Assignment 2 - Relation Extraction on NYT29 Dataset

In this task, we are trying to extract the semantic relation between entities on the NYT29 dataset. We have been provided with the dataset containing more than 74,000 sentences each annotated with the entity pairs present in the sentence and the corresponding relation between the entities. The relations provided correspond to relations in the freebase knowledge graph
An entity can be a place like new york and another entity can be the united states, in this case, the relation would be “contains”.

Approach

To solve this task I have for the most part followed the approach defined by Soares et all in the paper “Matching-the-blanks” [1], this method entails using a Bert-base-uncased[2] model implemented by the hugging face library[3] and adding entity markers in the input data which are special tokens denoting the start and end of entities. When building an encoded representation of the entity pairs from the BERT sequence output I use these entity start tokens of both entities and use them as input to a feed-forward layer for classification. In the paper, this approach is called the BERT-EM approach.
I have explained my process and the major components of my solution below.

Dataloader

I have built a custom dataloader to parse the dataset provided and prepare it in the format the model would be comfortable with. Firstly the data has been provided in the text file format, so I start with reading the text files, I am mainly concerned with the ‘.sent’ file to read the sentences from and the ‘pointer’ file which will provide information on the position of the entities and the relation. One more file for the list of relations has been provided, I read that file and map each class string to a corresponding integer value which would be easier to process called a class-to-int mapping.

Before we begin processing the entity pairs and sentences we can talk about the tokenizer. Since I am using the entity markers approach, I will be adding special tokens [e1] to denote the start of the first entity, [/e1] to denote the end of the first entity, [e2] to denote the start of the second entity and [/e2] to denote the end of the second entity. So that the tokenizer can understand these new tokens they have to be added to its vocabulary. We perform this part and update the model with the new vocabulary later. The tokenization length that I have employed for this solution is 256 since it can be observed that most of the reviews are within that word limit and a lower length also helps with the training time.

As provided in the task, we are not to assume all the entity pairs have been given, and hence we add another class to our class-to-int mapping for the class denoting no relation I call it here “no relation”. Now for each (sentence,entity_pairs) I permute all the entity pairs possible and if that entity pair has a relation class in the dataset I use that otherwise I use the “no relation” class. As I observe in the data entity pairs can have multiple classes so while one-hot encoding

the labels I consider all the classes for that entity pair. After doing this preprocessing step, the number of training samples increases considerably. Now, on this data, I apply tokenization which takes each input sentence and converts it into tokens which are then converted into IDs that are understandable by the model. The tokenizer provides with 'input_ids', "attention_mask", "token_type_ids" which I store. Along with this, I store the position of entity start markers in the tokenized data.

Model

For this task, I have used the Bert-base-uncased model to get the encoded representations of the sentences. This is a large language model trained on the task of masked word prediction and next sentence prediction. In my approach, I am fine-tuning based on relation extraction as a downstream task.

Firstly the model will provide us with the sequenced output of the input sentence which contains the encoded representation of all the words. In the input provided to the model, we are storing the entity start marker indexes, so we slice the sequenced input array, and get the encoded representations of the entity start markers for both entities using the indexes. Next, I just concatenate both these encoded representations. This tensor now is the representation of the entities. This tensor will be enough to represent both the entities and their corresponding relation.

Now, I will use a simple feed-forward linear layer and a dropout layer to classify the entity representation. Firstly we apply the dropout layer on the entity representation since this is generally a good practice, and also prevents overfitting within the model. Then we apply the feedforward linear layer.

Coming to the loss function, It can be observed in the data that this is a multilabel multiclass classification task, which implies that we cannot use a softmax function as the activation in my last layer, so we will apply sigmoid on each of the output logits and then use Binary cross-entropy[4] loss on that, this loss will ensure that each class becomes a binary classification task, allowing us to predict multiple labels.

Optimizers & Training

I have used the AdamW[5] optimizer with the learning rate of 5e-5 and a linear scheduler that linearly reduces the LR until it reaches 0. I have used AdamW as its a very effective optimizer, it maintains per parameter learning rates and changes them accordingly. AdamW improves upon the Adam optimizer and changes the weight decay method used. The Model takes 6 hours to complete one epoch and within the limits of google Colab of 6hours, I am able to train for up to 1 epoch safely without losing GPU access. During training, the loss reduces to 0.0001 and the accuracy on the training set is around 99%

Observations & Hyperparameters

For this task, how the model architecture is being developed played a very key role. Specifically, it matters how the entity representation is being created. In my approach, I performed two experiments to create entity representation. These two methods were also specified by the paper, Firstly we can use mention pooling to create the representation, this approach entails max-pooling the entity encoded representation spans to create a representation for one entity and then doing the same thing for the other entity and concatenating these two tensors, The second approach is the one I have finalized this uses only the entity start markers and we the encoded representation of the entity start markers and concatenate them. I observed that although the first approach takes into consideration all spans ideally it should perform better than the second approach but in reality, the second approach performs a little better than the first one while only using the entity start markers.

I have summarized all the hyperparameters that I have used below.

Tokenizer:

Length : 256
Truncation: True
Padding: upto Max length

Model:

Model: Bert-base-uncased
Linear layer hidden size: 2 x encoded representation length for Bert(768)= 1536
Dropout probability: Model hidden dropout (0.1 for bert)
Number of Linear layers: 1

Training:

BatchSize: 8
Epochs :1
Optimizer :AdamW
Scheduler : Linear
Size of Train Dataloader with all entity pairs: 1,45,688
Loss: BinaryCrossEntropy

Inference:

Threshold: 0.5

Results

To calculate the performance of the model on the test, Val and train split I am using the sklearn f1 score function and I am not performing negative sampling on the data as we are only considering the entity pairs that have been provided to us. Furthermore, As told by the professor

in the lecture, I am excluding the last class label “other”. The results I received are provided below. Finally, the threshold on the logits has been set to 0.5

DATA SPLIT	F1
TEST	90.2086383601757
DEV	96.4142052637627
TRAIN	97.1172836380209

Further Discussion

In my current approach, I have used entity markers and the bert-base language model to solve this task and followed the Matching the blanks paper, however, in that paper they also designed a pre-training step to perform even better on this task. Right now, we had tagged relations data but relations are very difficult to annotate even for proficient language speakers hence a method to train without tagged relations has been derived. Considering entity representations of two sentences r1, r2 if they both contain the same relation their inner product will be high and low if they represent different relations. Now using this assumption they have a new pre-training step along with the BERT masked word prediction pre-training. In this step, blanks may be introduced in the sentences in place of entities, so the model is also able to understand the entities. This method will perform better on the general task and also can do few-shot training.

Other than this approach to solve this task models like Span-Bert[6] may be used which have done modifications to bert so the masked predictions are now masked span prediction.

Another interesting approach is modeling the relation classification task as a span prediction QA[7] task, this would involve pre-processing the data and converting it into questions and answers. In future work, there could be a couple of modifications and more experimentation to improve performance.

References

1. Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2895–2905.
2. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.
3. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface’s transformers: State-of-the-art natural language processing. CoRR, abs/1910.03771, 2019.

4. Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In Advances in Neural Information Processing Systems, pages 8792–8802, 2018
5. Loshchilov, I. and Hutter, F. (2017). Fixing weight decay regularization in adam. arXiv preprint arXiv:1711.05101.
6. Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. Transactions of the Association for Computational Linguistics 8 (2020), 64–77. https://doi.org/10.1162/tacl_a_00300.
7. Amir DN Cohen, Shachar Rosenman, and Yoav Goldberg. 2020. Relation Extraction as Two-way SpanPrediction. arXiv preprint arXiv:2010.04829.