Interim Project Report

# Completed work

My aim for this project began as simply to build a system to help my colleagues access information. After initially considering an application based on a search engine, I hit upon the idea of constructing an email ‘conversational agent’ that could automatically respond to and process technical support requests. The system underpinning this would be an ‘intention classifier’ – a model designed to classify incoming email according to categories of user intent. The research and construction of this intention classifier is the focus of this project.

I first had to consider which technologies were most appropriate to the task. Artificial Intelligence today is a diverse field, on offer were a variety of technologies ranging from (now obsolete) rules-based implementations, to shallow learning methods like K-Nearest Neighbours (KNN), and more sophisticated Neural Network models such as Recurrent Neural Networks (RNN), Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU). I wanted to assess the viability of these technologies in practice, in the context of a typical work environment. As an industrial SME that does not specialised in software or data, my employer (Origin Frames ltd) is just such an example. A system implemented in this context would need 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 high level of set up and maintenance costs combined with limited results. However, a more intriguing question existed over which of the more modern methods produced better results. Neural Networks seemed more likely to produce better results, 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 must build and evaluate a number of these models.

Before I could begin working on any classification models, I first had to derive a data set. This involved a lengthy process of cleaning, formatting, anonymising and classifying over 700 email records. The data was cleaned by considering simple deterministic rules to determine which examples should never be responded too – for instance emails from outside the company’s domain name, or from within the support team itself. I 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 generic 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. However, I needed to go over the data personally at least three times to assure myself of complete anonymisation before commencing work. Additionally, the classification process required significant manual handling, although on reflection I could have made my work easier by implementing a K-means clustering algorithm to group similar emails.

Building models themselves has proved comparatively straightforward. I have been building KNN and RNN models with the Sci-Kit Learn and Keras libraries respectfully, using only a few lines of code.

**Word count: 570**

# Work to be done

Currently I am optimising the hyperparameters of my models to deliver better results. I have been using cross validation 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 seems ideal to analyse how performance 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’, which involve different ways of calculating a performance score based on four categories: True Positive, False Positive, True Negative and False Negative. In this instance a False Positive (sending a response which should have been ignored, or isn’t correct) is much more costly than a false negative (ignoring an email when an automated response or action would have been appropriate) as a false positive causes confusion, may introduce errors into production data, and requires 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 Precision, rather than Recall or Accuracy.

Additionally, the model outputs should be normalised so that only model predictions with the highest degree of certainty result in actions taken. Any predictions for which there is no clear classification will not be acted upon and therefore not contribute to the score. Elements in the confusion matrix can be aggregated based on how predictions would be used when implemented. This will require consideration for which classifications can neatly be handled in an automated fashion and therefore should be actioned automatically, and which are too complex or physical and must be handled manually by the support team. Once complete, this scoring work will be useful not just in cross-validation but also evaluation of performance between different architectures.

Using the cross-validation process will enable to me optimise hyperparameters more effectively, by enabling me to make objective design decisions. 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 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 with few examples in them. I don’t expect this to be a significant problem in practice however, as these groups will necessarily only rarely appear.

In order to accomplish my goals, I will need to build more models, based on LSTM and GRU algorithms. It is my intention to use the same Keras library that I used to build my RNN models, with which to build these. I anticipate that the implementation of these algorithms in code will be similarly complex.

Finally, as the project draws to a close, I will approach my employer for a second data export, with which to validate my results. Having then built several models using these technologies, I will be able to assess their suitability for use in a commercial environment, by considering ease of deployment and maintenance as well as performance in the form of precision.

**Word count: 565**

# Ethical Consideration

No human subjects were or will be involved in the production of this project, and there is no requirement for Ethical Approval.