textual Similarity` is the pearson correlation between the predicted and actual labels. These were also how we evaluated the individual models.

4.3 Experimental details

As a two-person team, we used two Google Cloud Platform (GCP) instances which demonstrated a surprisingly high amount of variability on the same code. For the same baseline model run using the gpu on last linear layer, batch size 16 one instance achieved 0.578 while the other only achieved a baseline of 0.496 using an identical version of the code. Regardless, we pursued development using a shared git codebase. When running the baseline model, we frequently encountered large runtimes (>6 hours) and out-of-memory errors, so we ran on last-linear-layer, with batch size 64, for a runtime of approximately hours. Later runs with the combined iteration training architecture had training times between 1-2 hours on full model mode and batch size 64 (or 8, to reduce memory usage for gradient surgery). For all runs, we used the default learning and dropout rates, default dimensions, and only reduced batch sizes to avoid memory errors.

4.4 Results

Dev Leaderboard Scores

Model	Run Mode[1]	Sentiment	Paraphrase	Similarity	Overall [2]
Baseline	lll	0.259	0.697	0.396	**0.574**
(separate training batches)	fm - epoch 3	0.447	0.654	0.385	**0.598**
POS embeddings	lll	0.381	0.680	0.362	0.580
Siamese network	lll	0.384	0.664	0.340	0.559
Combined Iteration	fm	0.521	0.675	0.362	**0.626**
Gradient Surgery	fm	0.535	0.722	0.347	**0.643**
Add Cosine Similarity to `predict_similarity`	fm	0.212	0.675	0.108	0.578
Sigmoid in STS	fm	0.529	0.695	0.343	0.632
Grad. Surgery + Sigmoid	fm	0.510	0.714	0.374	0.637

These results were surprising to us in many ways, particularly the scores that did not improve. While we expected sentiment classification to improve from the last-linear-layer baseline when using POS tagging, we did not expect it to impact the performance of the others so significantly that it was not viable when combined with other strategies. Based on the Reimers and Gurevych (2019) paper, we expected the siamese network implementation to show improvements in paraphrase detection and similarity. We believe the overall concatenation was ultimately treated as unhelpful noise. We discuss this further in section 6 (analysis).

[1] Last Linear Layer = lll, Full Model = fm

[2] Bolded entries were used as benchmark implementations

We also initially expected cosine similarity to improve functionality at least slightly, and were surprised that it decreased. However, the reduction in the STS correlation revealed to us that there was likely an implementation error. Upon closer inspection, we realized we had modified the linear layer to only input one sentence as opposed to two, which would have significantly reduced the efficacy of its training on the similarity of two sentences.

Similarly, we were surprised that the addition of the sigmoid function in `predict_similarity` led to an overall increase in performance but a slight decrease in STS performance, and an overall decrease when combined with the gradient surgery model. While in early epochs STS correlation appeared better than before, we deduce the final correlations and overall accuracies are lower for a similar reason to the failure of cosine similarit; simply adding the sigmoid function was initially helpful in producing an even spectrum of decimal values between 0 and 5, but by applying it after the linear layer, the transformation wasn't accounted for in the linear layer's weights/biases, and led to a slightly more unpredictable behavior in later epochs.

A final point of interest is that our highest STS correlation scores were in our baseline model. Examining our training architecture (left side of Figure 4) this naturally follows, as STS was the final task trained on, so the optimizer likely took many steps in directions which conflicted with the prior tasks' training, reducing their performance but improving the similarity task's correlation.

We ultimately submitted to the test leaderboard using the combined iteration model trained on full model mode and batch size 8 with only gradient surgery applied.

Test Leaderboard Scores

Sentiment	Paraphrase	Similarity	Overall
0.524	0.723	0.345	0.640

5 Analysis

5.1 POS Tag Concatenation

We did not expect to see a significant improvement in our model on the task of sentiment analysis, but we did expect it to help us identify stronger sentiments due to the way that emotion is expressed in the English language. For example, we typically see people use an influx of adjectives when they are excited (either positively or negatively). Our POS tag incorporation did better at detecting sentiment, but not whether it was positive or negative. For example, the following sentences were both categorized at a 4:

- It 's an old story , but a lively script , sharp acting and partially animated interludes make Just a Kiss seem minty fresh .
- A lousy movie that 's not merely unwatchable , but also unlistenable .

We believe our model may have been inadvertently trained to associate an abundance of adjectives with a strong (positive) sentiment. This belief is further supported by the fact that our model was less likely to classify sentiment as a 0 after the POS tag incorporation.

We expected to see more of an increase in paraphrase detection and semantic similarity detection due to the comparison between sentences. For paraphrase detection, we believe our POS tag incorporation improved the way that our model handles uncommon or unknown words. For example, a sentence pair initially classified as not being a paraphrase is seen below:

- Who was the best Mughal ruler? Why?
- Who was the best mughal ruler?

Our prior belief was that incorporating POS tags would provide a fuller picture of the sentence structure, aiding in identifying paraphrase detection. We theorized that additional contextual information about the composition of each sentence would increase result accuracy. This was not the case, the addition of contextual information from our incorporation of POS did not result in an increase in

overall dev accuracy. While the POS tags improved task performance for sentiment analysis, these increases led to a slight decrease in paraphrase detection and semantic textual similarity, which resulted in an overall decrease in dev accuracy. Because we saw a pattern of oscillation in the dev accuracy between iterations, we believe we introduced too much complexity into our model causing it to not effectively train on the data. We believe adding the POS tag embeddings introduced too much noise for the model to effectively learn the parameters. When we tried to isolate the POS tag inclusion to just the sentiment analysis task, our results were inconsistent, leading us to believe task interference is occurring wherein changes to one task are intertwined with the performance of the other tasks.

5.2 Siamese BERT-Network

For the Siamese BERT-Network implementation, we aimed to implement the concatenation strategies to see an improvement in overall downstream task performance despite the single-node implementation of BERT. Similar to the way our POS tag implementation was constrained by impractical runtime, the full Siamese BERT-Network implementation faced significant challenges due to memory constraints. We wanted to explore how much we could improve our model using the concatenation techniques and just a single instance of BERT.

Unfortunately, our concatenation techniques did not result in an improved overall performance. We faced the same task interference challenges as previously where changes to a single downstream task resulted in an overall redistribution of performance in the other two tasks. We believe the additional embedding containing the absolute difference of the two vectors resulted in potential overfitting due to the complexity of the inputs. We think the additional vector was interpreted as noise instead of meaningful data, resulting in a loss of precision in our model.

While the idea of concatenating a sentence difference initially held promise, we think the collective impact of the memory constraints and overfitting led to the models inability to generalize effectively to new data.

5.3 Combined Iterations and Full Model

Notably, the most dramatic increase in performance both in leaderboard score and in runtime was when we successfully reconstructed our training architecture to allow full model to run on a smaller amount of data, overcoming memory constraints. What this indicates to us is that regardless of the specific extension or technique we applied at different training stages, the most valuable difference was allowing the model to train on and learn all BERT parameters, and not just linear layer parameters.

Although a major takeaway from this course has been the importance of data: its quality, quantity, and biases all inform the final product, in this case, the extremely large Quora dataset was actively making both running the model and making simple adjustments to code an unwieldy, days-long, and literally costly endeavor. This was a reminder that, when starting with a basic model like minBERT, having a solid basis for our code is more important than simply adhering to/following others' findings. Ultimately reducing runtime also made it much easier to test small changes to code. We just wish we had made this discovery sooner.

While the combined iterations model was effective in reducing runtime, we also saw that it significantly reduced the performance of STS compared to when its data was prioritized as the last training batch (as well as paraphrase, when it was the second-to last training batch). However, the dramatic improvement to Sentiment Analysis once each task's loss was weighted equally balanced this out.

5.4 Gradient Surgery

Ultimately, gradient surgery was our most effective improvement upon the combined iterations model. Compared to the combined iteration model, gradient surgery on its own caused a noticeable decrease in sentence similarity correlation compared to sentiment and paraphrase accuracies. Gradient surgery allowed sentiment classification accuracy to significantly exceed the 0.517 classification score we had obtained by running sentiment classification on its own. To exemplify this specifically, the review "Half Submarine flick , Half Ghost Story , All in one criminally neglected film" was initially misclassified as a 1, but was accurately classified as a 2 after applying gradient surgery. Similarly, the review "As surreal as a dream and as detailed as a photograph , as visually dexterous as it is at times

imaginatively overwhelming." was initially misclassified as a 3 and later accurately classified as a 4. These samples, among many others indicate that gradient surgery made our model more sensitive to extremities and more accurately discerned between nearby classes in sentiment classification.

Although the similarity correlation is still significantly lower than its best score, we suspect that the gradient normal plane projections implemented in gradient surgery significantly improved the model's steps towards the other tasks by slightly reducing STS component of the gradient, eliminating the steps lost towards other tasks in the formerly competing updates.

6 Conclusion

Our greatest takeway from this project is that runtime and memory consumption will always be a primary determining factor in performance and the flexibility in the rudimentary stages of developing a model like minBERT. Through implementing gradient surgery, we discovered that prioritizing data quantity was not always the right choice, and that the method in which we applied the learning's from the data was much more impactful to the model's overall performance. Repeatedly, we discovered task interference made improvements incredibly difficult. Specializing the model to perform well at one specific task (i.e. sentence similarity or sentiment classification) caused it to reduce the performance of another task. This was an underlying reason which contributed to many of our expected improvements' failing to improve overall accuracy score. Furthermore, when we implemented extensions which we only expected to affect one task (e.g. sigmoid and cosine similarity), we discovered it impacted all tasks.

Prior to implementing the combined iteration model, we were extremely limited by runtime, and we recognize that that has affected the quality of our early data and ability to iterate on under-performing extensions. Thus, although the POS tag and alternate Siamese BERT-Network implementations were not quantitatively successful, we believe the area of input concatenation still has potential. Given more time and computation resources, we are still interested in exploring alternate concatenation techniques and what contextual information our model could learn from different forms of data representation. Regardless, we succeeded at our goal of extending our familiarity with NLP and exploring unseen avenues of downstream task implementations. We also see potential to further improve the gradient surgery implementation, as it frequently reached its best dev performance in early epochs and then often got worse. This we can attribute to outfitting to the training data, and if we had more time, we would have likely tried to better tune the hyperparmeters, perhaps through increasing dropout rate.

7 Key Information

TA mentor: Soumya Chatterjee
Contributions: Monica primarily owned POS tagging and Siamese Networks. Megan owned Gradient Surgery and other miscellaneous extensions, but both of us made small modifications/started runs based on the others' extensions. We contributed equally to the baseline model, training strategies, and writing the paper and poster.

8 Ethics Challenges, Societal Risks, and Mitigation

There are a variety of risks in this project, especially when training on a task of generalized sentence embeddings.

Looking at specific applications of our tasks, we discover more ethical implications associated with our minBERT implementation. Consider using sentiment analysis for a movie recommendation system. The effect of incorporating our extensions changed the way our model evaluated data; as we saw in our analysis section, incorporating POS tags harmed the model's ability to distinguish between extremely positive and extremely negative movie review sentiments, potentially basing the overall sentiment on something such as adjective count. An issue such as this could percolate up if it was incorporated into a recommendation system that used sentiment analysis to provide movie suggestions, potentially recommending movies that are hateful or in bad taste. This is a common problem in machine learning in general where the addition of parameters and layers creates an obfuscation that makes it hard to establish the exact factors the model is basing decisions off of. A

mitigation strategy could be do to some model auditing; we could use the example ids to track how our model classifies reviews with extreme sentiments, particularly reviews with trigger words such as "racist" or "misogynistic".

Another specific task application is using paraphrase detection to organize or prioritize questions or comments in a question forum like Quora or social media outlet like twitter/X. If all questions or posts are merged into just one or the first one, this could greatly reduce the nuance of any dialogue users can have. Furthermore, if these posts are ordered by number of paraphrases it has it could directly affect the perceived importance of a subject. Thus, if we utilize paraphrase detection as a metric, we could innaccurately hide or make certain questions, concerns, or messages effectively invisible. To mitigate this, the interface should be designed so that it does not utilize these metrics, and if it does, still randomly includes other questions so that a user's "feed" does not become an echochamber of a uniform, majority-only perspective.

Ultimately, in human-written data, human biases and errors emerge, as demonstrated in the word embeddings for gendered and professional words in Assignment 1. This will only be further exacerbated in any sentence-level data, and intensive care needs to be put towards screening and testing for any specific applications before these seemingly powerful, general-use models are applied to any real-world scenarios. Some of the common uses for generalized sentence embeddings might be for automated forum moderation, detecting spam/scam messages/posts/emails, and even for proofreading applications. A biased model based on biased training data may eventually cause ableist, racist, and sexist determinations/classifications that unfairly censor discussion, communication, and even deny financial opportunities to already marginalized groups of people. The most ideal way to mitigate this is to produce unbiased datasets for all NLP purposes, and in our case, diversify the training data with writing and labeling that is conscious of and challenges these biases Another mitigation method might be to create test datasets that are focused on specific minorities (e.g. women, disabled persons, ethnic minorities, etc.) and non-minorities to evaluate whether models contain severe biases or not, and use this to choose models that are most appropriate/equal for all persons.

References

Alexandra Benamar, Meryl Bothua, Cyril Grouin, and Anne Vilnat. 2021. Easy-to-use combination of pos and bert model for domain-specific and misspelled terms. In *NL4IA Workshop Proceedings*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Monica Hicks and Megan Liu. Github repo.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Ubiquitous Knowledge Processing Lab (UKP-TUDA)*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc.

A Appendix

Our team github repo can be found at: https://github.com/MonicaHicks/CS224nFinalProject Hicks and Liu