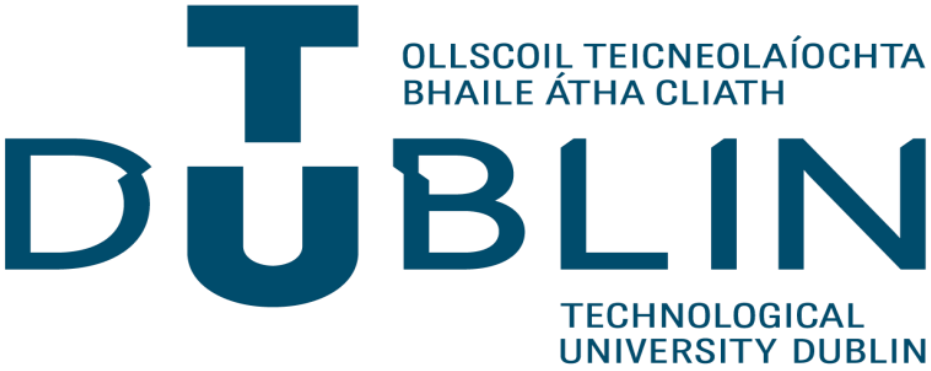

Student name:	Vyoma Mohan
Student number:	D22124454
Subject:	SPEC 9993
Stream:	MSc Computer Science (Data Science)
Assignment:	Deep Learning assignment
Submission type:	Solo

"I confirm that the document and related code here is all my own work and that I did not engage in unfair practice in any way"

Drive links:

Google drive links to code zip file and chosen saved models from the project

Link to zip file	https://drive.google.com/file/d/1kRDDSJX2oAXvicnOV36PM08racp9jz50/view?usp=share_link
Link to part 1 model 4	https://drive.google.com/drive/folders/1Q5wdXwa4PVQhG1dJc8gfr-i2Nyv_adse?usp=sharing
Link to part 1 model 6	https://drive.google.com/drive/folders/1jqnW2a0nsA2Bju7x9Nc5g2b_g6BMcRpX?usp=share_link
Link to part 1 model 7	https://drive.google.com/drive/folders/1jIf5UHSYrXjmEQmxrNljIf75l1bQFI80?usp=share_link
Link to part 2 model 1	https://drive.google.com/drive/folders/1lueThJhBft696XxjyAvFGIqNMVPDPY5a?usp=share_link
Link to part 2 model 5	https://drive.google.com/drive/folders/1hsBaCD4OjVi4250y7g7VZvkVfyLhVNFi?usp=share_link
Link to part 3 model	https://drive.google.com/drive/folders/1-I4cawHms40KFfKze3vnliP6T475t8Xb?usp=share_link

Part 1: News article prediction task

Overview:

The given task was to predict the section name of news articles from the guardian dataset. We were to mostly use the body of the article in order to predict the section name for the task.

Preprocessing steps:

A) Filtering the data to be used:

Here, we take a count of the top 20 topics present in the dataset and then filter only the records belonging to those topics. The reasoning for this step is given below:

- If a record belongs to a section which has only 1 record belonging to that section, there is a good chance that there was a typo in the section name or some other issues
- If there are less number of records belonging to the section, the model might not be able to learn the details anyway.
- If we know the number of sections we are taking, we can consistently set the softmax to 20
- This method ensures that we will spend less time cleaning. No need to check for empty section names or consolidate typos.

Furthermore, the demo notebook needs to be run separately. If we know the distribution, generating one-hot encoding separately might become easier/make more sense.

B) Check encoding

Some characters in the dataset had different letters (like a-hat). So to remove those, we encode into utf-8 and decode again. This is to ensure that the model is able to handle the data.

C) Take only topic sentences

There is a theory that the first sentence is the most important sentence in a paragraph. [1] In rare instances it will be the last line or something in the middle. But to make processing easier, we will take only the first sentence in the paragraphs. This will reduce the fluff sentences that might confuse the model and also speed up processing.

D) Remove punctuation

This is also a question of convenience. Since we are not generating sentences, it would be good to remove punctuation. The idea behind it is that keywords will have the most weight on a topic.

Sampling:

For the sample proportion, we are using 15% of the dataset in final presentation. (This was varied during tuning process. Mostly the models were tuned using 10% of the data)

Splitting the data into train and test:

For the final presentation of the model, the data is first split into 90% training and 10% test. Out of the 90% training, 10% is used as validation during model fitting.

Even though this is the case in the final presentation, in tuning, the validation set was varied a bit.

Running the trained models:

The models were run on the sample/proportion of data mentioned and the results are recorded in order to draw comparisons.

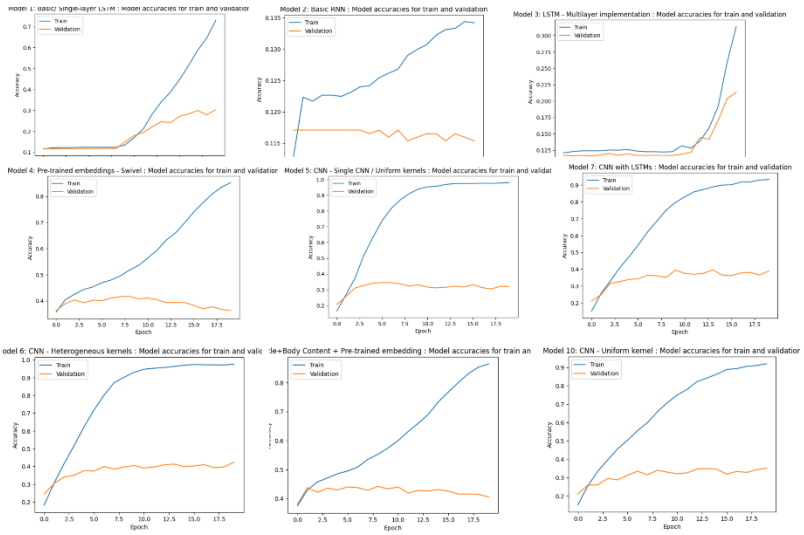

Fig 1: Collage of performances (Model 8 Gnb does not have epoch graph so its not there)

Model Number	Last training accuracy	Last validation accuracy	Test accuracy
1 – Basic/Single layer LSTM	0.7303	0.3027	0.3089
2 – Basic RNN	0.1342	0.1154	0.1295
3 – Multilayer LSTM	0.3139	0.2130	0.2149
4 – Pretrained swivel	0.8527	0.3621	0.3814
5 – Single CNN	0.9807	0.3198	0.3429
6 – CNN Heterogeneous kernels	0.9753	0.4238	0.4138
7 – CNN with LSTMs	0.9335	0.3889	0.3933
8 – Gaussian NB	-	-	0.3362
9 – Webtitle + content	0.8641	0.4055	0.4100
10 – CNN uniform kernels multiple	0.9199	0.3518	0.3701

From the above table, we can make the following notes for comparisons:

Models	Comparison statements
LSTM vs Basic RNN	LSTM outdid the basic RNN structure. Even inside more complicated placements, LSTMs seemed to do better. Thus while training, tended to try them more.
Single layer LSTM vs Multilayer	Surprisingly, the single layer did better than multiple layers. Also having many LSTM layers increases the training time.
On the fly embedding vs pretrained embedding	For pre-trained embedding, used the swivel embedding discussed in class. The performance was very good and it took significantly less training time than custom implementations.

	However, did not prefer using it as I was not able to figure out how to use time distributed layers (including LSTM and RNN) after the pre-trained embedding. One of the dimensions was lost after pre-training.
CNN single vs CNN multiple	Single CNN did better on training data whereas multiple CNN layers uniform had better results on test and validation data.
CNN uniform vs heterogeneous kernel	Heterogeneous kernel did significantly better. It is one of the models chosen for the demo notebook.
CNN vs CNN + LSTM	The combined model of LSTM + CNN does not outperform the heterogenous kernel model, but is better than single CNN or uniform kernels
Non – neural network	Chose gaussian Naïve bayes since it is probability based and I assumed it would be easier to train. Also Gnb does not have too much hyperparameter tuning. It did better than the Basic RNN and multilayer LSTM implementations on the test data.

Fig: Evaluation accuracies of Part 1 models

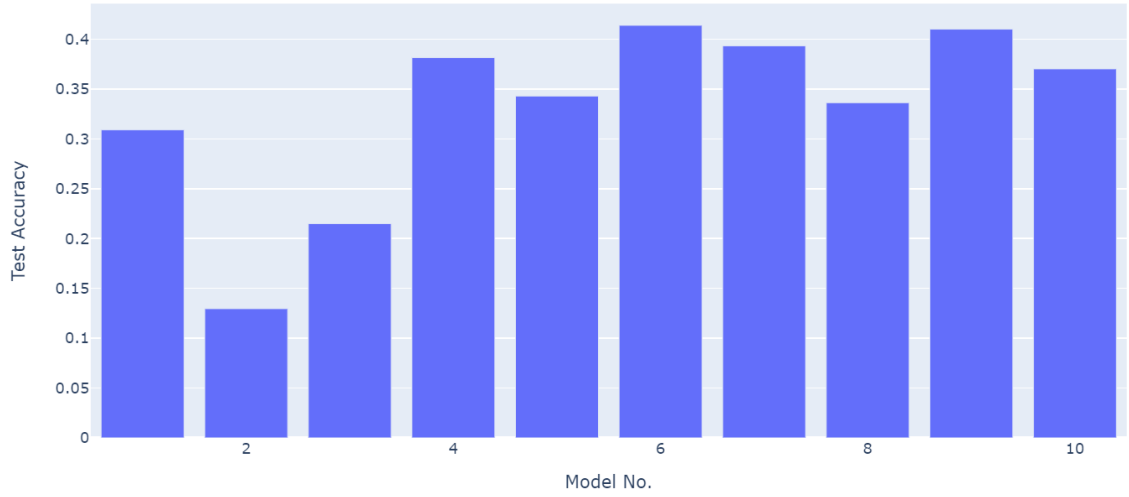

Fig 2: Test data accuracies for the models

Tuning:

The full tuning notes are not included as part of the report. However, a small sample of the initial stages is included in the end before the references. The same procedure of maximizing the layers that increased accuracy and removing layers when the accuracy goes down is followed throughout.

Saving the models and test data:

Model 6 and model 7 are chosen to be the models in the demo notebook. This is because of their good performance and also because they are custom models.

Model 4 is also included because it has pretrained embeddings.

The test data is also saved in an excel file so that it can be used in the demo notebook for the models.

Part 2: Transfer Learning

Overview:

In this section, we were asked to predict the category of news articles from a different dataset (BBC) with the models trained in the previous section. In addition to transfer learning, we also compare these models to models created from scratch using only the new dataset.

Preprocessing:

A) Check the categories

First, we check how many categories are there in the dataset. The BBC dataset has only 5 categories. So, no need to filter. Also, now we already have a fixed number for the final softmax layer.

B) Check encoding

This is done for the same reason as in the previous section. We encode into utf-8 and decode again.

C) Remove punctuation

Since we are not making sentences, we can remove the punctuation without any repercussions.

Splitting and proportion used:

The BBC dataset is small compared to the previous one. So, I opted to use the whole dataset.

Since a proportion for test and validation is not mentioned for this section, we take 20% of data for testing.

In the remaining 80%, 5% is validation.

Freezing procedure for transfer models:

Wherever we are using transfer models here, I chose to re-train the last layer as well as the embeddings layers for the transfer models. This is because I felt that the embeddings should try to accommodate the words in the new texts, and also I wanted to give a better transition into the 5 output softmax layer. Having a 20 softmax there without at least trying to change weights felt counterintuitive. Also, it is good practice to change the head and the last bit of a transfer network.

Running the trained models:

Both the transfer learning and from scratch models are created and run and their performance is recorded:

Model Number	Last training accuracy	Last validation accuracy	Test accuracy
1 – CNN pretrained	0.9811	0.9326	0.8808
2 – CNN + LSTM pretrained	0.7954	0.8202	0.6988

3 – CNN + LSTM pretrained + adjustments	0.7907	0.5506	0.6561
4 – From scratch LSTM	0.2322	0.2022	0.2179
5 – From scratch the steps in model 3	0.2466	0.2022	0.2157

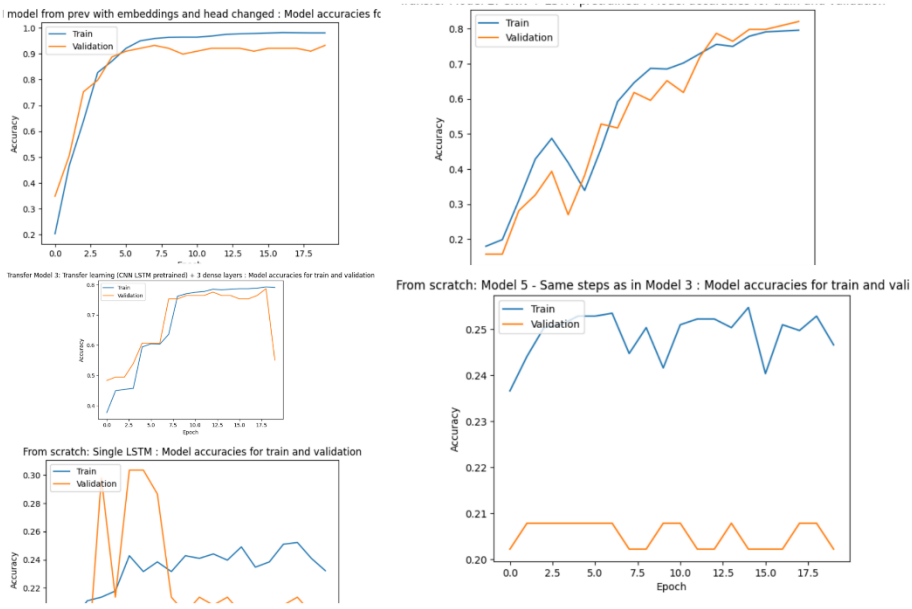

Fig 3: Performance collage of the transfer models (1 and 5 are the most important here)

Fig 4: Evaluation graph of the transfer and from scratch models

<u>Conclusions:</u>

It is not clear why the transfer models are out performing the from scratch models to such an extent. Some <u>theories</u> are:

- The transferred models were trained on a bigger, but similar dataset
- We're having marginally higher accuracies in testing for this dataset than the original because there are only 5 topics and the dataset is small.

The transfer models were also much quicker to train because most of the weights were adjusted already.

Part 3: Writing own news article

Overview:

In this task, we will use the BBC dataset to try to generate our own news article.

Thought process behind developing the model structure:

- The custom model was meant to find a simple way to generate sentences using the given corpus.
- The model uses unigrams in order to generate the word.
- We will treat the generation of the next word as a classification task.
- Given that X is a word, we will find the most probable next word Y by treating Y as a classification label.
- The downside is that we will have as many labels to train in the final layer as there are words in the corpus.
- Also, the computing requirements are very high to train even one epoch.

Variations tried:

- Some variations tried were using bigrams instead of unigrams. It still did not provide coherent sentences and just increased the training time.
- Also tried up to 10 epochs. The results are still very similar. So, the final model is trained with 5.

Procedure:

A) Filter categories

For the generation task, we will be using only 2 most popular categories. This is done in order to keep the kind of words similar, the model light and also the number of labels in Y low.

B) Sample and check encoding

40% of the resulting data is sampled for use. Also, the encodings are check as before.

C) Remove punction

Since we are generating sentences, it would have been good to retain punctuation. However, due to errors, we opt to remove them from the text.

D) Generate unigrams

To get the X and Y values, we will first generate bigrams. Then we will split it and take the first word as X and the second word as Y.

E) Constructing the model

For the model, one of the well performing models with 2 LSTM layers and 3 Dense layers (relu, sigmoid, relu) is taken. For this section, we do not tune the models much because it is computationally intensive. Also increasing the accuracy of the model in small amounts doesn't seem to change the sentence generated.

F) Trying to introduce novelty

The current approach seemed to be a bit deterministic. It keeps producing the same output for the given seed word. So, an attempt to introduce novelty by adding dropout was made. However, this didn't change the deterministic nature of the models.

Running the model:

When the model was run with a seed word (random word noted in the corpus), it generated the following result:

```
1/1 [==============================] - 0s 32ms/step
in
1/1 [==============================] - 0s 27ms/step
the
1/1 [==============================] - 0s 47ms/step
year
The generated sentence is:
Face the year in the year in the year in the year in the year in the year in the year
```

Fig 5: Result of sentence generation

Even if the word count to be generated was increased, it would loop between "the year in". The possible reason for this is that these words occur together and often. Since the model is not well trained it might loop between these three.

Running a different sentence generation model for comparison:

In addition to this model, decided to run an MLE model just to obtain a comparison between the two.

The sentence generated by the MLE model is:

```
[102]: word_list = model2.generate(40, random_seed=2)
       print("".join(word_list))

       ut asking of ists ge  inding  ecoxx togt
```

Fig 6: Result from the MLE model

This is also having a high perplexity.

Conclusions:

Thus, both the custom model and the predefined model are run for sentence generation. They both generated sentences which are not very clear.

Overall reflections:

Thus, the section identification is done with RNNs and LSTMs on the guardian data. The models generated from this section are used in order to classify the BBC data into its category. Finally, we tried to generate article text from the BBC data for two of the most popular categories.

We saved two custom models and one model with pre-trained embeddings for part 1, a transfer model and a from scratch model for part 2, and the generator model from part 3 for use in the demo notebook. These models are run on test sets and the results are reported.

Sample tuning run:

The following is a record of the tuning run for the first part of the assignment. This does not cover the whole tuning exercise. This is just to show the thought process behind the tuning.

Initial tuning runs for LSTMs:

Trial #	Model structure	Validation %	Max accuracy in 20 epochs for val set
1	LSTM -> Dense 20 tanh -> Dense 20 softmax	0	0.1519 (training)
2	LSTM -> Dense 20 relu -> Dense 20 softmax	0	0.1517 (training)
3	TD Dense 20 relu -> LSTM -> Dense 20 relu -> Dense 20 softmax	0	0.1419 (training)
4	TD Dense 400 -> LSTM -> Dense 20 relu -> Dense 20 softmax	0	0.5981 (training)
5	TD Dense 400 -> LSTM -> 3 x Dense 400 relu -> Dense 20 softmax	0	0.7421 (training)
6	3 x TD Dense 400 relu -> LSTM -> 3 x Dense 400 relu -> Dense 20 softmax	10	0.1910
7	TD Dense 400 -> LSTM -> 3 x Dense 400 relu -> Dense 20 softmax (Same as trial 5)	10	0.2039
8	TD Dense 400 -> LSTM -> 3 x Dense 400 relu -> Dense 20 softmax (Same as trial 5)	5	0.8519
9	2 x TD Dense 400 -> LSTM -> 3 x Dense 400 relu -> Dense 20 softmax	5	(interrupted by me)
10	TD Dense 400 relu -> 2 LSTM -> 3 x Dense 400 relu -> Dense 20 softmax	5	0.1714
11	TD Dense 400 relu -> LSTM -> TD Dense 400 relu -> LSTM -> TD Dense 400 relu -> Dense 20 softmax	5	0.2106
12	TD Dense 400 relu -> LSTM relu -> 3 x Dense 400 relu -> Dense 20 softmax	5	0.1282
13	TD Dense 400 relu -> LSTM -> 400 Dense relu -> 400 Dense sigmoid -> 400 Dense relu -> Dense 20 softmax	5	0.7022
14	TD Dense 400 relu -> LSTM 50 -> 3 x Dense 400 relu -> Dense 20 softmax	5	0.5209
15	TD Dense 400 sigmoid -> LSTM -> 3 x Dense 400 relu -> Dense 20 softmax	5	0.1219

Training data for LSTM:

TD – Time Distributed

If not mentioned, LSTM had 16 units in them.

Some fine tuning adjustments tested on random models:

Trial #	Model structure	Validation %	Max accuracy in 20 epochs for val set
1	TD Dense 400 -> LSTM -> Dense relu 400 -> Dense sigmoid 400 -> Dense relu 400 -> 20 softmax With L1 regularization for 3 dense layers	5	0.1202
2	TD Dense 400 -> LSTM -> Dense relu 400 -> Dense sigmoid 400 -> Dense relu 400 -> 20 softmax With L2 regularization for 3 Dense layers	5	0.1202
3	TD Dense 400 -> LSTM -> 400 Dense relu -> 400 Dense sigmoid -> 400 Dense relu -> 400 Dense sigmoid -> 400 Dense relu	5	0.1202

Did try tuning for some options like regularization, but they seemed to make the results worse, so decided not to.

RNN Trials:

Trial #	Model structure	Validation %	Max accuracy in 20 epochs for val set
1	TD 400 D relu -> RNN -> 3 x 400 D relu -> 20 softmax	5	0.1393
2	TD 400 D relu -> 3 RNN -> 3 x 400 Dense relu -> 20 softmax	5	0.1399
3	TD 400 D relu -> RNN variance scaling -> 3 x 400 dense (relu, sigmoid, relu)	5	0.1372

Multilayer LSTM trials:

Trial #	Model structure	Validation %	Max accuracy in 20 epochs for val set
1	TD 400 D relu -> 3 LSTM -> 3 x 400 Dense relu -> 20 softmax	5	0.1223
2	TD 400 D relu -> 3 LSTM -> 400 D relu -> 20 softmax	5	0.3955

With embeddings:

Trial #	Model structure	Validation %	Max accuracy in 20 epochs for val set
1	HL -> D 400 x 6(relu, sig, relu, relu, sig, relu) -> softmax 20	5	0.5185
2	HL -> D 400 x 3 (relu, sig, relu) -> softmax 20	5	0.5837
3	HL -> D 400 x3 (relu, relu, relu)-> 20 softmax	5	0.8340

HL – Hub layer

References:

[1] T. University, “Topic Sentence,” *www.touro.edu*. https://www.touro.edu/departments/writing-center/tutorials/topic-sentence/#:~:text=A%20topic%20sentence%20is%20the (accessed May 11, 2023).

[2] K. Team, “Keras documentation: Keras API reference,” *keras.io*. https://keras.io/api/ (accessed May 11, 2023).

[3] J. Brownlee, “How to Use the TimeDistributed Layer in Keras,” *Machine Learning Mastery*, May 16, 2017. https://machinelearningmastery.com/timedistributed-layer-for-long-short-term-memory-networks-in-python/