Introduction to NLP

Natural Language Inference

Team - 21

Project Summary

Problem Statement

NLI: Say we are given two text fragments, one named text (t) and the other named hypothesis (h). The task involves recognizing whether the text can infer the hypothesis. This task is of particular importance because two or more sentences can mean the same meaning, and although it might be obvious for us as humans to identify it is fairly challenging to ask for a model. This is especially useful in various settings, including detecting fake news and contract reviews. All these tasks are time-consuming for humans so we want to automate them with decent accuracy. Hence the usage of NLP models in this area is an exciting prospect.

DataSets

SNLI Dataset

The Stanford Natural Language Inference (SNLI) dataset is a natural language understanding dataset introduced in 2015 by Stanford University. It contains over 570,000 sentence pairs labeled with one of three categories: "entailment," "contradiction," or "neutral," and is widely used to train, evaluate and benchmark for NLI models

MultiNLI Dataset

MultiNLI is a natural language understanding dataset released in 2017 by the Allen Institute for AI. It contains over 433,000 sentence pairs from 10 different genres, labeled with "entailment," "contradiction," or "neutral." MultiNLI is modeled after the SNLI dataset schema but covers a range of spoken and written text. Therefore, MNLI can be used in

conjunction with SNLI and offers text from ten different genres It is used to evaluate the generalization ability of NLU models.

Flow of Project

Preprocessing

We have preprocessed the datasets. We padded them and trimmed them based on the requirements of the model. This is apart from the regular tokenization and other routines which are followed by all the models.

Baseline Model

In our project, we have taken LSTM based model as a baseline. We have implemented everything from scratch. The salient features of this model are that we are using neural networks to get various insights into how premise and hypothesis are related. This is much better than other models which are statistical-based. Using neural networks we can now train the embedding as well along with the other learnable parameters. But of course, there are inherent difficulties with this model which we will discuss later in the challenges faced section.

Improvements done

Based on the last few lectures on the transformers and how powerful they are we were really inspired to include this in our project. Unlike LSTM-based architecture which is better than RNN in terms of solving gradient descent problems, they still do suffer from it. Transformers which include various components like attention help us to avoid these kinds of situations. And there are models like BERT which are already pre-trained on large datasets. And these kinds of pre-trained models can be used and be trained on specific datasets and specific downstream tasks. Why is this important? This is very useful because the

3high-level objectives will be learned in the pretraining process, and specifics related to downstream tasks can be learned with relative ease considering only finetuning. This gives a massive boost in terms of computation.

Interactive Interface

We have also provided snippets that are easy for the end users to check if the two statements are contradicting, entailed, or neutral to each other. These also helped us to verify the correctness of the model on some of the common and general test cases which are used in day-to-day life. As we go along we will provide some of the results our best models gave.

Results

Whenever you are implementing models generally we tend to need more direction in which we would love to go. This is especially true if the code and the logic are involved. To judge how good our models are and to compare the results with different models we have considered the validation, training, and test accuracy of the models and we have trained them on different parameters.

Relevant resources we followed

Throughout the project, we were stormed with different kinds of questions regarding the design choice. We took the Stanford course offered in the year 2021 lecture videos available on youtube as our primary resource for doubts concerning the theory. We have referred to various medium articles, and the PyTorch documentation page, and for the most common errors encountered during the training and importing libraries we have used Stackoverflow as our primary platform for doubt clarification. We also referred to various TAs at different points in time for their valuable advice on how we should approach the project and accordingly scope the project. We have also used the reference books mentioned to clarify any doubts.

Ideas and Experimentation

The primary interest of us taking this project was to explore the idea of finding how similar two sentences were. This idea was novel to us, but being so new to this field and early in the course we do not know how to approach the model. So a significant amount of time was actually spent on how to actually tackle the problem at hand. But one well-known idea

is to use neural networks. In one of our previous courses, we learned how powerful neural networks are. And exploring a little bit we found that the same power can be used in NLP applications. So here comes our first cue on how to tackle the NLI problem. One of our friends suggested the idea of embedding the words. Although I am able to explain what it is now, it is not so intuitive at the beginning of the course. So our first adventure began with training using LSTMs.

We were able to get some cool results with it. The sense that the model we trained was able to give some good results when we interactively check for few results. But the real test for the model is running on test data. After seeing the result we were devastated by the fact that the accuracy was only 65%. We read more resources as to how to improve the accuracy of the model which in turn helped us to jump this accuracy to 75%. Which is still lower than what they quoted on the internet. This is partly because of the limited training we were able to perform on the local machine. Given large enough data and computational resources, we are pretty sure that the accuracy can be boosted to 90% at least.

After listening to some of the online lectures we came across transformers. We studied some of the cool aspects of it like how it provides long-term dependencies. And this we thought will be useful for the NLI task. We explored the transformers further. We saw how the multi-headed attention module helps in alignment problems that were not able to solve in NN. Interesting advantages lead us to explore the BERT model. Which we in turn implemented. And the result: We were able to get 98% accuracy within epoch on training on limited data. This is an instant improvement from the previous LSTM-based architecture we implemented. Then we explored various other models which are slight modifications of the BERT model but improve in accuracy and runtimes. Some of the models we touched upon are RobertA, DistillBERT. Then we thought of exploring what more we could do, we tried different things like adding a neural network to the BERT which might be able to improve further. In our limited training, we found that both of them are almost performing equally well.

Experimentation involved dealing with different models which are at times based on different architecture. And in each model, we have different parameters like the hidden size, and number of epochs to run. The model which gave the best results. Apart from this

the kind of loss function to use, learning parameter, embedding size and batch size and how batching is performed, and more importantly the trimming threshold we defined in various models. And drawing various graphs out of the model's results to decide which of the models is the better outcome.

Outcomes

We were able to get a model which will able to tell how the premise and hypothesis are related. It will classify them into one of the three tasks. Namely, if they are neutral, entailment, or contradiction. After training, we also printed several statistics concerning the validation dataset like accuracy, and error along with the ones with training data. We also provide an interactive interface where two sentences can be entered and we will check how they are related.

Challenges faced

Setting up jupyter notebook has been the primary issue, there have been memory leaks while running the model. The batch size needs to be adjusted appropriately based on the memory constraints in the collab. The computation time for running each model is huge so for debugging or inferring anything from the model you need to wait a significant amount of time. And owing to the large runtime there were various errors relating to idle time error. Large dataset size, it needs to be in the same session as the collab session, which would require us to attach the drive to the current session. Figuring out the correct version of the various libraries which are used. There are many instances when the library versions were conflicting. We built a virtual env to check which of the combination works for us. And then parallel computation is something that would save a lot of time, but writing the same in collab took us a significant amount of time. Finding the actual algorithm meant that we need to go throught the papers of the original models which took effort.

Analysis of errors

There have been various errors like size mismatch etc. We have printed the shapes of the sensors we are passing as the input in such cases. In the cases where we were not getting the dimensions of the desired length, we checked the intermediate results of the NN layers.

And apart from these, we checked the tokenizing and padding and trimming functions by checking the top 100 sentences or so. In certain cases, inputs has to be cleaned up. When ever we encountered an error with the model we have dry ran the code, and analyzed the different components we wrote. And as a second resort we have run it for a small number of samples and checked possible errors. All these are to make sure that we saving on the computation resources. Figuring out the correct version of the various libraries which are used. There are many instances when the library versions were conflicting. We built virtual env to check which of the combination works for us.

Some of the results

Only a few epochs have been shown here, for complete statistics as a file in the submitted files.

BERT model top results

100%|██████████| 2498/2498 [08:48<00:00, 4.72it/s]

100%|██████████| 250/250 [00:14<00:00, 16.94it/s]

Epoch: 01 | Epoch Time: 9m 3s

Train Loss: 0.639 | Train Acc: 72.14%

Val. Loss: 0.341 | Val. Acc: 87.57%

100%|██████████| 2498/2498 [08:51<00:00, 4.70it/s]

100%|██████████| 250/250 [00:14<00:00, 16.97it/s]

Epoch: 02 | Epoch Time: 9m 5s

Train Loss: 0.377 | Train Acc: 85.92%

Val. Loss: 0.208 | Val. Acc: 93.80%

100%|██████████| 2498/2498 [08:50<00:00, 4.71it/s]

100%|██████████| 250/250 [00:14<00:00, 16.95it/s]

Epoch: 03 | Epoch Time: 9m 5s

Train Loss: 0.240 | Train Acc: 91.65%

Val. Loss: 0.107 | Val. Acc: 96.90%

100%|██████████| 2498/2498 [08:51<00:00, 4.70it/s]

100%|██████████| 250/250 [00:14<00:00, 17.04it/s]

Epoch: 04 | Epoch Time: 9m 5s

Train Loss: 0.162 | Train Acc: 94.57%

Val. Loss: 0.065 | Val. Acc: 98.32%

100%|██████████| 2498/2498 [08:51<00:00, 4.70it/s]

100%|██████████| 250/250 [00:14<00:00, 17.00it/s]

Epoch: 05 | Epoch Time: 9m 5s

Train Loss: 0.117 | Train Acc: 96.25%

Val. Loss: 0.048 | Val. Acc: 98.61%

100%|██████████| 2498/2498 [08:50<00:00, 4.71it/s]

100%|██████████| 250/250 [00:14<00:00, 17.01it/s]

Epoch: 06 | Epoch Time: 9m 4s

Train Loss: 0.089 | Train Acc: 97.14%

Val. Loss: 0.031 | Val. Acc: 99.15%

LSTM model top results

Epoch: 25

Training

100%|██████████| 12272/12272 [00:48<00:00, 253.93it/s]

Validation

100%|██████████| 307/307 [00:00<00:00, 563.53it/s]

Test

100%|██████████| 308/308 [00:00<00:00, 544.50it/s]

Epoch: 25 Train Loss: 0.8525959626957288 Train Accuracy: 0.616767930899257

Epoch: 25 Val Loss: 0.916709728660335 Val Accuracy: 0.5801324503311258

Epoch: 25 Test Loss: 0.9056701356327379 Test Accuracy: 0.5783157038242474

Validation Classification report

	precision	recall	f1-score	support
entailment	0.68	0.41	0.51	3479
neutral	0.55	0.58	0.56	3123
contradiction	0.56	0.77	0.65	3213
accuracy			0.58	9815
macro avg	0.59	0.58	0.57	9815
weighted avg	0.60	0.58	0.57	9815

Test Classification report

	precision	recall	f1-score	support
entailment	0.66	0.42	0.51	3463
neutral	0.52	0.58	0.55	3129
contradiction	0.58	0.75	0.65	3240
accuracy			0.58	9832
macro avg	0.59	0.58	0.57	9832
weighted avg	0.59	0.58	0.57	9832

Epoch: 26

Training

100%|██████████| 12272/12272 [00:48<00:00, 253.68it/s]

Validation

100%|██████████| 307/307 [00:00<00:00, 541.04it/s]

Test

100%|██████████| 308/308 [00:00<00:00, 377.66it/s]

Epoch: 26 Train Loss: 0.8519464958910836 Train Accuracy: 0.6184027583256515

Epoch: 26 Val Loss: 0.9179715605434455 Val Accuracy: 0.6055017829852267

Epoch: 26 Test Loss: 0.9251190527499497 Test Accuracy: 0.6043531326281529

Validation Classification report

	precision	recall	f1-score	support
entailment	0.52	0.87	0.65	3479
neutral	0.65	0.39	0.49	3123
contradiction	0.81	0.52	0.63	3213
accuracy			0.61	9815
macro avg	0.66	0.60	0.59	9815
weighted avg	0.65	0.61	0.59	9815

Test Classification report

	precision	recall	f1-score	support
entailment	0.52	0.88	0.65	3463
neutral	0.64	0.38	0.48	3129
contradiction	0.82	0.53	0.64	3240
accuracy			0.60	9832

macro avg	0.66	0.60	0.59	9832
weighted avg	0.65	0.60	0.59	9832

Epoch: 27

Training

100%|██████████| 12272/12272 [00:48<00:00, 254.04it/s]

Validation

100%|██████████| 307/307 [00:00<00:00, 562.81it/s]

Test

100%|██████████| 308/308 [00:00<00:00, 536.93it/s]

Epoch: 27 Train Loss: 0.851361037874315 Train Accuracy: 0.6173077804543904

Epoch: 27 Val Loss: 0.9362718565844558 Val Accuracy: 0.6012226184411615

Epoch: 27 Test Loss: 0.9498773010133149 Test Accuracy: 0.596826688364524

Validation Classification report

	precision	recall	f1-score	support
entailment	0.56	0.79	0.66	3479
neutral	0.53	0.56	0.55	3123

contradiction	0.85	0.44	0.58	3213
accuracy			0.60	9815
macro avg	0.65	0.60	0.59	9815
weighted avg	0.65	0.60	0.60	9815

Test Classification report

	precision	recall	f1-score	support
entailment	0.56	0.79	0.65	3463
neutral	0.52	0.55	0.54	3129
contradiction	0.86	0.44	0.58	3240
accuracy			0.60	9832
macro avg	0.65	0.59	0.59	9832
weighted avg	0.65	0.60	0.59	9832

Some of the interactive results we got from BERT model

```
premise = 'I am lying down on bed.'
hypothesis = 'I am resting on bed.'

predict_inference(premise, hypothesis, model, device)
```

```
'entailment'
```

```
premise = 'I go to office on my personal car.'
hypothesis = 'I have to share office cab for reaching office.'

predict_inference(premise, hypothesis, model, device)
```

```
'contradiction'
```

```
premise = 'I love to play cricket.'
hypothesis = 'I enjoy playing football.'

predict_inference(premise, hypothesis, model, device)
```

```
'contradiction'
```

```
premise = 'He is techy.'
hypothesis = 'He has no idea of tech.'

predict_inference(premise, hypothesis, model, device)
```

```
'contradiction'
```

```
premise = 'I am using mobile phone.'
hypothesis = 'I have mobile in my hand.'

predict_inference(premise, hypothesis, model, device)
```

```
'entailment'
```