1. LR
   1. Introduction

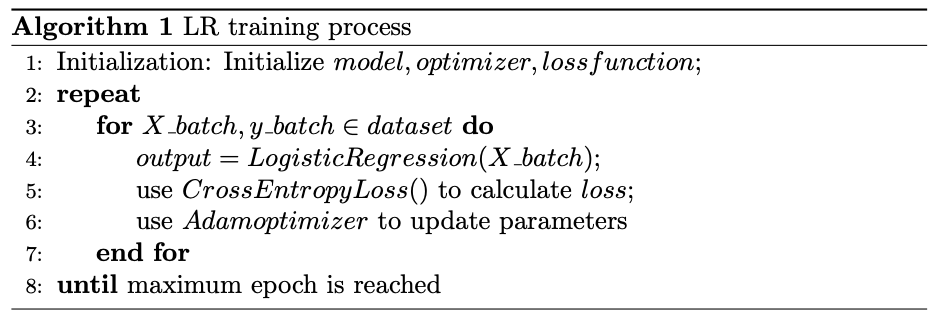
Multinomial Logistic Regression is the model that we first try to deal with the problem of our project. It’s the simplest and most intuitive way to handle the multi-classification problem. Our model is using a linear function to predict the topic of a vector of problem text and using cross entropy loss to optimize our model.

* 1. Model structure
  2. Hyperparameters setting

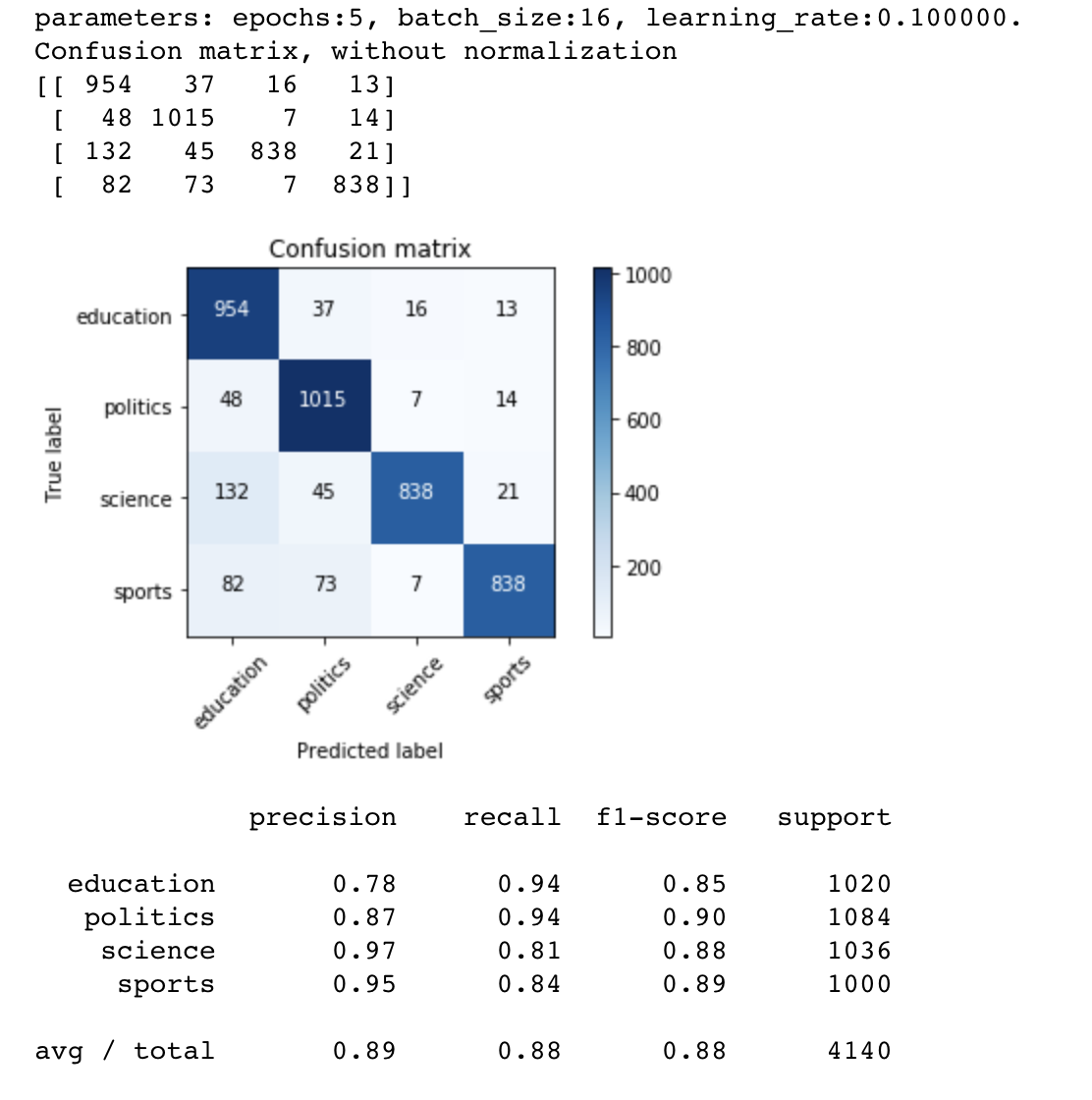
To find effective hyperparameters, we varied one hyperparameter at a time while keeping others fixed. Our choice is shown as below:

|  |  |
| --- | --- |
| Hyperparameter name | value |
| Epochs | 5 |
| Batch size | 16 |
| Learning rate | 1e-1 |

* 1. Training



* 1. Results



1. CNN
   1. Introduction

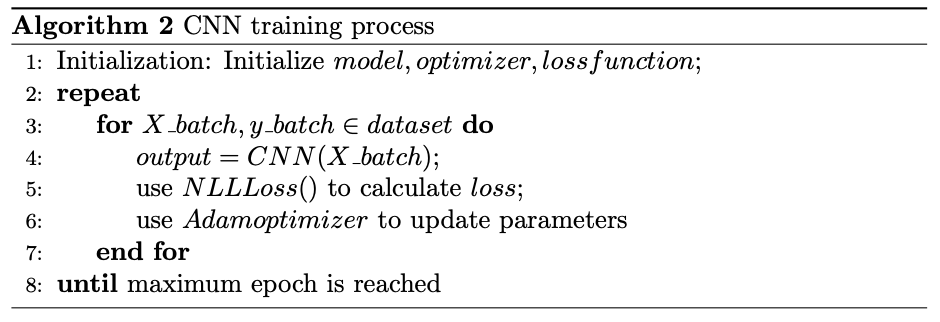
We notice that using Logistic Regression to handle the text classification problem is not an optimal solution because it ignores the relation between the words which is an important feature of a sentence. Therefore, we use a multi-layer CNN to obtain the local relationship between words of a sentence and see whether it can improve our model.

* 1. Model structure
  2. Hyperparameters setting

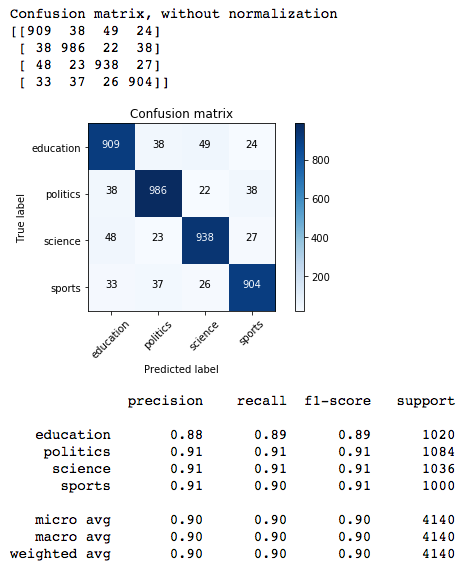
To find effective hyperparameters, we varied one hyperparameter at a time while keeping others fixed. Our choice is shown as below:

|  |  |
| --- | --- |
| Hyperparameter name | value |
| Epochs | 5 |
| Batch size | 16 |
| Learning rate | 1e-4 |

* 1. Training



* 1. Results



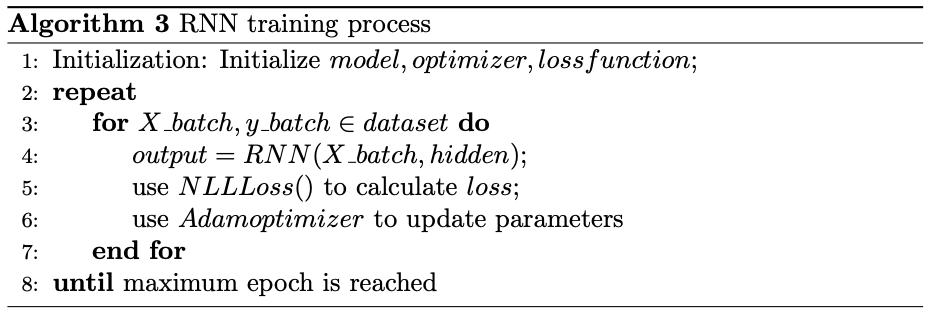
1. RNN
   1. Introduction

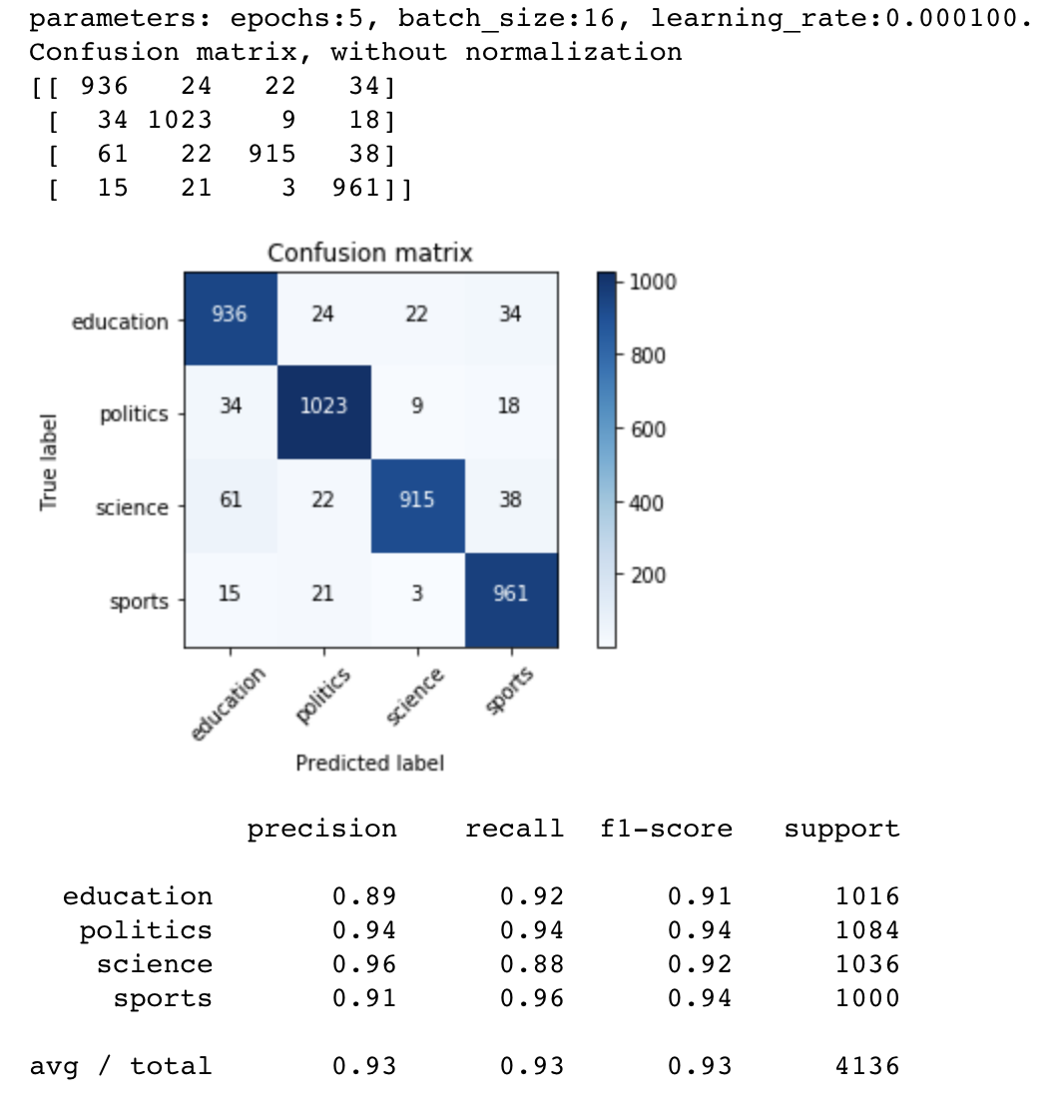
Although the CNN model can get some local relation features of a sentence, the information is constrained by the filter size of convolutional layer. In order to express the context features better, we use RNN as our improved model.

* 1. Model structure
  2. Hyperparameters setting

To find effective hyperparameters, we varied one hyperparameter at a time while keeping others fixed. Our choice is shown as below:

|  |  |
| --- | --- |
| Hyperparameter name | value |
| Epochs | 5 |
| Batch size | 16 |
| Learning rate | 1e-4 |

* 1. Training
  2. Results



1. LSTM
   1. Introduction

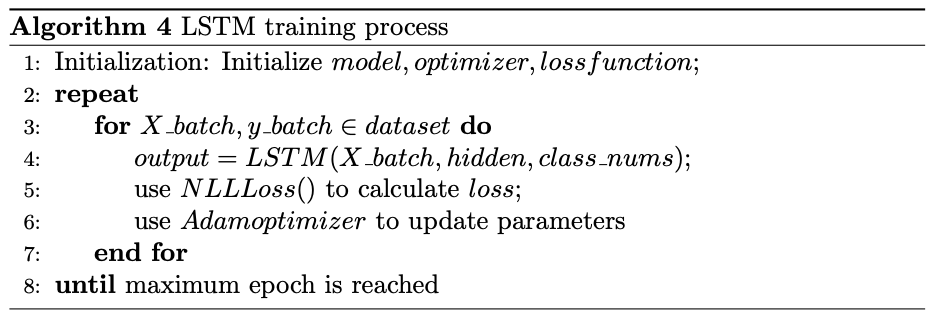
After training our model by using RNN, we have already got pretty good results in our test dataset. However, we want to find whether we can get a better result by using a more complicated model. Therefore, we use the LSTM to replace basic RNN model.

* 1. Model structure
  2. Hyperparameters setting

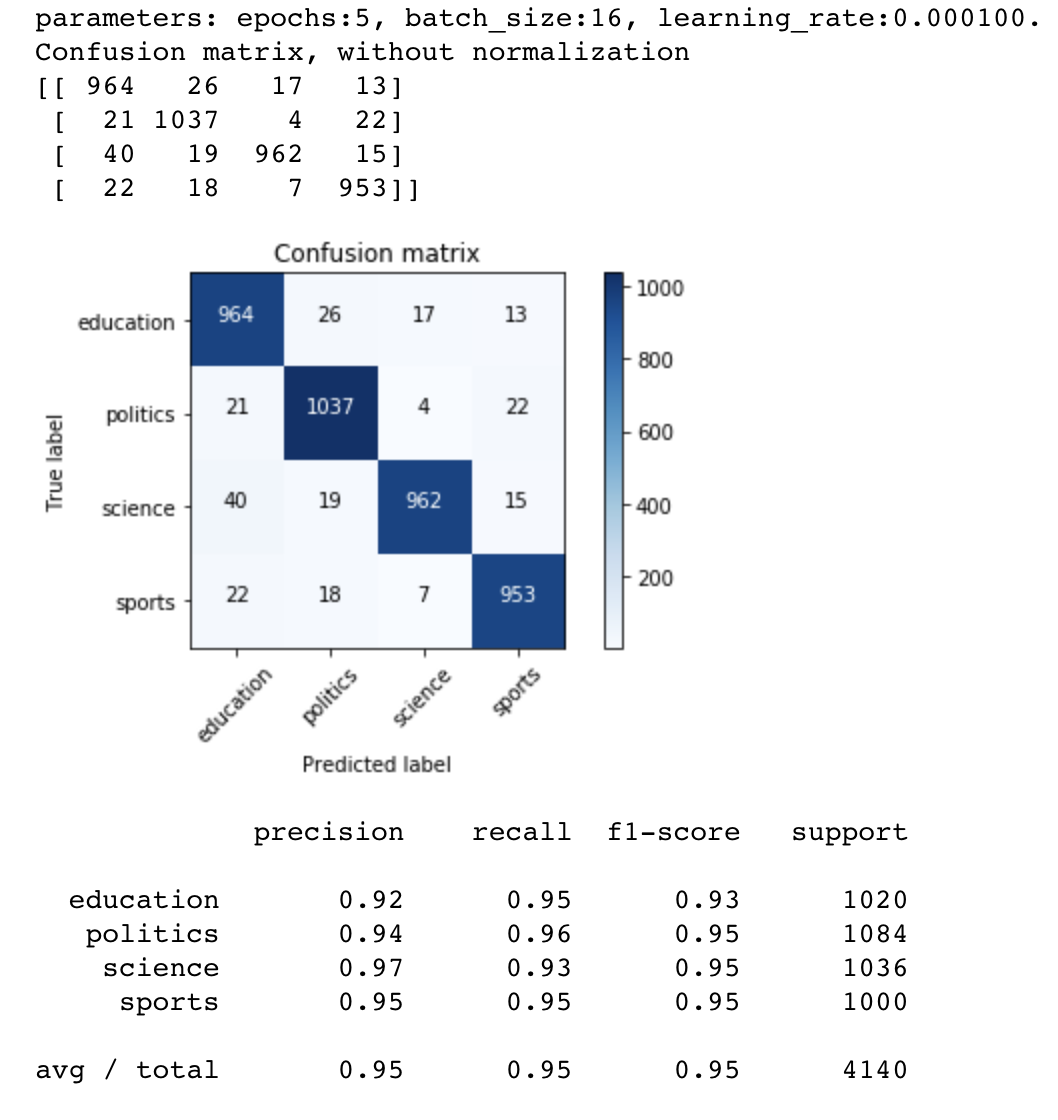
To find effective hyperparameters, we varied one hyperparameter at a time while keeping others fixed. Our choice is shown as below:

|  |  |
| --- | --- |
| Hyperparameter name | value |
| Epochs | 5 |
| Batch size | 16 |
| Learning rate | 1e-4 |

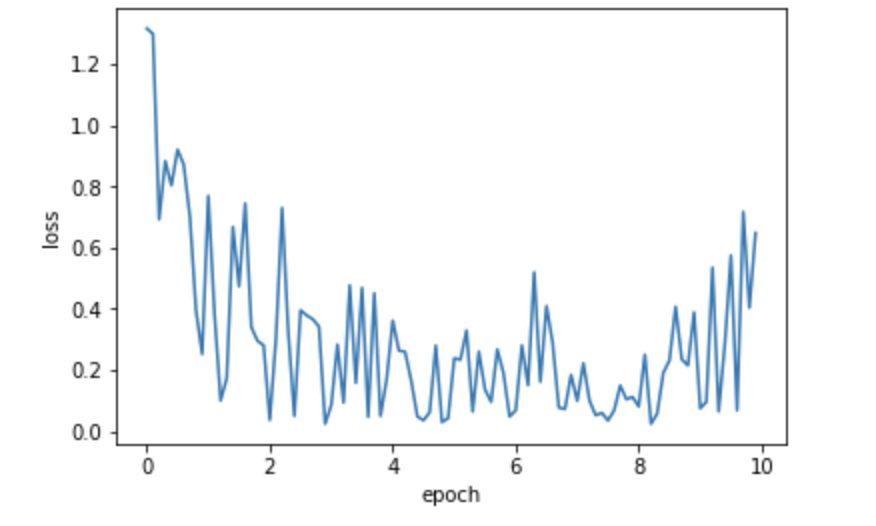
* 1. Training

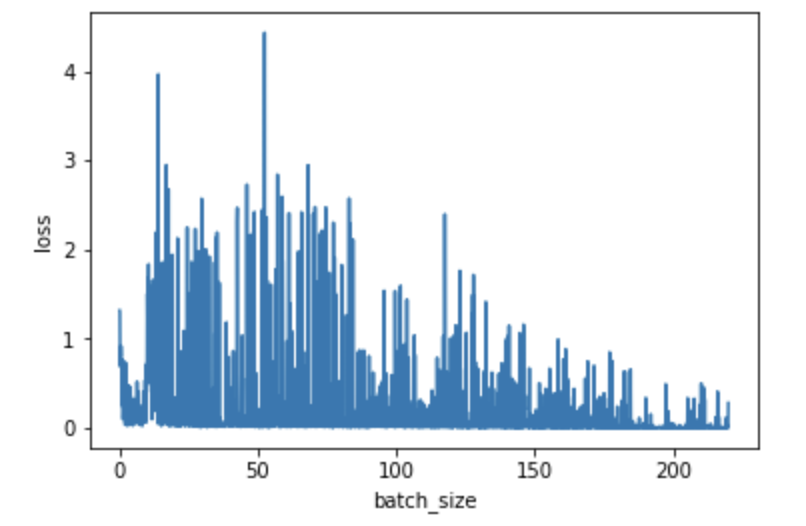


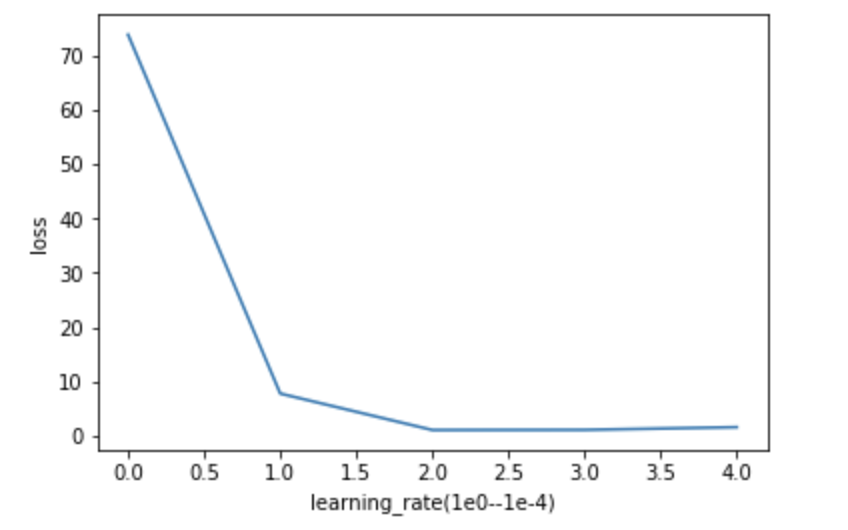
* 1. Results



Reference: (Some charts which show the hyperparameter finding process.)







Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | F1-score | Training time | Pros | Cons |
| LR | 0.88 | 3s | Simple and fast | Ignore relations between words |
| CNN | 0.90 | 59s | Can express local relation features between words | Can’t obtain features out of filter |
| RNN | 0.93 | 29s | Can express context features of the whole sentence | Can’t abandon useless information |
| LSTM | 0.95 | 80s | Can choose most useful information of a sentence, most precise | Too complicated and train slowly |