Building a conversational agent for a closed domain

# Research question

What machine learning technologies and architectures will supply the most accurate responses in the context of a well-defined, closed domain?

Specifically, the project domain will be the provision of internal technical support within the offices of my current employer, Origin Frames ltd., a manufacturer of bespoke doors and windows.

# Objectives

The objectives featured in this section have been validated by building simple demonstrative prototypes. A plan of when and how such objectives will be achieved is set out in the Method and Plan sections and is based on this early prototyping.

#### Core

Research approaches to similar problems and select appropriate technologies and architectures.

Build two models, and train them on historical technical support interactions.

Build an application to host interactions with users through a technical support mailbox.

Analyse the feedback and compare the overall accuracy of the two models.

#### Advanced

Build additional machine learning models using a combination of available embedding models and shallow or deep learning algorithms.

Select answers from the models by using a ‘voting’ approach – then use the feedback to continuously train the models together.

Analyse and compare the performance of the models over time and category.

#### Conclusions

By completing this project, I will be able to make a recommendation as to which architectures are best suited to closed-domain question answering.

# Background

The earliest Question Answering systems originated in the 1960s and functioned by translating natural language into structured database queries (Dwivedi and Singh, 2013). However, early systems such as BASEBALL and LUNAR were always limited by the amount of pre-programmed information that they carried. This knowledge limitation was augmented in the 1990s by the arrival of the ‘World Wide Web’ (Katz, 1997), however the technology itself was fundamentally similar to earlier models; systems such as START were enhanced using a system of rules and site-maps rather than a database. In short, until very recently Question Answering platforms depended on large amounts of human effort and ingenuity – the apparent dynamism of such entities was a fragile illusion.

This archaic, labour-intensive methodology has been swept away with the rise of Neural Networks and Machine Learning, made possible through advances in hardware, particularly GPUs and TPUs (Jouppi et al., 2017). Modern applications of question answering typically utilise Neural Networks in one of two ways. An answer-selection approach usually employs recurrent neural networks as a classifier to match natural language questions onto user intentions (Mensio, 2019). A more advanced, and state of the art approach utilises a multi-layer architecture often incorporating Attention models and / or Sequence to Sequence generation to achieve its goal (Natural Language Computing Group, 2017), and typically relies upon a sample text within which to source its answers. A typical benchmark for such advanced models is the now-renowned Stanford Question Answering Dataset (Rajpurkar, 2016).

Both methodologies remove much of the human involvement required in historical models, however, they each have their optimal scenarios. An Answer Selection model performs well in a closed domain where original questions are unlikely and consistency of response is key, however, for an open domain a Machine Comprehension approach may be more appropriate as it can respond in ways that require no prior programming and can be expanded easily by simply adding text to the training material. This project will employ an answer selection approach, as it is better suited to a closed domain, and utilises a relatively light-weight, maintainable infrastructure.

Word count: 344

# Methods

#### The nature of the work

This will be a practitioner project, as I will be working with a real client in a live business environment. Training data will be provided by the client, in the form of historical technical support emails. I will be using this data to train a model to provide automated responses on the company’s two technical support mailboxes. While model accuracy will be crucial to it’s performance, it is equally important that the model is able to distinguish when there is no clear match – it should not be providing answers to questions that it cannot answer.

#### Methodology

The infrastructure of the application will require several stages of construction:

1. Infrastructure
   1. Retrieving historic and new emails from the server
   2. Access to send emails to users (limited to those on the @origin-global.com domain)
   3. Access to databases for retrieving dynamic results
2. Processing
   1. Identifying and cataloguing replies as separate from new enquiries
   2. Removing junk data including salutations / signatures and empty lines
   3. Extracting and storing identity data eg. Order numbers, User IDs etc
3. Intentions
   1. Identify user intentions by manually categorising the data
   2. Build compact dialogue trees for each intention to capture follow up data as required
   3. ‘Answers’ should be fully formed to aid in user verification and understanding
4. Modelling
   1. Utilising consistent input and output layers to ensure inter-compatibility
   2. Models to be built using a unique combination of architecture, layers and embeddings
   3. Enable each model to be serialised for implementation and backup
   4. Optimise each model by considering architecture and training time
   5. Provide some way of indicating confidence
   6. Store prediction results for future analysis
5. Application
   1. Consulting various models to produce a consensus
   2. Collect and store feedback, preferably directly from users but by domain experts if not
   3. Continuously train models based on user feedback
6. Implementation
   1. Soft launch – requires approval before email sent, allows for live integration testing
   2. Full launch – application can send automated responses without approval

#### Quantifying results

Typically, accuracy is regarded as the most simple and obvious metric of model performance. However, this is not a useful metric where the dataset is imbalanced (Fawcett, 2015). For instance, if the dataset was comprised of 80% login queries, a model could be built that predicted only login queries, and yet achieve an 80% accuracy. A much better tool for analysis of results is the confusion matrix (Nabi, 2018). It is possible using this tool to determine at a glance how each model performed at predicting each class, and which classes they mis-labelled with each other. Consistently poor performance in a particular class could indicate insufficient data.

# Word count: 447

# Plan

I have planned my work on a week-by-week basis as below. I refer to the stages listed above.

#### August

Week 1: Infrastructure

Week 2: Processing; Unit testing

Week 3: Intentions

Week 4 (SCP 3): Intentions

Week 5: Modelling; Unit testing

#### September

Week 1: Modelling; Unit testing

Week 2 (SCP 4): Application; Integration testing

Week 3: Complete IPR

Week 4: Complete IDV

#### October

Week 1 (IPR and IDV due): Implementation (Soft)

Week 2: Monitor results and maintain application

Week 3 (SCP 5): Monitor results and maintain application

Week 4: Monitor results and maintain application

#### November

Week 1: Implementation (Full)

Week 2 (SCP 6): Monitor results and maintain application

Week 3: Monitor results and maintain application

Week 4: FPR – Ethics, Problem Statement and Aims / Objectives

Week 5: FPR – Research Question, Methods and Methodology

#### December

Week 1: FPR – Application and Evaluation

Week 2: FPR – Discussion, Conclusions, Future Developments

Week 3: FPR – finish for first draft, submit to supervisor for review

Week 4: FDV

#### January 2020

Week 1: Final amendments

Week 2 (FPR and FDV due): Final amendments

# Resources

I will require access to a number of readily available resources belonging to the client:

* SQL Database access for retrieving and storing data dynamically
* Remote Desktop access for deploying the app
* Company email address for the app to send from
* Inclusion on the 2 technical support mailboxes within the company

Together with some resources that I hold privately:

* Sufficient compute resource to build and train neural network models
* VS Code (Integrated development environment) with Python (Anaconda distribution)
* Keras, TensorFlow, PyInstaller and Pyzmail libraries
* Access to a private github repository

And some skills that I have acquired during the course of my research:

* General python programming
* Using source control to back up and maintain project
* Manipulating Numpy arrays in Python
* Manipulating Pandas dataframes in Python
* Using relational databases to store and retrieve data
* Serializing and deserializing Python data structures using PKL files
* Building models using Keras (incorporating TensorFlow)
* Deploying Python applications using PyInstaller
* Sending and receiving emails using IMAP, SMTP and PyZmail

# Relation to target award

There are a number of components or aspects of this project that make it suitable for a Master’s level Computer Science Project:

* Analysing results to build, debug and optimise software
* Designing, building and training Neural Networks
* Storing and retrieving information efficiently using a relational database
* Programming using efficient, readable code
* Using a modern programming language (Python) and tools (github)
* Working with a real client to achieve a goal using software

-

# Ethics approval

(**1 mark**): describe how to obtain Ethics Approval for an MSc project in the University of Hertfordshire; and a brief discussion on whether you will need Ethics Approval for your work.

# References

Dwivedi, S. and Singh, V. (2013). Research and Reviews in Question Answering System. *Procedia Technology*, 10, pp.417-424.

Katz, B. (1997). Annotating the world wide web using natural language. In: *RIAO '97 Computer-Assisted Information Searching on Internet*. Paris: LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, pp.136 - 155.

Jouppi, N. et al. (2017). In-Datacenter Performance Analysis of a Tensor Processing Unit. In: *44th International Symposium on Computer Architecture (ISCA)*. [online] Available at: https://arxiv.org/ftp/arxiv/papers/1704/1704.04760.pdf [Accessed 14 Jul. 2019].

Natural Language Computing Group (2017). *R-NET: MACHINE READING COMPREHENSION WITH SELF-MATCHING NETWORKS*. [online] Microsoft Research Asia, pp.1 - 11. Available at: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf [Accessed 14 Jul. 2019].

Rajpurkar, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. In: *2016 Conference on Empirical Methods in Natural Language Processing*. New York: Association for Computational Linguistics, pp.2383 - 2392.

Fawcett, T. (2015). *The Basics of Classifier Evaluation: Part 1*. [online] Silicon Valley Data Science. Available at: https://www.svds.com/the-basics-of-classifier-evaluation-part-1/ [Accessed 14 Jul. 2019].

Nabi, J. (2018). *Machine Learning — Multiclass Classification with Imbalanced Data-set*. [online] Medium. Available at: https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a [Accessed 14 Jul. 2019].