

Uncertainty-Aware Self-Training for Paraphrase Detection

+ Learned Reranking for Sonnet Generation with GPT-2

Stanford CS224N Default Project

Alex M Michael
Department of Computer Science
Stanford University
alexmm@stanford.edu

Luis Marc Botin-Sanz de Sautuola
Department of Computer Science
Stanford University
lmbotin@stanford.edu

Xander Hnasko
Department of Computer Science
Stanford University
xanderh@stanford.edu

Abstract

We implemented GPT-2 for the default project. Our extension focused mainly on uncertainty-aware self-training (UST) for paraphrase detection with limited labeled data. The second extension is a Best-of- N sonnet generation with a learned reranker, trained using chrF supervision. In parallel to these we used gradient clipping, a linear learning rate scheduler, weight decay, cloze-style prompting, symmetric inference, and threshold calibration to bring dev accuracy to 0.903. For the first extension we showed that UST improves over seed-only training and classic self-training in the low-label regime ($K \in 500, 1000$ labeled examples per class), where the teacher model is weak enough that variance-based pseudo-label filtering improves student quality. At higher K (20,000 per class), the seed alone already produces a strong enough teacher and UST brings little further gain. Since collecting labeled data is often the main bottleneck in modern ML systems, this makes a strong case for UST when labels are scarce or expensive. For sonnet generation, the learned reranker reached 41.875 dev chrF, which exceeded other decode only baselines.

1 Key Information

- Mentor: Ahmed Ahmed
- External Collaborators (if you have any): None
- Sharing project: No
- Late Days (if you are using any):

Team member	Late days remaining
Alex M Michael	4
Luis Marc Botin-Sanz de Sautuola	1
Xander Coulter Hnasko	2
Total pooled	7
Team size	3
Late days applied for the report	2

2 Introduction

Our project focused mainly on paraphrase detection, motivated by the uncertainty-aware self-training (UST) paper of Mukherjee and Awadallah [1]. Our main research question was whether uncertainty-aware pseudo-label selection could maintain paraphrase accuracy in settings where labels are scarce (simulated by accessing only a fraction of the Quora labels). The motivation comes from the constraint in machine learning work of accessing high-quality labeled data. Labeled data usually requires manual human input or domain expertise which makes collecting large labeled datasets expensive and hard to scale. Unlabeled data on the other hand is usually abundant and easier to get. This motivates semi-supervised learning approaches that take advantage of large unlabeled datasets and rely on only a small labeled seed. UST is one such method. It adds pseudo-labeled examples selectively based on model uncertainty in order to expand the effective training set without introducing too much label noise.

As an additional extension we explored Best-of- N sonnet generation with a learned reranker trained to predict quality using chrF supervision. Our goal was to test whether a learned selector could improve over plain decode-only generation by choosing among multiple candidate continuations for the same prompt.

In parallel, we built on the default requirements of the paraphrase detection task substantially. We first used gradient clipping, a linear rate scheduler, weight decay, and cloze-style prompting, bringing dev accuracy on the paraphrase task to 0.895. Then we used symmetric inference and threshold calibration to further improve to 0.903.

3 Related Work

As above, our UST extension is based on the Mukherjee and Awadallah paper on uncertainty-aware self-training (UST) for few-shot text classification [1]. UST extends classic self-training by explicitly estimating predictive uncertainty, selecting pseudo-labeled examples according to an uncertainty criterion, and then optionally weighting pseudo-labeled losses by confidence. This framework is especially relevant to paraphrase detection because the Quora dataset is large and the task is binary, with GPT-2 already being able to produce an interpretable yes/no prediction through its LM head.

Our second extension relates to stochastic decoding and reranking for generation. Nucleus sampling improves generation quality by restricting sampling to a high probability nucleus instead of the full vocabulary [2]. Best-of- N decoding takes this even further by adding a second step where the model generates many candidates and chooses one using a separate ranking rule.

4 Approach

The changes we made are in task framing, training strategy, and inference (no external datasets and no modifications to the transformer architecture itself).

Sentiment classification

For sentiment analysis we implemented the standard GPT-2 classification head from the starter code. It takes in an input sentence and creates contextualized hidden states, then using the last token representation as the sentence summary. This representation is passed through dropout and a linear layer in order to predict the sentiment label. We evaluate finetuning (both in last layer and the full model) on SST and CFIMDB. Sentiment is not the focus of our extension but it is a good sanity check that our implementation of GPT-2 works as expected.

Paraphrase detection

We first implemented:

- Gradient clipping (by constraining the global gradient norm to at most 1.0 each step and thus preventing unstable updates from large gradients). [3]
- Linear learning rate schedule that decays from 1.0 to 0.1 through training
- Weight decay of 0.01 in AdamW to regularize parameters and avoid overfitting [4]

Using this setup we reach 0.895 dev accuracy. We further improved this accuracy, first by using a cloze-style prompt:

Is " s_1 " a paraphrase of " s_2 "? Answer "yes" or "no":

GPT-2 processes this prompt and predicts the next token. Instead of adding a separate classifier head we take the model’s vocabulary logits at the last position of the prompt and keep only the logits for no and yes. The paraphrase detection task effectively turns into binary prediction of the next token.

We make this stronger by adding two more modifications at inference. First we use symmetric inference. Because order doesn’t matter for paraphrase we can evaluate both (s_1, s_2) and (s_2, s_1) and average the logits [5]. We also use threshold calibration. Instead of fixing the paraphrase threshold at 0.5 we sweep thresholds on the dev set and choose the best value. The threshold for the strongest run was 0.47. This improved accuracy to 0.903.

UST for paraphrase detection

We simulate a setting with scarce labels by making a stratified labeled seed with K examples per class and treating the remaining training examples in the Quora dataset as an "unlabeled pool". A teacher model is trained on this seed set. We then estimate uncertainty over the unlabeled pool, construct, and finally select pseudo-labeled examples. These pseudo-labeled examples are combined with the labeled seed, and a student model is trained on this combined dataset from pretrained GPT-2 weights (to avoid inheriting any bias from the teacher).

We compare UST against two baselines in the same low-label regime. The first is standard seed-only fine-tuning. The second is classic self-training (CST), which is different from UST in how it picks pseudo-labels. In CST, the teacher runs a single forward pass (dropout off), and for each unlabeled example we take $\max(p(\text{yes}), p(\text{no}))$ as a confidence score. The top 30% most confident examples are selected with uniform weight. The limitation with this approach is that it’s common for models to be overconfident, making softmax probability a bad proxy for label correctness.

MC dropout addresses this by optimizing for the stability of the prediction instead of confidence. If the teacher predicts “yes” with high probability on every one of 10 stochastic MC dropout passes, that signals the label is reliable and probably correct. If we instead see large swings (depending on which neurons are dropped), this signals that the label is unreliable. Variance across passes captures this stability in a way softmax cannot. We also report a weighted UST variant where each pseudo-label’s loss contribution is scaled by $1 - v/v_{\max}$ (lower variance get assigned higher weight). However, we saw that in practice the selection step mattered more than any weighting.

Sonnet generation with a learned reranker

We train GPT-2 on the Shakespeare sonnet corpus, and then provide the first three lines of a sonnet as a prompt and ask the model to generate the remaining lines. At inference time, we sample many candidate continuations for the same prompt and use a separate learned reranker model to pick the final output.

For a candidate sonnet x_i consisting of both the prompt and the generated continuation, GPT-2 creates hidden states for each token $H_i = (h_{i,1}, \dots, h_{i,T_i})$. We condense this data for the candidate with

$$r_i = \left[\frac{1}{T_i} \sum_{t=1}^{T_i} h_{i,t} ; h_{i,T_i} \right],$$

effectively appending the mean pooled hidden state to the last token hidden state. A separate MLP then maps this representation to a scalar reranker score

$$s_i = f_{\theta}(r_i)$$

To train the reranker, for each prompt, we generate several candidate continuations and compute chrF scores y_i (on just the continuation) against the reference continuation. We then optimize a pairwise ranking loss so that candidates with a relatively higher chrF receive higher scores

$$\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j) \in \mathcal{P}} \log(1 + \exp(-(s_i - s_j)))$$

$$\mathcal{P} = \{(i, j) : y_i > y_j\}$$

At inference the reranker scores the sampled candidate pool and picks the continuation with the highest predicted score.

5 Experiments

5.1 Data

Task	Train	Dev
SST sentiment	8,544	1,101
CFIMDB sentiment	1,707	245
Quora paraphrase	283,003	40,429
Shakespeare sonnets	263	25

For the UST experiments we made a labeled seed and then used the rest of the Quora training data as an unlabeled pool. We kept the hidden labels only for to evaluate the quality of pseudo labels. K is the number of labeled examples per class, so the largest low-label setting we test ($K = 20,000$) uses a total of 40,000 labeled examples which is 14% of the 283,003 examples in the Quora training set.

5.2 Evaluation method

Per the handout, we evaluate both sentiment and paraphrase using dev set accuracy. For the low label experiments in paraphrase we also track dev F1. For sonnet generation we evaluated by chrF on the held-out dev sonnets. We compare the results against the posted baselines (0.882 for paraphrase and 41.414 for sonnet).

5.3 Experimental details

To recap the base GPT-2 we implemented, the model has 12 transformer layers, hidden size 768, 12 attention heads. Sentiment and paraphrase are finetuned with AdamW. Our best paraphrase setup for full data uses 10 epochs, batch size 8, learning rate schedule, gradient clipping (at most 1.0 each step), weight decay of 0.01 in AdamW, symmetric inference, and threshold calibration.

For UST the logged experiments use $K \in \{500, 1000\}$ and a last efficiency test with $K = 20,000$. We use selection rate 0.3 (bottom 30% variance), and $T = 10$ MC-dropout passes. MC dropout relies on the dropout layers that already exist in GPT-2 (with $p = 0.1$), which are kept active when estimating uncertainty by running the model in training mode without computing gradients.

The student model is always initialized from pretrained GPT-2 weights instead of teacher checkpoint. We borrowed this from Mukherjee and Awadallah [1] with the idea being that the teacher might have overfit to the small seed set, so starting the student from pretrained weights lets it learn from the larger combined dataset without inheriting the biases of the teacher.

We found that student models overfit quite quickly on pseudo-labeled data so we reduced student training to 5 epochs (teacher kept at 10).

CST uses the same seed split but picks the pseudo-labels with most confidence from a single deterministic teacher pass. The accuracy of the pseudo-label are calculated only for analysis (never used in train).

As for sonnet generation, the learned reranker uses a candidate pool that is drawn from checkpoints 5, 7 and 9. The simplified final reranker path uses $N = 3$ candidates for each checkpoint, max generation length 128, hidden size of reranker 128, and pairwise ranking loss. The reranker is trained on all the training sonnets available with 80/20 split internal for reranker training + validation. We also evaluate a plain decode only baseline for comparison in which GPT-2 makes a single continuation straight from the prompt without any learned reranking.

5.4 Results

Setting	Ours	Reference
SST last-linear dev	0.456	—
SST full-model dev	0.509	—
CFIMDB last-linear dev	0.857	—
CFIMDB full-model dev	0.971	—
Paraphrase dev (full-data baseline)	0.895	0.882
Paraphrase dev (full data, enhanced)	0.903	0.882
Paraphrase test (leaderboard)	0.901	—
Paraphrase dev (UST, $K = 20,000$, best)	0.8510	0.882
Sonnet dev chrF (plain decode-only)	40.646	41.414
Sonnet dev chrF (learned reranker)	41.875	41.414
Sonnet test chrF (leaderboard)	42.530	—

Table 1: Main development and test results. Ref values for paraphrase and sonnet are the posted baselines. The 0.8510 UST result is in low label setting and is not directly comparable to the full data results.

Method	K	Round	Dev Acc.	Dev F1	# Pseudo-labels	Pseudo-label Acc.
Standard fine-tune	500	—	0.7499	0.7389	—	—
Classic self-training	500	—	0.7587	0.7345	84,600	0.9080
UST (unweighted)	500	0	0.7580	0.7435	84,600	0.9047
UST (unweighted)	500	1	0.7676	0.7565	84,600	0.9416
UST (unweighted)	500	2	0.7651	0.7524	84,600	0.9284
UST (weighted)	500	0	0.7603	0.7481	84,600	0.9047
UST (weighted)	500	1	0.7680	0.7557	84,600	0.9409
Standard fine-tune	1000	—	0.7807	0.7698	—	—
Classic self-training	1000	—	0.7685	0.7636	84,300	0.9235
UST (unweighted)	1000	0	0.7897	0.7828	84,300	0.9469
UST (unweighted)	1000	1	0.7884	0.7788	84,300	0.9597
UST (unweighted)	1000	2	0.7880	0.7757	84,300	0.9571
UST (weighted)	1000	0	0.7845	0.7789	84,300	0.9469
UST (weighted)	1000	1	0.7847	0.7779	84,300	0.9544
Standard fine-tune	20,000	—	0.8490	—	—	—
UST (unweighted)	20,000	0	0.8510	0.8452	72,900	0.9939
UST (unweighted)	20,000	1	0.8494	0.8435	72,900	0.9943

Table 2: Ablation table of low-label paraphrase results for standard training, CST, and UST. K is the number of labeled examples per class

Paraphrase system	Dev Acc.	Labels used
full-data supervised	0.903	100%
fully-supervised ($K = 20,000$)	0.8490	$\approx 14\%$
UST ($K = 20,000$)	0.8510	$\approx 14\%$

Table 3: Label-efficiency comparison for paraphrase detection.

6 Analysis

6.1 UST

Our results show that UST is especially useful in settings where labels are scarce by optimizing label efficiency. UST improves over the $K = 500$ seed baseline, improves over CST at $K = 500$, and also beats the seed baseline at $K = 1000$. Interestingly CST actually hurt at $K = 1000$ (0.769 vs 0.781 for seed only). This tells us that naively adding pseudo-labels (which might be noisy) can drag down the student. UST avoids this by filtering through MC-dropout variance instead of softmax confidence.

As a final "efficiency" test we ran UST with $K = 20000$ UST setting. That run reached 0.8510 dev accuracy and 0.8452 dev F1 while using only about 14% of the original training labels. Comparing to our original paraphrase baseline of 0.903 (using all data) this performs 94% as well. In most modern ML systems where the issue is often the lack of quality labeled data, this is quite notable, as we see the model reach nearly the same performance with only a small fraction of the labels.

At $K = 500$ the accuracy of pseudo-label for examples using MC-dropout 0.905 compared to just 0.674 for the rejected pool. This tells us that selection based on variance works well. As K increases the teacher gets stronger and the gap becomes smaller since even the hard examples become easier. At $K = 20,000$ the selected pseudo-labels are above 99% accurate. In theory this creates a positive feedback loop of UST improvement. However, at $K = 20,000$ standard finetuning on only the seed gets 0.849 (which essentially matches 0.851 of UST). When the teacher is strong enough its predictions on the unlabeled pool are mostly correct regardless of whether we filter by uncertainty, so the pseudo-labels don't add much beyond what the labeled seed provides already.

The value of UST is instead in the regime where the teacher is weak enough to make real mistakes, and where variance-based filtering can actually separate good predictions from noise.

The low-label results show that such gains are quite sensitive to the seed size and training round and the improvement plateaus quickly. At $K = 500$, UST round 1 was the strongest and round 2 regressed slightly. At $K = 1000$, round 0 was already best. This implies that the system is vulnerable to cascading noise after the first round. It also means that it's easiest to understand our implementation as evidence that UST can help in cases where there is not much labeled data (but doesn't guarantee that more rounds necessarily increase performance).

The weighted-loss variant did not show a clear advantage over the unweighted version. At $K = 500$ weighted UST was competitive and slightly better in raw accuracy than the best unweighted run but not much of a difference. At $K = 1000$ the unweighted variant was clearly stronger. This says that most of the benefit of UST came from uncertainty-aware selection instead of from loss post-selection weighting.

6.2 Sonnet Re-Ranker

The sonnet reranker gave good results when using the pairwise ranking loss and chrF based supervision. In development evaluation the chrF of the reranker was 41.875. This improves over the plain decoder only baseline of 40.646 and also better than the posted baseline of 41.414. This shows that the separate reranker model is learning useful additional information and also that the summarized GPT-2 hidden states have enough information to tell apart stronger from weaker generated continuations.

A strength of the method is that it scores the full sonnet text rather than relying only on specific manually chosen features. By encoding the entire candidate with GPT-2 and summarizing it through the mean pooling plus the final token hidden state, the reranker can access higher level contextual information about the fit of the prompt and stylistic similarity. The pairwise ranking loss objective is also important since the real inference problem is to choose the best candidate among many alternatives for the same prompt instead of simply predicting an absolute score.

At the same time it needs mentioning that the reranker is trained using chrF scores between each generated continuation and the reference continuation, and chrF is also the evaluation metric. This is a reasonable choice because it aligns the training objective with evaluation but also means the reranker is optimized directly for the benchmark metric instead of toward some objective notion of writing quality.

7 Conclusion

We completed GPT-2 per the default project and extended it in two major directions. The core focus was implementing UST for low-label paraphrase detection, motivated by common bottleneck in AI for quality labeled data. Separately we used learned reranker for sonnet generation. The final project supports two main conclusions.

Our most noteworthy finding is the ability for UST to maintain almost all of its accuracy in the paraphrase task with only 14% of the labels. It improved over both seed-only finetuning and classic self-training in the low-label regime. UST’s main contribution is at lower K , where it consistently outperforms the seed and CST. With more labels and an already strong seed teacher, it seems to make less of a difference (and almost certainly does not warrant its greater ($\approx 3\times$) compute cost).

A second key result is that the extra learned sonnet reranker improved performance meaningfully over the decoding only generation and produced performance above baselines without the reranker.

The strongest non-UST paraphrase setup with all optimizations reached 0.903 dev accuracy. The learned sonnet reranker reached 41.875 dev chrF. Together these results show that base GPT-2 can still be meaningfully improved through both additional semi-supervised training strategies and separately learned candidate selection for generation.

Team contributions

Alex implemented the second batch of paraphrase improvements (which included the cloze-style yes/no classifier path, symmetric inference and threshold calibration), sonnet reranking experiments, submission-path debugging, and report/poster drafting. Xander implemented the UST pipeline at low labels on the VM, including few-shot split construction, MC-dropout uncertainty estimation, classic self-training, weighted and unweighted UST variants, and the main UST experiment logging, and also did most of GPT-2 training/debugging on GCP. Luis did the bulk of the experiment design, research framing for UST, the first patch of improvements for paraphrase (gradient clipping, learning rate scheduler, weight decay), experiment analysis, interpretation of the label efficiency result, and first pass at writing the final writeup. All team members contributed equally to project direction, interpretation of results, and revising the writeup.

References

- [1] Subhabrata Mukherjee and Ahmed Hassan Awadallah. Uncertainty-aware self-training for few-shot text classification. In *Advances in Neural Information Processing Systems*, 2020.
- [2] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.
- [3] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.
- [4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.
- [5] Haohan Wang, Da Sun, and Eric P. Xing. What if we simply swap the two text fragments? a straightforward yet effective way to test the robustness of methods to confounding signals in natural language inference tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.