NLP homework 1

Amine Ahardane: 2050689

Abstract

This report presents an approach to event recognition in natural language processing using deep learning techniques, specifically gated recurrent unit (GRU) neural network architecture. The model is trained to predict a label to the corresponding word given a sentence in the BIO format. We compare the performance of our model with random baseline and our baseline. Our experiments show that our method achieves competitive results, demonstrating the effectiveness of deep learning methods for event recognition in natural language processing. We also provide a detailed analysis of the behavior of the model and discuss its limitations and potential for future development.

1 Introduction

The neural netwrok architecture used is simple but still very effective. we will be using a pretrained word embeddings layer (Glove) a GRU layer and at the end a shallow linear layer. We also need to preprocess the text applying some tecnique to improve the model accuracy.

2 Text Processing

With glove we already have a vector rappresentation for every token in the training set. We considered every token as lowercase to reduce unnecessary variability and improve the accuracy of the model during word embedding tuning. By converting all tokens to lowercase, we ensure that words that differ only in case (e.g., 'Apple' and 'apple') are treated the same by the model, reducing unneccesary distinctions that do not contribute to the underlying semantics of the text

We decided to use 50 as the dimensionality in the GloVe embedding layer because our training dataset is relatively small. so using a high-dimensional embedding can lead to overfitting, adding unnecessary memory/processing power without improving model accuracy. Meanwhile a lower-dimensional embedding can improve generalization performance on unseen data.

We chose Glove because it behave very well in Named Entity Recognition tasks, because is able to capture the semantic relationships between words that are important for identifying named entities, taking into account the frequency of co-occurrence between each pair of words in the corpus.

I tried also to consider all the words in test set that are not in training set to be considered as unknown (keyword <unk>) but it worsen the model. Note that i assign the keyword <unk> for words that are not in GLOVE.

3 Model Architecture

So the model as we said is pretty straigthforward. One important note to add is that we used a GRU model with bidirectional set to true for event detection. The bidirectional parameter allows the model to take into account the context of the surrounding words on both sides of the target word, which is important for event detection as it involves identifying the beginning and end of an event within a sentence. By using a bidirectional model, the GRU is able to capture not only the past context of the word, but also the future context, allowing for a better representation of the entire sentence. This can be useful for event detection, where the labels for each word in the sentence depend on the labels of the neighboring words.

Howard Ruder and Kirkpatrick et al. suggested to freeze for the first several epochs the embedding layer. The problem is that when you start training your word embedding matrix, the gradients will be huge because your model will drastically underfit the data for the first few batches, and you will lose completely the pre-trained embedding matrix.

4 Document Body

4.1 Fine-tuning model

At first, we started with an LSTM layer, but I found out that the GRU layer is faster than LSTM, and in my case, LSTM behaved like GRU in terms of accuracy. The number of GRU units is 512, and adding more units or layers didn't improve the model's performance; it just made the model slower. Also, the bidirectional set to true did improve the accuracy by 4%. For the loss function, we used cross-entropy loss because we are dealing with multi-class classification. The optimizer we chose was SGD instead of the more commonly used Adam optimizer. SGD helped us achieve an f1 score value of 71%, and when I tried to use Adam, even with a low learning rate, the f1 score was not able to go further than 69%.

SGD optimizer iteratively update the model parameters in the direction of the negative gradient of the loss function. In this way the model tries to converge towards the global minimum given a loss funciton. On the other hand, Adam optimizer uses adaptive learning rates for each parameter in the model, which can help it converge faster than SGD in some cases.

4.2 Evaluation

We were able to achieve an (macro) F1 score of 71% and an accuracy of 94% on the test set. Probably if we add more samples to the dataset the score would improve, still using the GRU architecture. We could improve by alot the results by using a more advanced architecture such as the Transformer, which has shown to outperform RNN-based models in various NLP tasks when we have enough samples in the dataset. Let's take a look to the confusion matrix Figure 2. represented as a heatmap, a table used to evaluate the performance of a classifier. The heatmap (confusion matrix) shows the count of observations that fall into each cell, with the intensity of the color corresponding to the number of observations. From the heatmap, we can see that the actual class O is correctly classified 97% of the time, which is not surprising since O is the most common class in the dataset, meaning that it can learn and classify it more accurately than other classes. Also the model misclassifies I-ACTION as O 42% of the time, which is expected since the class I-ACTION has fewer examples compared to O. This is a issue that arise in multi-class classification when classes are imbalanced, and the model tends to misclassify the classes with low occurrences.We used heatmap becasue is very useful, it provides us a visualization of the model's performance and helps to identify where the model needs improvement. We can see also that we are well above from the random baseline in terms of accuracy, but our baseline(let's name it common-baseline) that considers the most common label for every word in the training set and uses it to classify instances in test set, has alot in common with the result of this model accuracy.Let's consider the diagonal. Common-baseline had an accuracy of 0 for classes like I-ACTION and I-CHANGE here we do not. The other accuracy are very similar to our common-baseline, so it seems that for some classes the neural model learned to take the more common label in training dataset like what the common-baseline do.The neural model behave better then the common-baseline for classes that are not so common like the example above, in which depending on the application the common-baseline can be enough and is very fast using almost no memory.

4.3 References

References

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification.

Neil Rabinowitz Joel Veness Guillaume Desjardins Andrei A. Rusu Kieran Milan John Quan Tiago Ramalho Agnieszka Grabska-Barwinska Demis Hassabis Claudia Clopath Dharshan Kumaran Raia Hadsell James Kirkpatrick, Razvan Pascanu. 2018. Overcoming catastrophic forgetting in neural networks.

A Appendix


```
{
  "O": 0.9784072688401924,
  "B-ACTION": 0.6801063358440408,
  "B-CHANGE": 0.750575594781274,
  "B-SCENARIO": 0.6422893481717011,
  "B-POSSESSION": 0.7272727272727273,
  "B-SENTIMENT": 0.64576802507837,
  "I-SCENARIO": 0.9148936170212766,
  "I-ACTION": 0.0,
  "I-CHANGE": 0.0,
  "I-SENTIMENT": 0.0,
  "I-POSSESSION": 0.0
}
```


Figure 1: Common-baseline results. for every class we have the accuracy percentage(how many label are correctly classified)similar to Figure 2 (diagonal values)

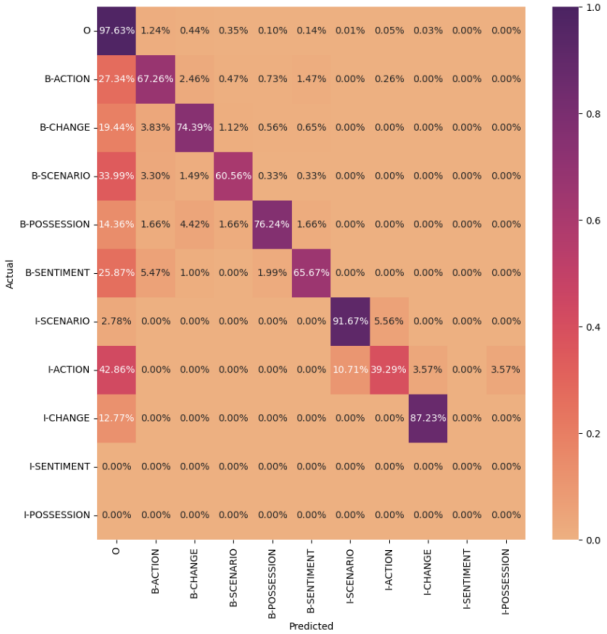

Figure 2: Confusion matrix