Advanced NLP **Due: December 11, 2021**

Advanced NLP Project Report

Name: Ahish Deshpande, Jai Bardhan Roll Numbers: 2018102022, 2018113008

Objective of the Project

We have done both the **baseline** and the **bonus** part of the project.

Thw aim of our project was to implement a novel method for performing Natural Language Inference based on local text alignment and aggregation, using **A Decomposable Attention Model for Natural Language Inference** (Parikh et al. [2016]) as the baseline, and evaluating its performance on a standard dataset. As the bonus for this project, we have also incorporated **Intra-Sentence Attention** (Cheng et al. [2016]) as an addition to the above baseline, in order to improve its performance.

Introduction to the Task

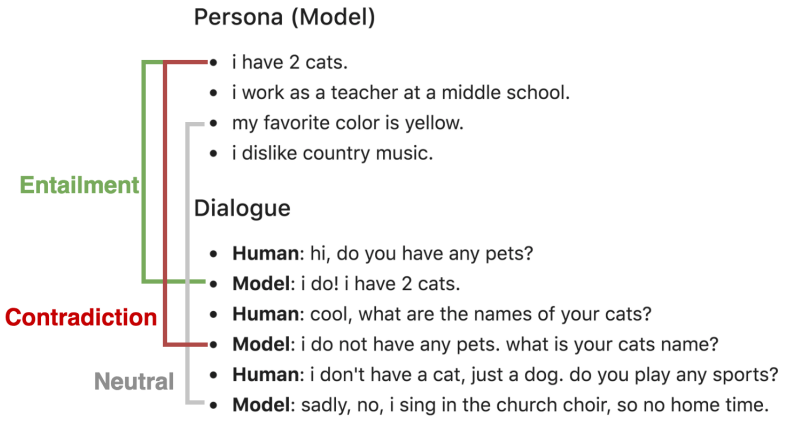

Figure 1: Example of Natural Language Inference Task

Natural Language Inference is the task of looking at a premise and a hypothesis and determining entailment, neutral and contradiction relationships with the given context sentences, as shown in the figure above. This task has traditionally been tackled by building complex models for text representation using CNN (Convolutional Neural Networks) or LSTM (Long Short Term Memory) based models.

In this project, we have used a novel approach wherein we align bits of local text substructure and then aggregate this information.

Our Approach to the Task

We tackled the Natural Language Inference task by aligning bits of texts between sentences using soft neural attention. We implemented the method as suggested by the authors of the papers on decomposable attention models and intra-sentence attention (Parikh et al. [2016], Cheng et al. [2016]). The authors of the paper argue that the task of Natural Language Inference can be easily solved by using **alignment between specific words in pairs of sentences**.

To support their argument, the authors design a **decomposable attention model** to align words between sentences by allowing the attention mechanism to weigh the words in the sentences. Through the attention mechanism, the authors generate a **global level representation** which is then used by a subsequent classifier to classify whether it is in agreement or contradictory.

By simply aligning words in sentences using attention, the authors are able to achieve competitive performance to the SOTA model while having significantly **fewer parameters** and a **highly parallelizable model** (leading to faster training and inference times).

Embeddings and Dataset(s)

- We make use of the Stanford Natural Language Inference Dataset (SNLI) (Bowman et al. [2015]), as it is the standard benchmark used for evaluation natural language inference models. The Stanford Natural Language Inference (SNLI) corpus (version 1.0) is a collection of 570k human-written English sentence pairs manually labeled for balanced classification with the labels *entailment*, *contradiction*, and *neutral*.
- We make use of the GloVe Word Embeddings (Pennington et al. [2014]) as it is able to capture the global statistics, ie. word co-occurrences in its word vectors

Implementation

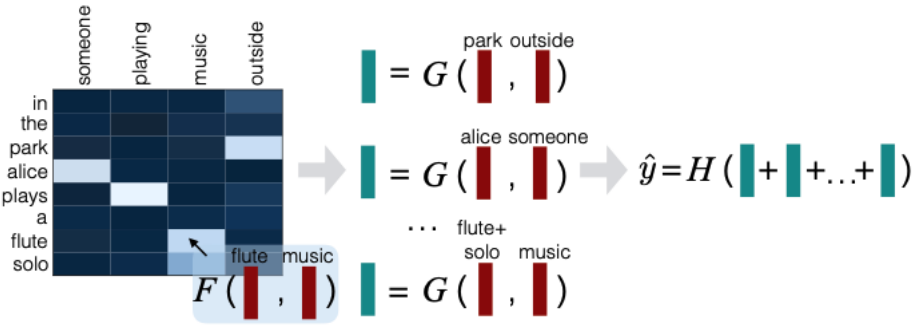

Figure 2: Model Pipeline (Parikh et al. [2016])
Stage-1: Attention, **Stage-2:** Comparison, **Stage-3:** Aggregation

We have implemented the decomposable attention model using `PyTorch` and `PyTorch Lightning`. Our code is fully vectorised and parallelizable over multiple GPUs using `DistributedDataParallel`. Our code is also modular and clean and uses objects completely from the `PyTorch` ecosystem.

The baseline model can be divided into three logical components at a high level:

- **Attention**: In this stage of the model, we soft-align the words of the two input sentences, using the idea of neural attention (Bahdanau et al. [2015]). Once this is done, we can then proceed with decomposing this task into comparison of aligned subphrases.

$$e_{ij} := F'(\bar{a_i}, \bar{b_j}) = F(\bar{a_i})^T F(\bar{b_j}) \tag{0.1}$$

The weights are then normalized as:

$$\beta_i = \sum_{j=1}^{l_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_b} \exp(e_{ik})} b_j,$$

$$\alpha_j = \sum_{i=1}^{l_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{l_a} \exp(e_{kj})} a_i. \tag{0.2}$$

Here β_i is the subphrase in $\bar{b}$ that is softly aligned to $\bar{a_i}$ and vice versa for α_j.

- **Comparison**: In this stage, we compare each of the aligned subphrases found in the first stage, which gives us a set of vectors that represent a non-linear combination of the first sentence and its softly aligned subphrase in the second sentence, and vice versa

$$v_{1,i} := G([\bar{a_i}, \beta_i]) \qquad \forall i \in [1, \ldots, l_a] \tag{0.3}$$

$$v_{2,j} := G([\bar{b_j}, \alpha_j]) \qquad \forall j \in [1, \ldots, l_b] \tag{0.4}$$

where $[\cdot,\cdot]$ denotes concatenation and G which is a feed-forward network.

- **Aggregation**: In this stage, we simply aggregate the set of vectors in the previous step, and use the result to predict the final label for the given pair of sentences

$$v_1 = \sum_{i=1}^{l_a} v_{1,i}, \qquad v_2 = \sum_{j=1}^{l_b} v_{2,j} \tag{0.5}$$

which is then fed to H a feed-forward network.

$$\hat{y} = H([v_1, v_2]) \tag{0.6}$$

- **Bonus: Intra-Sentence Attention:** In addition to the above stages of the baseline model, we have also added **Intra-Sentence Attention** as a bonus task. This is helpful as instead of using simple word embeddings, intra-sentence attention allows us to encode compositional relationships between the words of each sentence, thus augmenting the attention stage of the baseline model (Cheng et al. [2016]).

$$f_{ij} := F_{\text{intra}}(a_i)^T F_{\text{intra}}(a_j) \tag{0.7}$$

where F_{intra} is a feed-forward network.

$$a'_i := \sum_{j=1}^{l_a} \frac{\exp(f_{ij} + d_{i-j})}{\sum_{k=1}^{l_a} \exp(f_{ik} + d_{i-k})} a_j \tag{0.8}$$

The distance-sensitive bias terms $d_{i-j} \in \mathbb{R}$ provides the model with a minimal amount of sequence information, while remaining parallelizable.

For real-time logging the training progress, we have used `Weights and Biases`. This allowed us to decide in real-time whether a particular configuration performed well or not.

Main Results

We attained the results from the paper with the configurations mentioned in Table 1. However, due to the lack of exact code, we have tested a few different configurations to give us as similar of a performance as possible.

Another implementational gripe is the definition of the intra-sentence attention. The paper lacks a complete description of the intra-sentence attention mechanism to augment the inputs. Thus the code has been written purely as per the formulas mentioned in the paper.

It is crucial to note that our results are supposed to be somewhat worser due to the difference in the implementation of `max_length` and dataloader. The original code throws away examples of the dataset that might be harder to train/test on, while we still train on those examples.

	Hyperparameters					Accuracy		
S. No.	**Sentence Length**	**Intra**	**LR**	**Optimizer**	**Epochs**	**Train**	**Dev**	**Test**
1	10	No	0.05	Adagrad	250	78	72	77
2	10	No	0.001	Adagrad	250	71	68	67
3	42	No	0.05	Adagrad	150	84	83	82
4	10	Yes	0.001	Adam	100	65	62	63
5	10	Yes	0.05	Adagrad	150	72	69	69
6	42	Yes	0.05	Adagrad	150	80.4	78.3	77.4
7	83	Yes	0.05	Adagrad	150	80.3	78.9	78.4

Table 1: Results of different configurations tested

Plot 3 shows the Validation training curve for the best performing model in our case.

Ablative Studies

1. Changing the max sequence length:

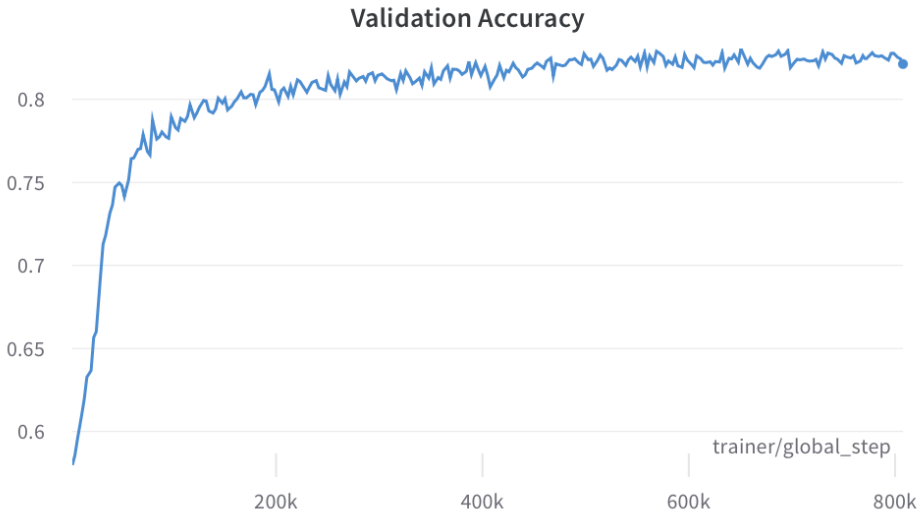

Figure 3: Validation Accuracy for Run 3

We have tested the performance of the models by changing the maximum sequence length of the sentence that are allowed. It is crucial to note that when the actual sequence length is larger than the maximum sequence length set we truncate the sentences to maximum sequence length.

The main trend that we see is that the performance increases as we allow the maximum sequence length to increase, and this is in line with the expectations as now the attention method has more of the sentence to align information. Interestingly, we see that we get a larger performance boost going from `max_length=10` to `max_length=42`, than from `max_length=42` to `max_length=83`. This can again be explained by noting two things: (1), the amount of information increase in the first case is more substantial than in the second case, and (2) there are not many examples in the dataset with crucial words at position 42 and beyond.

The difference in the validation accuracy between the 3 sequence lengths is shown in Figure 4.

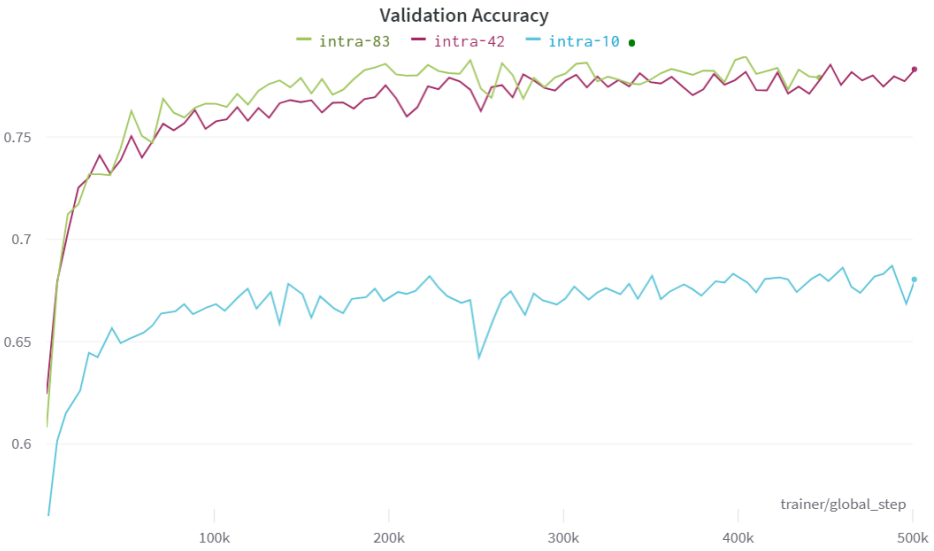

Figure 4: Validation Accuracy between the 3 sequence lengths.

2. Optimizer:

We tested two optimizer, namely: (1) `Adagrad` and (2) `Adam`. Of the two optimizers, Adagrad gave the better results. This is again expected as per the authors' note on this matter. What they failed to mention was the reason for this, and we see that the Adam optimizer in fact learns much faster than Adagrad, but also overfits and falls into strong local

minimas for training dataset. This leads to poorer performance on the validation dataset. A possible further direction for ablation would be to use some form of lr scheduler like the `CosineLRScheduler` with the Adam optimizer, which could prevent it from falling into such kinda of local minimas. However, as the original authors did not employ any lr scheduling, we have abstained from such studies here. We show in Figure 5, the comparision between Adam and Adagrad. We can see the exact trend that Adam learns faster at first.

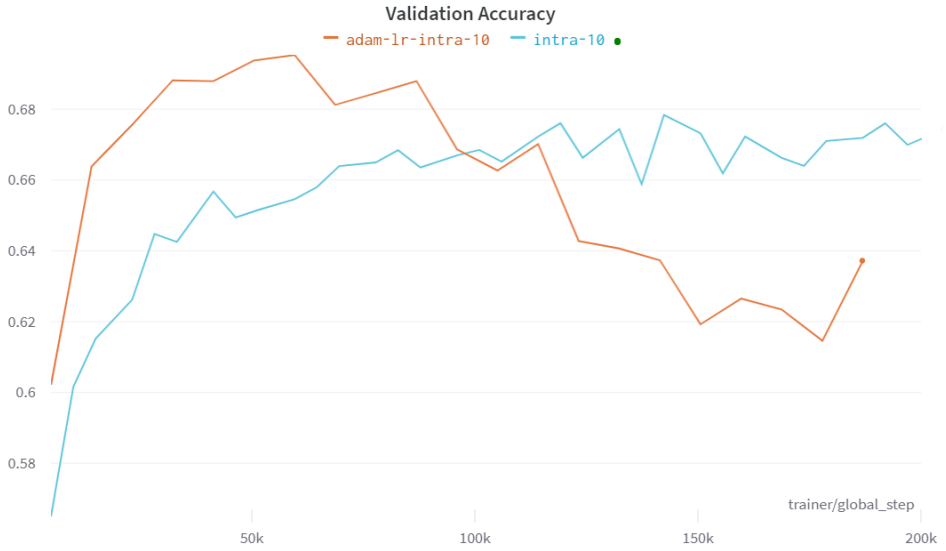

Figure 5: Validation Accuracy between Adam and Adagrad.

3. Learning Rate:
A lower learning rate here reduces the speed of the convergence and thus by the end of same number of epochs we get a slightly worse performance.

Computational Power Utilised

In order to run all our experiments, we made use of two ADA accounts, to train our models with 2 GPUs and 30 CPUs using `DistributedDataParallel`. Each hyperparameter configuration took about 6 hours to train with the above mentioned resources.

Inference Speed

We find that for the case with `max_length=83` and Intra-Sentence Attention, the time per batch is 0.0047296 seconds on a single gpu.

References

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference, 2016.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading, 2016.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation, 2014.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine trans- lation by jointly learning to align and translate, 2015.