COMP30027 Assignment 2 - Report

Anonymous

21 May 2021

This task involved classifying recipes according to their duration, based on the recipe name, ingredients, steps, number of steps, and number of ingredients. Each recipe originated from one of three classes; `short`, `medium` or `long`, which were encoded as `1.0`, `2.0` and `3.0` respectively.

The dataset is provided courtesy of *Shuyang Li et. al.* [2], who scraped the recipe data from `food.com`. 40,000 labelled training instances were provided, and accuracies presented in this report were generated from the public Kaggle competition, for which 10,000 predictions were submitted and 3,000 were assessed for accuracy.

1 Classifiers

1.1 Support Vector Classifier

A Support Vector Classifier (SVC) using a Radial Basis Function (RBF) was implemented with *Scikit-learn* [4]. The RBF kernel was used since it is relatively simple to tune the hyperparameters of the model and achieve good performance. After tuning, an ultimate value of $C = 1$ was reached, and the performance of the SVC with the RBF was superior to linear and polynomial kernels.

This classifier used the provided `doc2vec100` datasets, which were concatenated, and the extra numeric features were standardised, then added as additional attributes to produce a 203-dimensional dataset.

1.2 Gaussian Naïve Bayes Ensemble

As an alternative and experimental model, an ensemble was composed out of 5 unique Gaussian Naïve Bayes (GNB) [4] classifiers. For the training data of these classifiers the entire training set was partitioned by its 5 columns, each partition containing a single attribute of the original dataset. This was motivated out curiosity for what would happen if an ensemble model was generated by partitioning over the features instead of instances, as this was found rarely in literature. So that GNB classifiers would be compatible with each of these datasets the count vectorized data was used for each of the textual features, and the numeric features were taken directly.

After training they were combined into an ensemble model through simple voting, which was implemented manually due to the assumption made by *Scikit-learn's* `VotingClassifier` that all models are trained on identical training data [1].

1.3 Convolutional Neural Network

The final model generated was a sequential Convolutional Neural Network, implemented with *Keras* [1] and *TensorFlow* [3].

As input, this model takes a string containing all columns of the training data present in the CSV. The first layer of the model then vectorizes this input into a length 300 tensor of integers, then passes the numeric vector to an embedding layer, which produces a 128-dimensional representation of each word. Groups of 5 of these embeddings are passed into a convolutional layer, then max-pooling is applied, and finally a dense layer is used to generate the final classifications. A diagram of this neural network architecture is included as Figure 1.

2 Evaluation

2.1 Support Vector Classifier

In training, this classifier used the provided `doc2vec100` dataset since this performed better than the `doc2vec50` instances. This is because SVCs do not suffer from the curse of dimensionality, so it is likely that in the 100 dimensional space the SVC has an easier time generating a decision boundary due to the increased amount of attributes which it could use to differentiate instances.

[1]MLxtend's `EnsembleVoteClassifier` also supports this vertical feature partitioning `http://rasbt.github.io/mlxtend/`

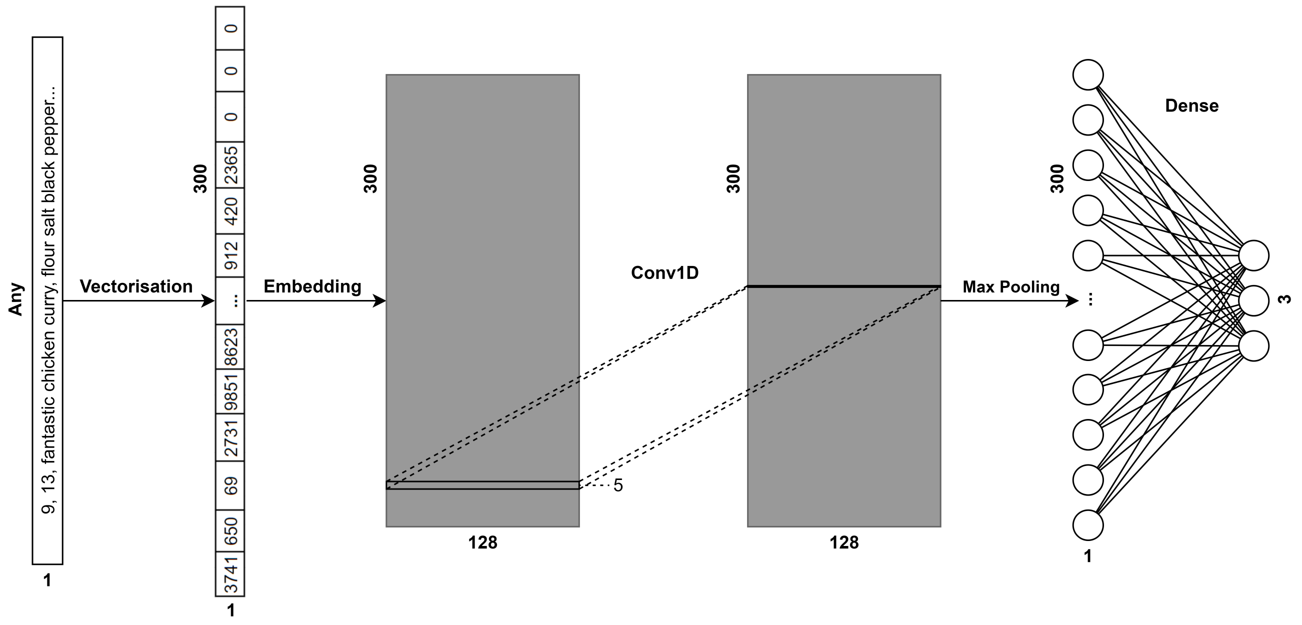


Figure 1: The Architecture used in the Convolutional Neural Network Classifier.

This model performed with 72.37% accuracy on the public Kaggle competition, outperforming a SVC trained on identical data but using a linear kernel by 0.87%. The precision and recall statistics for the model are provided in Tables 2 & 3.

2.2 Gaussian Naïve Bayes Ensemble

Tested against an isolated split of the training data, each of the 5 GNB models which comprise the ensemble scored as shown in Table 1.

Feature	Data Type	Accuracy
names	`doc2vec`	0.4197
steps	`doc2vec`	0.4033
ingredients	`doc2vec`	0.1893
number of steps	`int`	0.5948
number of ingredients	`int`	0.6026

Table 1: Accuracies of the Individual GNB Models

The reasoning for these scores is discussed further in section 3.

When combined with equal weight voting, the model achieves an accuracy of 56.62%. But since it would therefore be more accurate to simply use either one of the integer models, weighted voting was implemented.

Since the accuracies of the models were approximately at a ratio of `2:2:1:3:3` these were the first weights implemented in the voting system. After testing a variety of alternatives this ultimately provided the highest level of ensemble accuracy, slightly better than any of the individual models at 62.33%.

2.3 Convolutional Neural Network

The CNN had the best performance out of all the classifiers. It achieved an accuracy of 81.83% on the public subset of the test data. This is an improvement of 9.46% over the SVC, and is 19.50% better than the GNB ensemble.

2.4 Precision and Recall

Generally these statistics are as could be expected, however it is somewhat surprising that both the precision and recall of the GNB classifier on class `3.0` are so poor. This indicates that it was not only predicting class `3.0` for many instances of other classes, but also that it was missing the true class `3.0` instances most of the time.

Classifier	`1.0`	`2.0`	`3.0`
SVC	70.0%	74.2%	35.5%
GNB ensemble	79.4%	49.2%	43.5%
CNN	80.7%	83.7%	68.3%

Table 2: Precision for Each of the Classifiers and Classes

Classifier	`1.0`	`2.0`	`3.0`
SVC	69.4%	71.4%	65.4%
GNB ensemble	61.2%	78.2%	21.0%
CNN	81.5%	82.6%	72.0%

Table 3: Recall for Each of the Classifiers and Classes

3 Discussion and Error Analysis

Gaussian Naïve Bayes makes the assumption that features are independent, and that instances are normally distributed within each class. To see how these assumptions are violated, consider the text based attributes, the names, steps, and ingredients. By projecting the 100-dimensional text vectorisation space down into 2 dimensions it is possible to visualise the distribution of a sample of this data [5]. Such a visualisation is shown in Figure 2, where it is clear that the vectorized words from the recipe dataset form an annulus shape.

From this distribution it is clear that the instances violate the assumption of normality within each class. It is also clear that the independence assumption is violated, as there is a clear correlation between these points.

The assumption of normality is more reasonable for the numeric attributes, which is why these see significantly better performance and are represented more favourably in the weighted ensemble model, however strong correlation can be observed between the number of ingredients and steps. All of these violations contribute to the poor performance of the GNB ensemble.

The **Support Vector Classifier** performed significantly better at the classification task. As mentioned in section 1.1 this is due to it's ability to work effectively with high-dimensional data. After experimentation with a range of values for C, the regularisation parameter, a value of 1 was selected. This moderate value is reasonable considering the high dimension of the feature space, since in the high dimensional space the model finds it easier to create boundaries between classes, and therefore does not require the softer margins afforded by smaller values of C.

Finally, the **Convolutional Neural Network** which performed significantly better than both aforementioned models. It is believed that the CNN classifier is able to achieve this performance due to its use of both embedding and convolutional layers.

Although the single convolutional layer in the model is only capable of learning linear decision boundaries on its input, it is important to realise that the input to the convolutional layer is not a direct vectorisation of the text data. Instead, the vectorized text is passed through an embedding layer, which transforms the data into a 128-dimensional embedding space. This introduces non-linearity into the model, and since the embedding is designed to place related words nearby in the embedding space, the single convolutional layer is able to form linear boundaries in this space which differentiate classes effectively.

When training the CNN, out of the 40,000 training instances, 10% were reserved for testing, and of the remaining 36,000 20% (7,200) were reserved for validation. The other 28,800 were used to train the model. The training duration was determined by the amount of epochs required before the validation loss begun to increase, which resulted in a training time of 3 epochs. A visualisation of the training and validation accuracies and losses which shows this appears in Figure 3.

References

[1] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[2] Bodhisattwa Prasad Majumder et al. 'Generating personalized recipes from historical user preferences'. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019.

[3] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[4] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[5] Radim Řehůřek and Petr Sojka. 'Software Framework for Topic Modelling with Large Corpora'. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. `http://is.muni.cz/publication/884893/en`. Valletta, Malta: ELRA, May 2010, pp. 45–50.

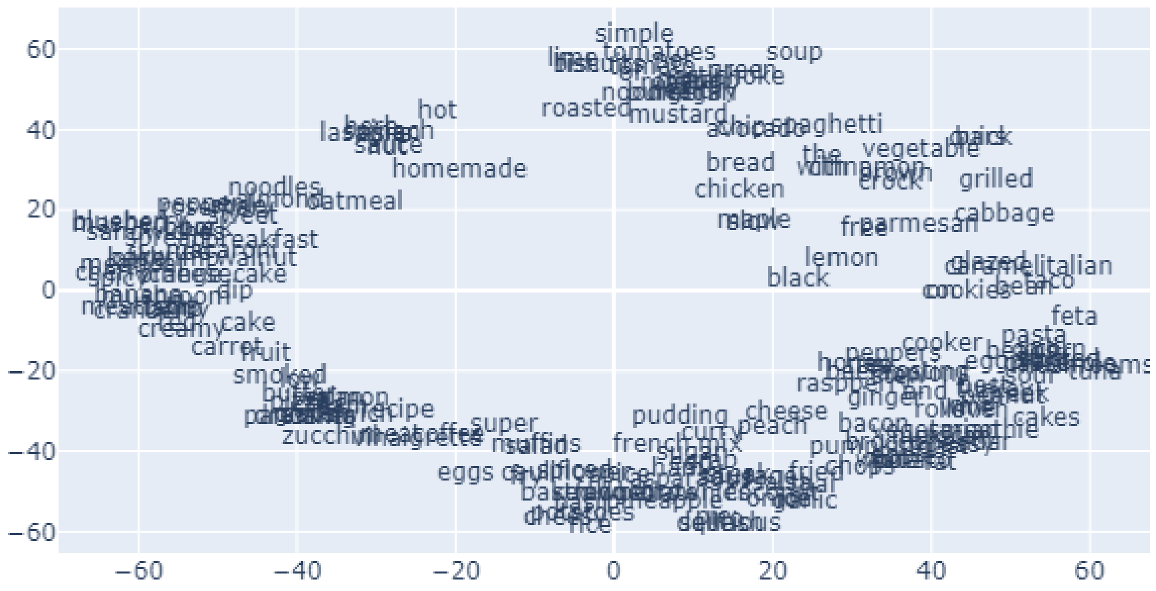
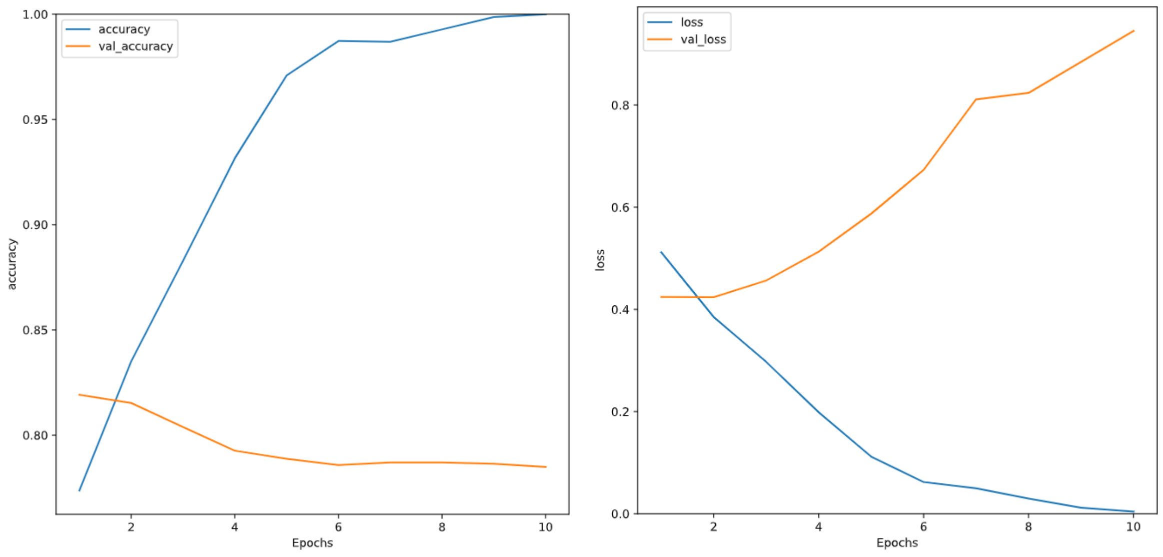


Figure 2: A 2D Projection of the Distribution of 200 Words in the 100-dimensional doc2vec Space.



(a) Accuracies

(b) Losses

Figure 3: CNN Training and Validation Statistics