esade

NLP Assignment – AG News Analysis

1. First binary models

We first decided to train both SGD-based and Naïve Bayes binary classifiers for each of the labels (science, sports, world and business) without any hyperparameter tuning (neither for the model, nor for the vectorizer) and compare their performance. The test results are shown in table 1.

		Stochastic Gradient Descent			Naïve Bayes		
		Precision	Recall	Accuracy	Precision	Recall	Accuracy
Science	0	95%	96%	93%	92%	98%	92%
	1	87%	86%		91%	83%	
Sports	0	90%	99%	91%	98%	99%	98%
	1	97%	67%		97%	94%	
World	0	86%	100%	87%	94%	99%	94%
	1	98%	50%		95%	82%	
Business	0	84%	99%	85%	92%	97%	92%
	1	92%	43%		89%	76%	

Table 1: Performance metrics of binary classifiers for each of the labels

We can derive three main takeaways from this table:

- Except for science articles, for which the models perform in a quite similar way, the Naïve Bayes classifier outperforms the SGD classifier on accuracy for every other label
- The lowest metric is always the recall for the positive class, i.e. the share of articles belonging to a certain class the model is correctly labelling
- Although general model performance is already quite high, the SGD-classifiers don't perform very well at identifying World and Business articles

2. Hyperparameter tuning

Based on the findings of the previous section, we tried to tune the hyperparameter of the SGD-based classifier for the science articles and Naïve Bayes classifiers for the other labels. More specifically we tried to vary the size of our bag of words in the vectorizer and the alpha parameter (strength of regularization) in our classifiers. For alpha we tested the values 1e-03, 1e-6 and 1e-9, and we tested for bag of words sizes from 1 to 4. Results are shown in table 2.

	Best parameters		Resulting metrics			
	N-gram	alpha	Label	Precision	Recall	Accuracy
Science (SGD)	3	1e-06	0	96%	96%	94%
			1	89%	88%	
Sports (NB)	2	1e-03	0	99%	99%	98%
			1	96%	97%	
World (NB)	2	1e-03	0	96%	98%	95%
			1	93%	87%	
Business (NB)	3	1e-03	0	95%	96%	93%
			1	88%	84%	

Table 2: Best parameters resulting from the gridsearch and according performance metrics

While the accuracy of the resulting models didn't improve by a lot compared to the ones previously trained, the recall for the positive class improved a lot for all the models (up to 7 percentage points for the business classifier).

3. Setting the right thresholds

Even with finetuned models, the trade-off between a high-precision and a high-recall remains and the decision here depends on the model application. If, for instance we were labelling legal documents for a law firm or medical papers for a hospital to create internal databases, we would want a very high recall, since in these fields, missing a document of a certain label can have major consequences. In our case however, we can assume that we are building a classifier for a knowledge management division at a press company, hence the consequence of misclassification may not be as dramatic. Yet, it would be nice to have classifiers with recall above 90% (i.e. at least 9 out 10 articles from a class are labelled correctly).
In section 1 under point 1.3 in the code appendix we plotted the precision/recall curve of the classifiers with tuned hyperparameters and tried to set the decision threshold to values that achieve 90% without having big drops in precision:

- Science classifier: pushing the recall above 90% results in a precision that drops slightly below 90%.
- Sports classifier: we are already above 90% on the recall and precision and the curve indicates a clear optimum
- World classifier: by setting the decision threshold to 0.14, we achieve both a precision and a recall of 90%
- Business classifier: there is more of a trade-off than for the two previous classifiers, as for the science classifier, it is impossible to achieve both precision and recall above 90%

4. Multiclass model approach

To compare model performance of a binary classifier against a multi-class approach, we trained the same models considering all the 4 labels at once within the same dataset. However, comparing the best-performing model (an SGD classifier with trigrams and an alpha threshold of 1e-06), overall accuracy and precision seem to be consistently lower: for instance, the lowest accuracy for the binary classifiers is at 94% while the multiclass model yields an accuracy of 91%. Comparing the model with default probability threshold settings, however, we can detect a slightly higher average recall of the multiclass model, indicating higher model performance in identifying most instances of the respective positive class.

	Best parameters		**Resulting metrics**			
	N-gram	alpha	Label	Precision	Recall	Accuracy
SGD	3	1e-06	Science	89%	89%	91%
			Sports	95%	98%	
			World	91%	90%	
			Business	89%	87%	

Table 3. Best parameters resulting from the gridsearch and according metrics for the multiclass model

5. Classifier weights for the multi-class approach

In order to understand what the model learned during the training process, we computed the parameter weights of the best-performing multiclass SGD classifier (for detailed results, refer to Section 2 – 1.3.2 in the script). Surprisingly, the words with the highest weights (e.g. 'space',

'apple', 'technology') can be associated with topics in the fields of science or business – the labels our models had most issues with. This could point to some implicit selection bias in our model, for instance as many articles could be extracted from scientific journals focused on the field of astronomy. In order to refine our results, we conducted a further analysis with bigram parameter weights and subsequently inspected how the model assigns weights to the words. This resulted in a shift of weights from more subject-matter-related topics to news agency names (e.g. 'washingtonpost com' or 'reuters online') – which could explain the higher predictive power of models using n-grams because of their ability to differentiate classes.

6. Neural Network Approach

The first step necessary in the neural approach was to convert the data into an appropriate format. Thus, the text feature of the train and test data was selected and converted into a numpy array format. This transformation allowed us to then prepare the vectorizing layer of the neural network with the tensorflow function TextVectorization, this layer converts the words into numbers that can then be used in the neural network.
The next step was to create an embedding layer that will learn different features from the different words. Then, we needed to implement a layer to flatten the embedding layer information before being served to a dense neural network, thus, the Flatten layer.
We had three parameters specific to the word vectorization and embedding: number of dimensions of the embedding layers, the max number of words considered by the algorithm (most common words) and length of the sequence. The length of the sequence was set arbitrarily to 250, while the two other parameters were fixed after analysis and several trials. Thus, from the 50k words present in the dataset, we trained our models with 35k words and chose 100 dimensions for the embedding layer in the binary case and 150 for the multiclass case, respectively.
Concerning the performances of the neural networks, all the models had similar performances to the previous classifiers except for the multiclass model that proved to have slightly lower performance.

a. Binary-Classifier Model

After creating some basic models with two layers of dense neural networks (32 neurons and 16 respectively), we came to the conclusion that we were suffering from one main issue, overfitting of the neural networks on the training data leading with a performance of 98% on the training set while only 88% on the test set.
Therefore, we tried several techniques to reduce the overfitting:

- Simplification of the model
  - Reduction of the number of neurons and layers
- Regularization:
  - Implementing a L2 penalty (L2 regularization)
  - Dropout with different rates
  - Batch normalization

After several trials, we decided to use the simplified model in the Jupyter Notebook report.

b. Multiclass-Classifier Model

One of the first task was to convert the labels of the dataset into an appropriate format for the Softmax layer. In the multiclass model, we faced similar overfitting issue that we solved with the same techniques as before.