Completed work

My initial aim for this project was to build a system to help my work colleagues access technical support information. After considering an application based on a search engine, I hit upon the idea of constructing an email 'conversational agent' that could automatically process technical support requests. The system underpinning this would be an 'Intention Classifier' – a machine learning model designed to classify incoming emails according to various categories of user intent.

I first had to consider which technologies were most appropriate to the task, which comprised a variety of technologies including rule-following systems, shallow learning methods like K-Nearest Neighbours (KNN), and more sophisticated models based on Neural Networks such as Recurrent Neural Networks (RNN), Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU). For such models to add value to in the proposed business context, they needed to deliver high quality results, require little maintenance and have a low total cost of ownership. By this definition, a rules-based approach was instantly dismissed due to its lengthy set-up time and maintenance costs combined with limited results. All that remained was to discover which of the other methods produced better results. Neural Networks seemed more promising by virtue of their reputation, but if a simpler and more maintainable KNN classifier could deliver similar quality then this could also be a desirable option. To answer this question, I had to build and evaluate a number of models adequate to the studied problem.

Before I could begin working on any classification models, I first had to derive a data set. This involved a labour-intensive process of cleaning, formatting, anonymising and classifying 698 email records. The data was first cleaned by eliminating certain examples that should never be responded too – for instance emails from outside the company's domain name, or from within the support team itself. I then formatted the data by combining the subject and body of the email, removing email signatures, and anything but the latest message in the thread. Anonymisation was trickier, involving removal of any personally identifiable or contractual information. I was able to retain meaning by generalising with placeholders, for instance <NAME>, <PRODUCT> or <COMPANY>.

Up to this point, I was able to use automated processes and data from the email export itself to do most of the processing automatically. This was a positive result not just for my labour time, but also because the process must be perfectly repeatable if the system is to be meaningfully implemented. Additionally, generalising the data will improve results as the appearance of unique or sparse names would be difficult to train on.

Building models themselves has proved comparatively straightforward. I have been building KNN and Neural Network models with the Sci-Kit Learn[1] and Keras[2] libraries respectfully, using the Python[3] programming language and the Visual Studio Code IDE[4].

My original schedule of work entailed completing the data processing stage in August. This did run into September; however, it did not take me as long as anticipated to construct models and at the time of submission of this assignment I am now back on schedule. No human subjects were or will be involved in the production of this project, and there is no requirement for Ethical Approval.

Word count: 594

[1] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[2] Keras.io. (2015). Keras: The Python Deep Learning library. [online] Available at: https://keras.io [Accessed 30 Sep. 2019].
[3] Van Rossum, G. & Drake Jr, F.L., 1995. Python tutorial, Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.
[4] visualstudio.com (2015). Visual Studio Code. [Online] Available at: https://code.visualstudio.com/download [Accessed 30 Sep. 2019].

Work to be done

Currently I am optimising the hyperparameters of my models to deliver better results. I am intending to use cross validation[5] to evaluate the models, as my data set is too small to meaningfully break into training and test sets. To this end, I am considering how best to score predictions. This score can then be fed into the cross-validation procedure in order to prioritise the factors I care about most. A multi-class confusion matrix[6] seems ideal to analyse how performance varies across classifications, however, in order to use this for scoring, it will need to be aggregated into a single number.

Three common approaches to this include 'Precision', 'Recall' and 'Accuracy'[7], which involve different ways of calculating a score for each class based on four categories: True Positive, False Positive, True Negative and False Negative. In this instance, a False Positive is much more costly than a False Negative as it could cause confusion and prompt further action that may introduce errors into production data - requiring extra work to fix and follow up with a correct response. Meanwhile, a false negative merely requires the support team to action the email correctly. Therefore, I am intending to score my models on the mean precision of all actionable classes (not all classes require a follow up action or response).

Scoring prediction results with cross validation in this way will enable objective optimisation of the hyperparameters. For instance, the hyperparameter *k* of a KNN model can be optimised by iteratively building KNN models with increasing values of *k* and then plotting *k* against mean score. Then, the choice of optimal *k* is possible by selecting a value that corresponds with the highest mean score. Of course, if *k* is valued too high, it will become impossible to classify groups which contain few examples in them. This is not expected to be a significant problem in practice of course as there will be few examples in the live production environment.

Additionally, the model outputs should be normalised, so that only model predictions with the highest degree of certainty result in actions being taken. Any predictions with low degree of certainty will not be acted upon, and therefore, they will not contribute to the score. Once complete, the score will also be useful in the evaluation of results between different architectures.

In order to accomplish my goals, I will need to continue to enhance my model architectures and optimise their hyperparameters. This will be a very experimental phase that will involve building many variations of very similar models, which would benefit from more research on how to optimise Neural Network architectures. I plan to spend October analysing results, enabling me to further improve my models and investigate new approaches. I have scheduled to complete my final report and demonstration video during November and December.

Finally, as the project draws to a close, I will approach my employer for a second data export, which will be used to validate my results. Using these results and the experience I have gained of model deployment and maintenance I can assess which models will perform best in the proposed context.

Word count: 589

---

[5] Sanjay, M. (2018). Why and how to Cross Validate a Model?. [online] Towards Data Science. Available at: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f [Accessed 30 Sep. 2019].

[6] Nabi, J. (2018). Machine Learning — Multiclass Classification with Imbalanced Dataset. [online] Towards Data Science. Available at: https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a [Accessed 30 Sep. 2019].

[7] Shmueli, B. (2019). Multi-Class Metrics Made Simple, Part I: Precision and Recall. [online] Towards Data Science. Available at: https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2 [Accessed 30 Sep. 2019].