Document Classification with Neural Networks
Deep Learning Homework

Bálint Gyimes, Kornél Tóth, Zsolt Bartis

December 2021

1 Introduction

1.1 General overview

Text classification is a task, that all publishers face from news sites to scientific journals. Classifying these articles can be a very hard task, even for the fields' experts, because of the similarities of different categories. These large, unstructured datasets are not optimle for deterministic computer models. Our approach for this task uses the capabilities for deep learning models, which is built for such tasks.

1.2 Main goal of the project

The main role of this project is to create a neural network for document classification. As a first approach, only 5 classes are used to build networks. After that, the accuracy of classification is investigated after adding further classes to the dataset. Two ways of training are considered: based on multiple datasets, where the titles and the articles are different datasets, and based on a single dataset, after merging the title with the article. The difference of the two ways of training is investigated. [10]

2 The dataset

2.1 Source

We use the DBpedia dataset, which was obtained from the source provided in pythorch documentation.

The DBpedia ontology classification dataset is constructed by picking 14 non-overlapping classes from DBpedia 2014. They are listed in classes.txt. From each of the 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. The files train.csv and test.csv contain all the training samples as comma-sparated values. There are 3 columns in them, corresponding to class index (1 to 14), title and content. The title and content are escaped using double quotes ("), and any internal double quote is escaped by 2 double quotes (""). There are no new lines in title or content. Some of the class names are: Company, EducationalInstitution, Artist, Athlete

2.2 Data preparation

After importing the datasets from the csv files into pandas Dataframes, we select the required amount of categories and separate the labels and the texts and then the titles from the descriptions. Then the separate texts are processed using the sklearn CountVectorizer and a stopword list from nltk. This results in vectors containing the tokens and sparse matrices of token counts. The matrices are further processed using a custom normalization function. Finally the validation datasets are obtained from the test data, labels are converted to categorical type and the title and description matrices are concatenated if needed.

2.3 Data generator

Due to the size of the dataset we did not have to opportunity to load the training set directly to the RAM, so we had to implement a data generator. The first idea was the ImageDataGenerator [2] implemented by the keras library, but because of the special shape of our dataset, we decided to implement our own generator. For this, we have implemented a class deriving from the keras.utils.Sequence object [9]. This gave us the opportunity to customize the generator, while having the multiprocessing capabilities of the Sequence object.

3 The model

During the research for the ideal model design, we came accross two of the most used types for this kind of task. CNN (Convolutional Neural Network) [4, 11, 5] and MLP (Multi Layer Perceptron) [7, 3]. By focusing on the shape and size of the preprocessed data we have already had, we came to the conclusion that adding convolutional layers would not really affect the accuracy that we can get out of the preprocessed data, so we chose an MLP model based on this paper [1].

3.1 Hyperparameter tuning

After we added the desired layers, we added a range for the hyperparameters based on this paper [3]. For tuning itself, we implemented a Hyperband object, from the kerastuner library [8]. One of the selling points of this library was the compatibility with Keras, and the distribution strategy property, which gave us access to the parallelism of GPUs. For the sake of comparison we used the same hyperparameter values for all model variation. The logs are available in our git repository.

3.2 MLP

As it was mentioned in the beginning of this chapter, we decided to use a MLP model. For the base of the implementation we used a Sequential object from Keras. This model remained the same during our tests, changing only the input shape of the first, and the output shape of the last layer of our model.

Our model variations (six in total) includes the combinations of input sets based on titles, descriptions and there concatenations and outputs for five and ten class classification.

4 Results

4.1 Comparison for five class classification

For our three tests, we concluded that accuracy increases based on the size of the input data in a linear relationship, until it reaches a threshold. Our model with the biggest input size reached 97.8% accuracy with a few outliers, which helped us find the similar categories: Categories *Company* and *EducationalInstitution* and categories *Artist*, *Athlete* and *OfficeHolder* had the better chance to be mismatched with each other.

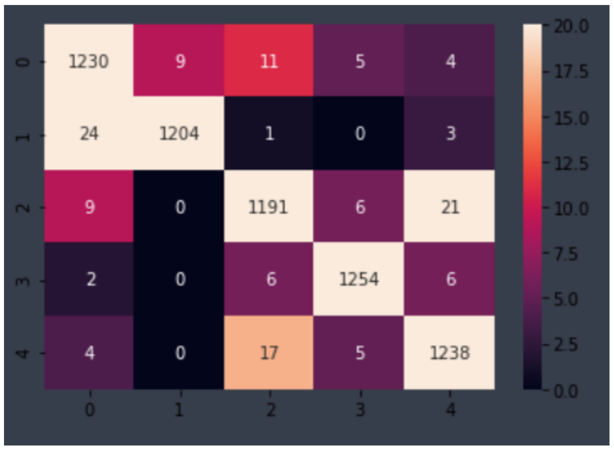
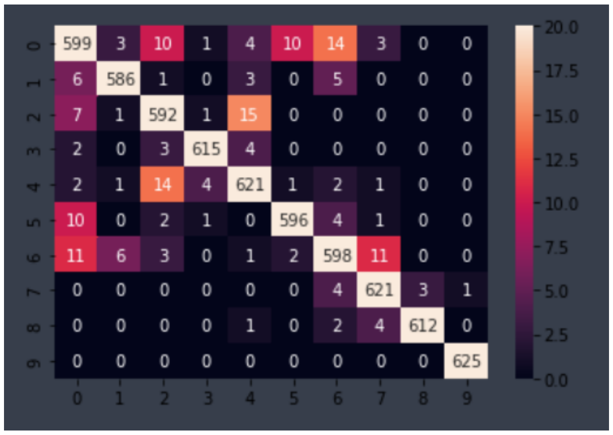


Figure 1: Heatmap for 5 classes

4.2 Comparison for ten class classification

Suprisingly the model with the same hyperparameter tuning provided similar results as the classification of 5 classes, using the same amount of data. The best accuracy, with the largest input shape was 97.0% having a few, but well defined separable outliers: classes *Artist*, *Athlete* and *OfiiceHolder*, classes *Building* and *NaturalPlace*, and classes *NaturalPlace* and *Village* respectively had the best chances to be mismatched with each other.



	0	1	2	3	4	5	6	7	8	9
0	599	3	10	1	4	10	14	3	0	0
1	6	586	1	0	3	0	5	0	0	0
2	7	1	592	1	15	0	0	0	0	0
3	2	0	3	615	4	0	0	0	0	0
4	2	1	14	4	621	1	2	1	0	0
5	10	0	2	1	0	596	4	1	0	0
6	11	6	3	0	1	2	598	11	0	0
7	0	0	0	0	0	0	4	621	3	1
8	0	0	0	0	1	0	2	4	612	0
9	0	0	0	0	0	0	0	0	0	625

Figure 2: Heatmap for 10 classes

5 Summary, further plans

We preprocessed text data using count vectorization and eliminating extreme outliers, built an MLP model using Keras, optimized the hyperparameters, and reached 97%+ accuracy on both five and ten class classification. During the evaluation process we provided insights on the change of accuracy based on the size of the input and the number of classes and found the classes which showed the most similarities to each other. Our further plans include finding a preprocessing method where we can use the capabilities of convolutional layers to our advantage [6], building a CNN model and reaching for higher classification accuracy.

References

[1] Fayçal Benrekia, Mokhtar Attari, and Mounir Bouhedda. Gas sensors characterization and multilayer perceptron (mlp) hardware implementation for gas identification using a field programmable gate array (fpga). *Sensors*, 13(3):2967–2985, 2013.

[2] Francois Chollet. Building powerful image classification models using very little data. *Keras Blog*, 5, 2016.

[3] Lukas Galke and Ansgar Scherp. Forget me not: A gentle reminder to mind the simple multi-layer perceptron baseline for text classification. *arXiv preprint arXiv:2109.03777*, 2021.

[4] Brownlee Jason. Best practices for text classification with deep learning. 2017.

[5] Rie Johnson and Tong Zhang. Convolutional neural networks for text categorization: Shallow word-level vs. deep character-level. *arXiv preprint arXiv:1609.00718*, 2016.

[6] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016.

[7] Leonardo Noriega. Multilayer perceptron tutorial. *School of Computing. Staffordshire University*, 2005.

[8] Mohamad Zaim Awang Pon and Krishna Prakash KK. Hyperparameter tuning of deep learning models in keras. *Sparklinglight Transactions on Artificial Intelligence and Quantum Computing*, 1(1):36–40, 2021.

[9] Venkatesh Satvik. Data generators with keras and tensorflow on google colab. 2020.

[10] Jocelyn Sietsma and Robert JF Dow. Creating artificial neural networks that generalize. *Neural networks*, 4(1):67–79, 1991.

[11] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657, 2015.