Contrastive Representation Learning for Exemplar-Guided Paraphrase Generation

Advanced Natural Language Processing (Monsoon 2024)

Interim Submission

Rage against the Machine Learning (Team 33)
Bassam Adnan (2023121003)
Samyak Mishra (2022101121)
Varun Edachali (2022101029)

November 1, 2024

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

1 Data Processing

The current implementation contains two scripts to process the *QQP-Pos* and *ParaNMT* datasets. Both of them contain a few fundamental concepts:

- store the vocabulary of the dataset (the tokens in the train, and optionally validation set) and persistently store *word_to_index* and *index_to_word* mappings. Note that the only explicit data "cleaning" that is done is to convert the words to lowercase.
- store the indices corresponding to each token, for each sentence in the train, test and validation sets.
- store the bert-ids / encodings corresponding to each token for each sentence in the train, test and validation sets - corresponding to a pre-trained BERT model.
- store **similar sentences** for each sentence by style. More specifically, store upto 5 sentences that have a minimum edit distance from our current sentence, in terms of their parts-of-speech tags. This is the exemplar that guides our source to create the target during training.

Thus, we create a vocabulary for each dataset, store the indices corresponding to each line in the corpus' as well as the IDs assigned to them by a pre-trained BERT model, before finding the most similar sentences (by parts-of-speech tags, an indicator of style) for each sentence in the corpus.

During training, we take the source sentence to paraphrase (we maintain the semantics from here) and a sentence from the similarity list of the target sentence (we maintain the style / syntax from here) to create the target sentence. An example from the *quora dataset* is below:

Source Sentence: can you suggest a best budget phone below 15k ?

Similar Sentences (to Target):

- which mobile is better under 15k ?

  tags: ['WDT', 'NN', 'VBZ', 'RBR', 'IN', 'CD', '.']

- which bicycle should i buy under 10k ?

  tags: ['WDT', 'NN', 'MD', 'VB', 'VB', 'IN', 'CD', '.']

Target Sentence:

which phone is best to buy under 15k ?

tags: ['WDT', 'NN', 'VBZ', 'RBS', 'TO', 'VB', 'IN', 'CD', '.']

Note that the "tags" are the index_to_word mappings of the PoS tags of the respective sentences. For example, *NN* denotes a singular noun and *VBZ* denotes a verb.

Notice the similarity in content to the source, and the similarity in style (quantified by edit distance in parts-of-speech tags) to the similar sentences, shown in the target. Thus, the similarity list is integral to the training loop.

NOTE: the number of sentence-paraphrase pairs in the datasets are as below:

	QQP-Pos	ParaNMT
train	137185	493081
test	3000	800
valid	3000	500

2 Training the Model

Since we don't have a pre-trained model, we had to train a model from scratch. The motivation for this was to have a baseline we can compare to as we train our own model. With the default parameters, this was a non-trivial task due to versioning between dependencies and code errors. Furthermore the size of the datasets and the models make it time consuming. Regardless, we have trained the QQP dataset with the default configuration (hyper-parameters) as prescribed in the paper.

Parameter	**Value**
Word embedding size	300-d (GloVe-6B)
Style encoder E_s Type	BERT-based architecture
Style feature dimension	768
Content encoder E_c Type	GRU
Content encoder hidden state size	512
Teacher forcing rate	1.0
Balancing parameters (Style & Content Loss)	0.1
Optimizer	Adam
Learning rate	1e-4
Training epochs (ParaNMT)	30*
Training epochs (QQP-Pos)	45

Table 1: Model and Training Parameters

*Owing to the size of the dataset, we trained on 10 epochs

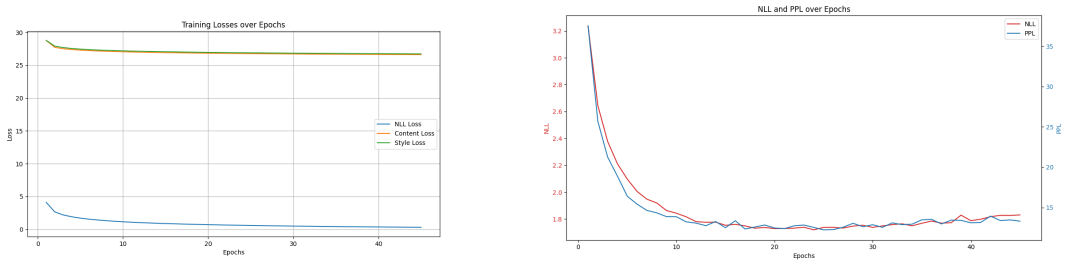
2.1 Loss Graphs

The authors have calculated Negative Log-Likelihood Loss NLL and for evaluating between epochs they calculate the PPL Loss, we are not sure what exactly is this metric, but the code suggests its a per-token/sentence metric at a specific mask value. The equation for the loss they minimize is given by:

$$L = \sum_{i=1}^{n} L_{nll}^{i} + \lambda_1 L_{ccl} + \lambda_2 L_{scl}$$

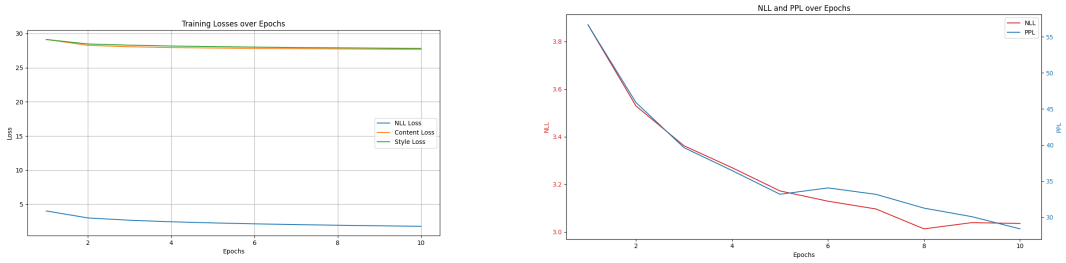
The values of the balancing parameters λ_1, λ_2 are given above, n is the batch size (128) and remaining terms are according to the paper.

The graphs during training and validation are given below:



(a) NLL, Content and Style Loss during training (b) NLL and PPL Loss (validation)

Figure 1: Training and Validation Losses on Quora dataset



(a) NLL, Content and Style Loss during training (b) NLL and PPL Loss (validation)

Figure 2: Training and Validation Losses on ParaNMT dataset

3 Results

We present the results on training the baseline model from scratch, the metrics we obtain are from following the instructions in the model repository. We compare the metrics against the baseline (Paper) and in turn benchmark them at different epochs. The motivation is to enable benchmarks of our own ablation studies at different epochs and compare its performance.

Epochs	**BLEU**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**METEOR**	**S-TED**	**T-TED**
15	41.120	68.330	48.605	70.864	41.507	6.313	7.430
30	44.480	70.310	51.679	72.607	44.442	5.953	7.478
45	45.400	70.518	52.063	72.734	45.095	6.040	7.656
Paper	45.8	71.0	52.8	73.3	45.8	-	-

Table 2: Model performance across different epochs on Quora dataset

Epochs	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	S-TED	T-TED
5	14.360	48.168	23.670	50.063	26.255	8.06	8.571
10	12.500	44.891	21.482	47.998	23.844	9.107	8.668
Paper	16.2	50.6	25.3	52.1	28.4	-	-

Table 3: Model performance across different epochs on ParaNMT dataset

The results suggest that the model does learn the patterns and reduces the loss, which is consistent with the validation loss. Moreover, the authors had trained their models till 30 epochs, while we had only trained for 10 epochs, the decrease in the metrics from the 5 epoch benchmark suggests that the model was learning new patterns while we had stopped it at the 10^{th} epoch. For the most part, our results come really close to the results given in paper.

4 Writing the DataLoaders from Scratch

For our own implementation of the paper, we began with the Dataset class, defined in `src/dataset.py`. We have studied the authors' implementation and kept our Dataset class similar to theirs. The authors are storing the data from the processed files, namely the source sentences, target sentences, word-to-index and index-to-word mappings, similarity, and the BERT indices files for source and target sentences. We are storing them too, except the authors had not used the word-to-index and index-to-word maps inside the Dataset class anywhere, so we have chosen to drop them to save space. They can, of course, be loaded elsewhere in the codebase whenever required.

In line with the authors' implementation of the code that fetches items from the dataset (namely, the `__getitem__` method), we are processing the source and target sentences at the requested index. The sentences are trimmed, if necessary, to a maximum allowed length. From the target sentence, we return the content, and two copies, one prepended with the Start-of-Sentence token and the other appended with the End-of-Sentence token. We also return the corresponding BERT IDs of the words in the source, target and exemplar sentence, which we choose randomly from the similar sentences (to the target sentence) as stored persistently for each of the train, test and validation datasets.

Other than this, in the same file, we have added some basic logging code that can print the shapes of the first few entries in the dataloaders created from the train, test and validation sets, as well as the sentences in the first entry, for manual verification if needed.

5 Conclusion

Having processed and analyzed the data, we believe we have gained an in-depth understanding of the task at hand. We train the baseline model from scratch, the modified source code, models and the evaluation code are linked below and are attached as part of submission. Being on schedule with our deadline, we plan to tweak dimensions of the BERT-based architecture and try different BERT architectures, switch to LSTM models for context encoding etc. upon the completion of our model from scratch, which will be our target for the final submission.

5.1 Pipeline

With this in place, we have processed the data, obtained their BERT token indexes, created DataLoaders for the same. Furthermore, we have also set up evaluation scripts which will help us immensely when we test our model during ablation studies.

5.2 Links

- GitHub Org - Link
- QQP Training Notebook - Link
- ParaNMT Training Notebook- Link
- Dataset - Link

5.3 Submission

Our zip file contains the following:

- `CRL-EGPG-From-Scratch`: The codebase where we implement the paper on our own, currently containing data processing and dataset class codes.
- `NLP-EVAL`: Standard code to evaluate the model.
- `REPORT.pdf`: Well, our report (this document).