Character-Level Convolutional Networks for Text Classification

SMAI Project

Jay Ghevariya
Atharv Sujlegaonkar
Tirth Motka
Mitul Garg

TABLE OF CONTENTS

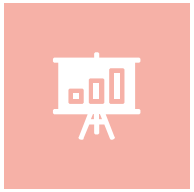
Abstract

This work looks into the use of convolutional networks on character level instead of traditional methods for text classification. Large datasets are used to show that results are as competitive as other methods. Comparison is done with models such as bag of words, n-grams, and deep learning models such as word-based ConvNets and RNNs.

PROJECT CHECKPOINTS

GOAL 1

Researching on the topic, exploring various techniques to implement the project

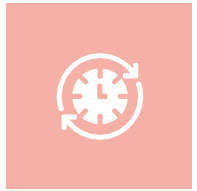


GOAL 2

Studying about the convNet and its implementation

GOAL 3

Implementation and drawing out useful inferences



GOAL 4

Error analysis and comparison of our model with different models/ transformers

TIMELINE

5th November

Project Allocated

1st & 2nd week of November

Paper and relevant work reading, discussions and initial project layout.

15th November

Mid-Evaluation

3rd Week of November

Implementation of core algorithm.

4th Week of November

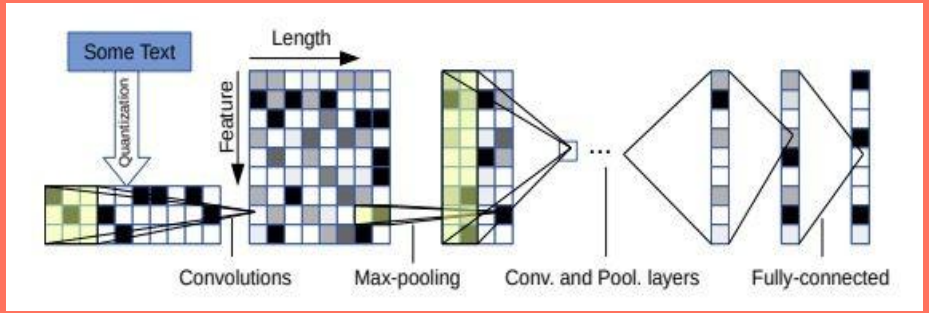
Comparisons/ Modifications in the model and Error Analysis & Final Report

Introduction

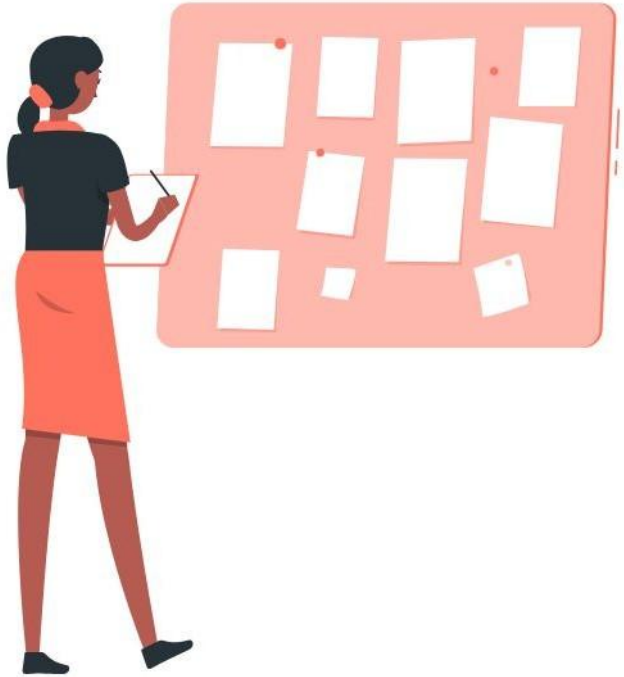
- The ConvNets based text classification model has enabled us to introduce deeper ConvNets with more than 3 layers which provides the model with the ability to extract data in a better way.
- Deeper Neural networks enable to train the same network structures for variety of datasets, and obtain optimal results.

Layer	Large Feature	Small Feature	Kernel	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

Layer	Output Units Large	Output Units Small
7	2048	1024
8	2048	1024
9	Depends on the problem	



Theory



Datasets used

Datasets

AG’s News

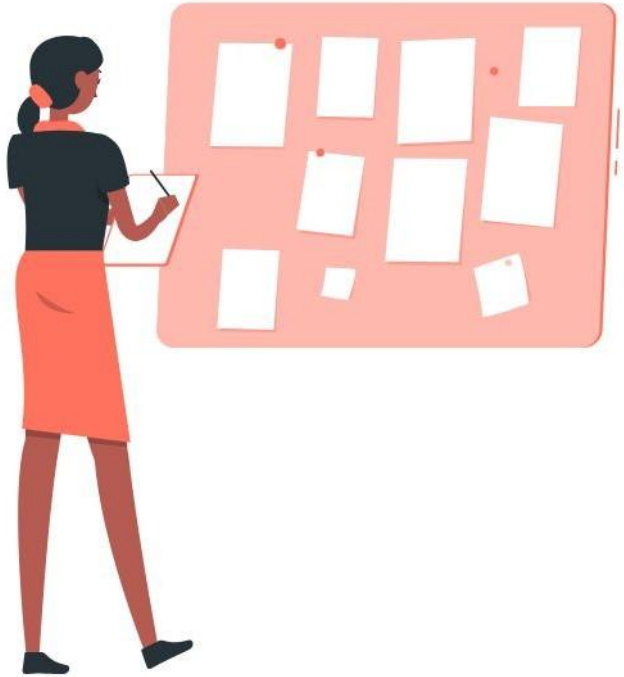
The AG's news topic classification dataset is constructed by choosing 4 largest classes from the original corpus. Each class contains 30,000 training samples and 1,900 testing samples. The total number of training samples is 120,000 and testing 7,600.

The file classes.txt contains a list of classes corresponding to each label.

DBPedia Ontology

They are listed in classes.txt. From each of these 14 ontology classes, we randomly choose 40,000 training samples and 5,000 testing samples.

Therefore, the total size of the training dataset is 560,000 and testing dataset 70,000. There are 3 columns in the dataset (same for train and test splits), corresponding to class index (1 to 14), title and content.



Implementation

Layers description

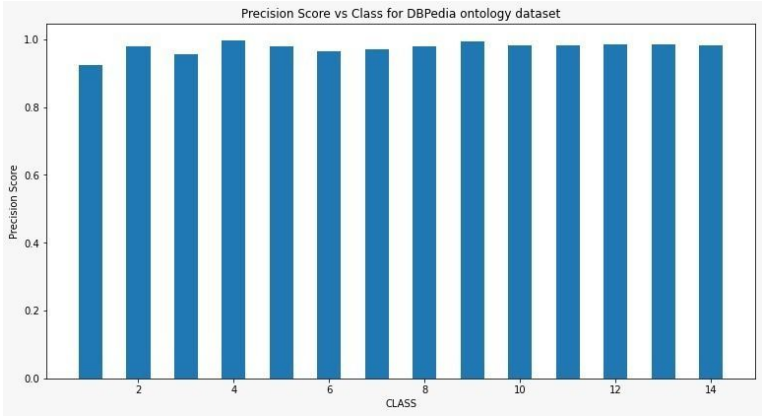
- This is the description of the structure of the neural network.
- The no of classes in the final linear layer can be varied as per the no of classes in the problem statement

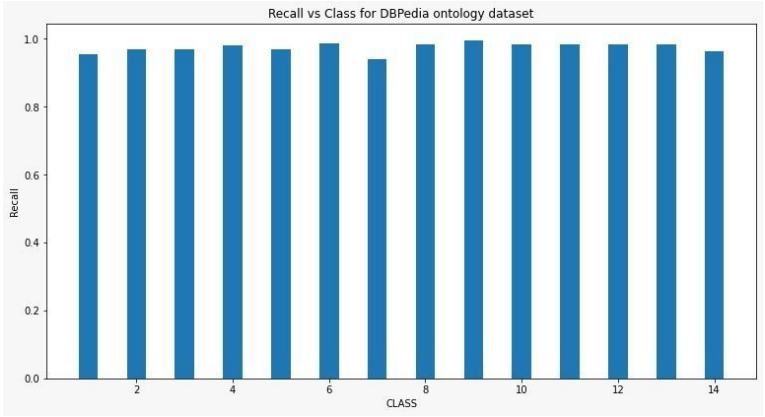
```
CharacterConvolutionNetwork(
  (conv1): Sequential(
    (0): Conv1d(68, 256, kernel_size=(7,), stride=(1,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (conv2): Sequential(
    (0): Conv1d(256, 256, kernel_size=(7,), stride=(1,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (conv3): Sequential(
    (0): Conv1d(256, 256, kernel_size=(3,), stride=(1,))
    (1): ReLU()
  )
  (conv4): Sequential(
    (0): Conv1d(256, 256, kernel_size=(3,), stride=(1,))
    (1): ReLU()
  )
  (conv5): Sequential(
    (0): Conv1d(256, 256, kernel_size=(3,), stride=(1,))
    (1): ReLU()
  )
  (conv6): Sequential(
    (0): Conv1d(256, 256, kernel_size=(3,), stride=(1,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  )
  (fc1): Sequential(
    (0): Linear(in_features=8704, out_features=1024, bias=True)
    (1): Dropout(p=0.5, inplace=False)
  )
  (fc2): Sequential(
    (0): Linear(in_features=1024, out_features=1024, bias=True)
    (1): Dropout(p=0.5, inplace=False)
  )
  (fc3): Linear(in_features=1024, out_features=4, bias=True)
)
```

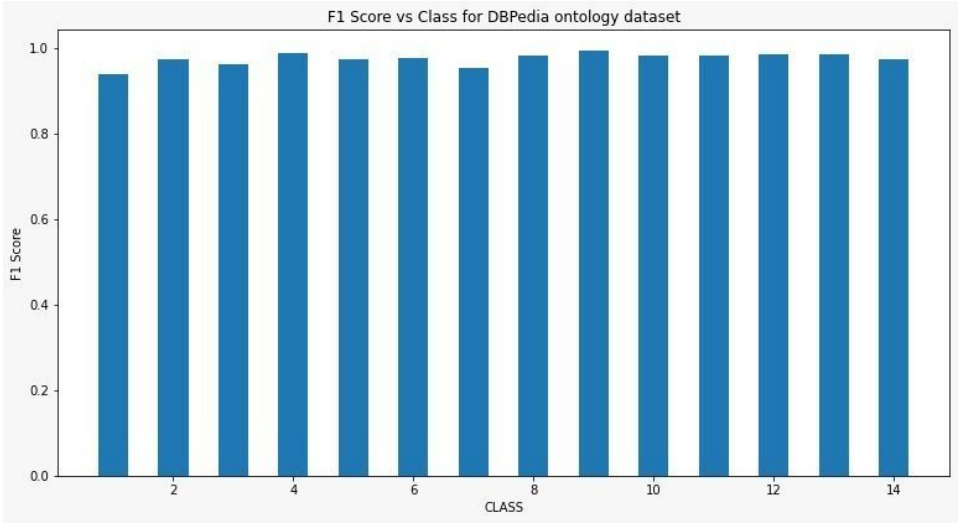
Implementation of the model is explained in detail in the report.

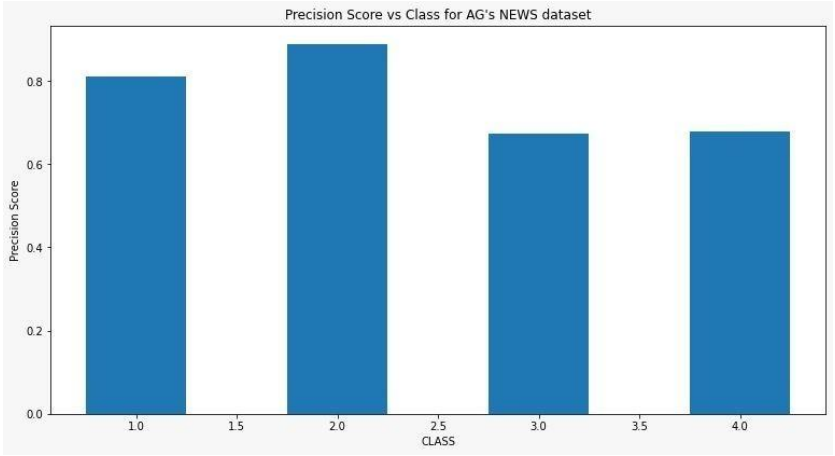
Observations

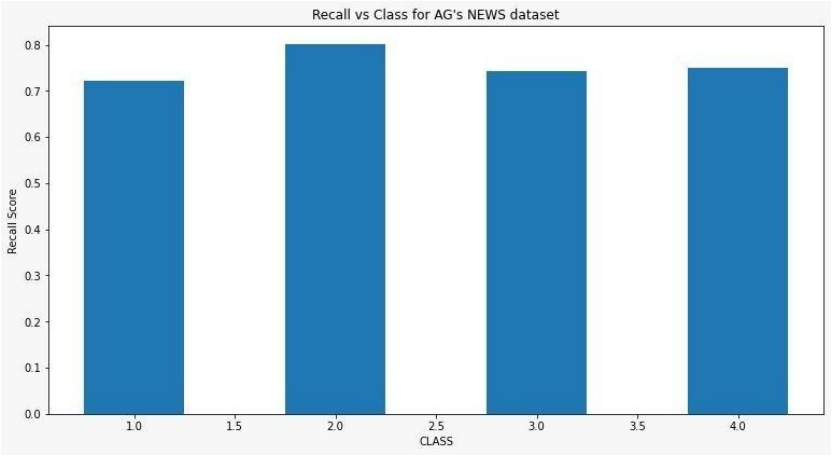
- We trained our CNN model on two different datasets for 6-8 epochs.
- Results we got had high variance and were highly dependent on the dataset.
- Results of the model are shown in the following slides in terms of Accuracy/ Precision/ Recall & F1 score.

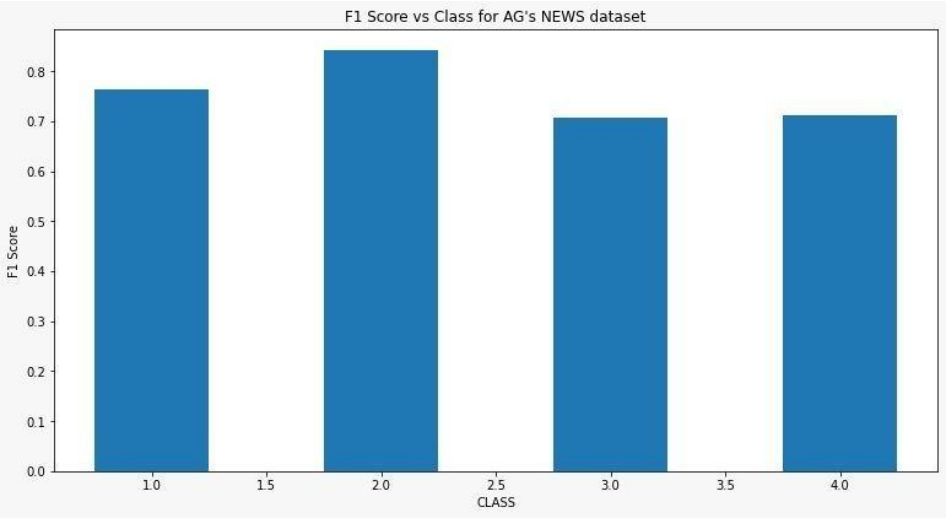

Precision Score vs Class for DBPedia ontology dataset
Precision Score
CLASS
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
12
14

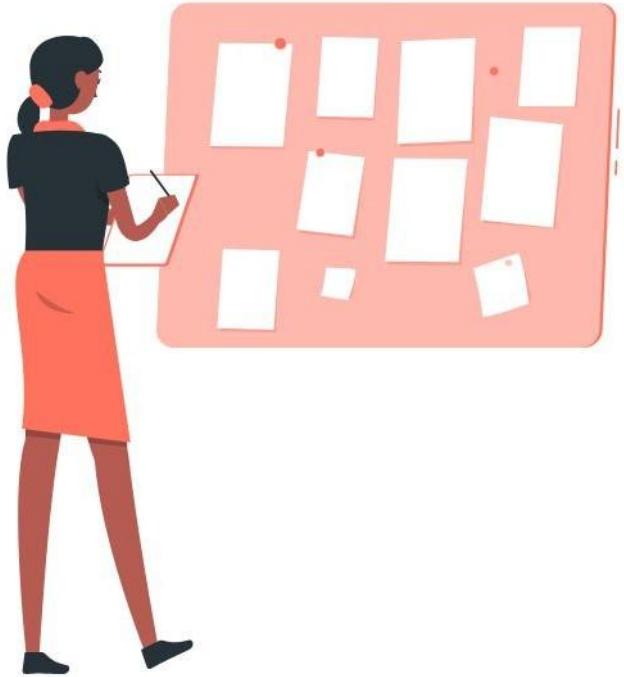

Recall vs Class for DBPedia ontology dataset
Recall
CLASS
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
12
14


F1 Score vs Class for DBPedia ontology dataset
F1 Score
CLASS
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
12
14


Precision Score vs Class for AG's NEWS dataset
Precision Score
0.0
0.2
0.4
0.6
0.8
1.0
1.5
2.0
2.5
3.0
3.5
4.0
CLASS

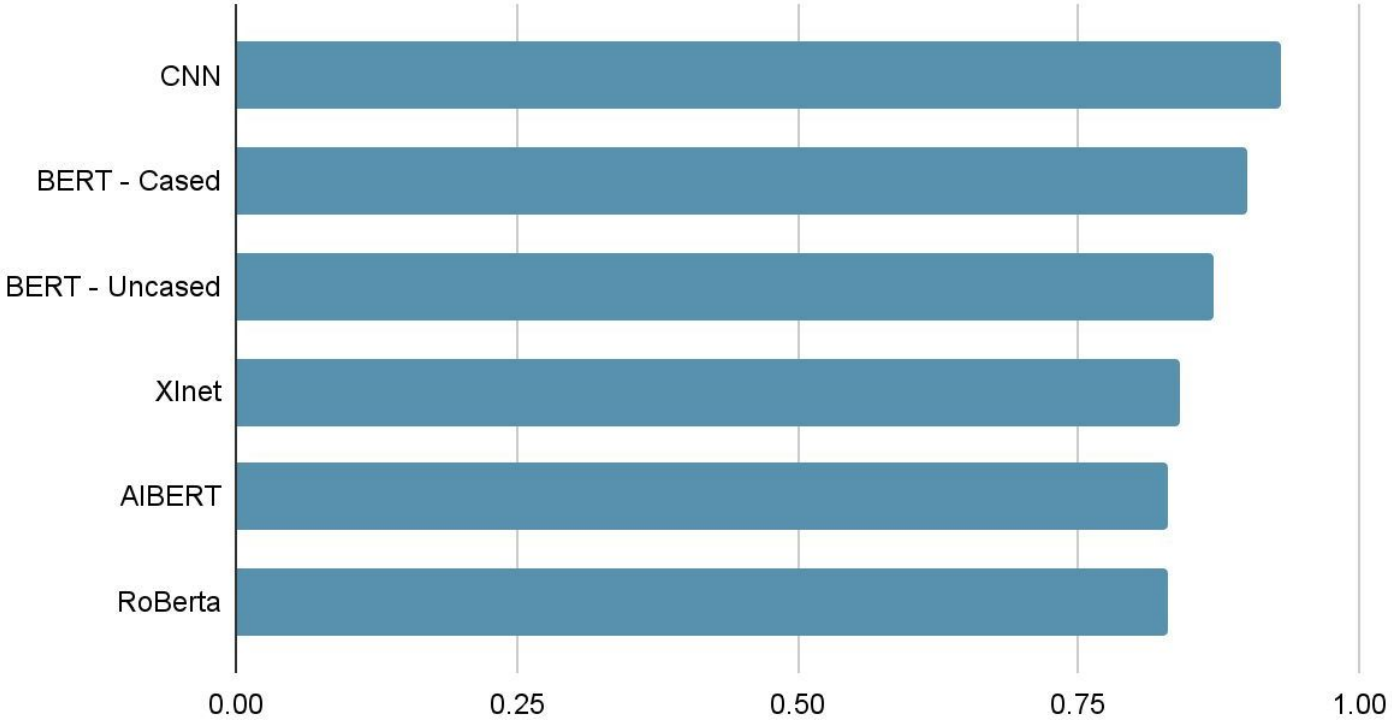

Recall vs Class for AG's NEWS dataset
Recall Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1.0
1.5
2.0
2.5
3.0
3.5
4.0
CLASS


F1 Score vs Class for AG's NEWS dataset
F1 Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1.0
1.5
2.0
2.5
3.0
3.5
4.0
CLASS



Model Comparisons

Analysis by Comparison



Conclusion

Why CNN performed better?

- **Effective method**- Character level ConvNets works for text classification without need of words. This indicates that language can be interpreted as a signal of characters.

- **Effective on user generated data**- ConvNets work well on user generated data. The degree of word curation varies in human generated data significantly.

- Our results show that ConvNets works better in real world scenarios, like identifying exotic character combinations, misspellings, emoticons and exclamatory words (like hmm and oh!).

Ending remarks!

The research paper given uses ConvNets at character level for text classification. The implementation of the algorithm and the approach used in this paper was very interesting and challenging. As the authors have mentioned, this research can be used to apply character-level ConvNets for a broader range of language processing tasks as a future work.

THANK YOU!!!!!