Task-Oriented Dialogue Classification: MLP vs SVM

Ollie Keers

Brief description and motivation of the problem

Task-oriented dialogue has become a feature of modern life with both voice-assistants and chatbots becoming ubiquitous. After receiving natural language input of a few words from the user, often uncontextualized, the agent must determine what is being asked before carrying out the task accordingly. Such queries can result in many diverse tasks such as information retrieval ("What is my booking reference?")[1], knowledge graph querying ("How many people live in the capital of Belize?")[2], or task execution ("Call John")[3]. In all cases, the user expects both rapid and correct execution of their request.

Domain-specific chatbots are also used to handle customer interactions. Such agents are more limited in scope and are therefore more likely to receive inputs that they are unable to process correctly. It is therefore important for these agents to be able to recognise when a query is out of scope to enable an appropriate action to be taken, such as passing the request over to a human worker. Performing such classification is non-trivial, with considerable work being devoted to this task in recent years[4]–[6].

Text classification is a well-studied field within NLP, with spam filters[7] and sentiment analysis[8] just two examples of common areas of work. However, task-oriented dialogue poses several challenges not faced in other fields of text classification, namely: a short input text length of a few words, a large breadth of potential classifications (including out-of-scope) and a user-led requirement for rapid performance. This objective of this work is to develop a neural model that is able to perform well when faced with these three requirements.

Description of the dataset including data types (discrete, continuous, etc.)

CLINC 150[5] is a dataset of text queries generated for the purposes of intent classification. Rather than labelling pre-existing queries, the dataset was generated by producing a list of 10 general domains and asking crowd workers to generate commands and questions within that domain. These domains are: Banking, Credit Cards, Kitchen & Dining, Home, Auto & Commute, Travel, Utility, Work, Small Talk, and Meta. Once a small number of queries had been generated for each domain, they were further split into 15 intents per domain. Crowd workers were then asked to generate an expanded corpus through rephrasing an existing input or developing a scenario-contextualised version of it. Finally, out of scope (oos) queries were generated from both inputs that had not matched one of the 150 intents, and through creating new inputs, in a similar manner outlined previously, that were outside of the 150 intents. These out of scope queries can therefore belong to one of the 10 domains and be fairly similar to an intent, or very far removed. We have used the full dataset of 23700 queries for this work, which are already split into separate training, validation & testing sets, outlined in Table 1.

	Training	Validation	Testing	TOTAL
In Scope	150 x 100	150 x 20	150 x 30	22500
Out of Scope	100	100	1000	1200
TOTAL	15100	3100	5500	23700

Table 1. The distribution of instances across the dataset.

All words in the queries are lowercase, and do not have punctuation at the end of the sentence. Some examples of these, with corresponding intent, are given in Table 2.

Query	Intent
how could i say twin in chinese	translate
how do i get my pin number, i forgot mine	pin_change
set a ten second timer	timer
affiliate's definition is what	definition
i wanna know the meaning of life	meaning_of_life
can my credit limit on my discovery card go up	credit_limit_change
what is the largest state in the us	oos
can you list me tiger wood's stats	oos

Table 2. A selection of sample queries and their intents from the training data.

The choice of sentence representation method has a notable effect on classification accuracy [6]. The authors of the dataset generation paper[5] took different approaches when examining different classifiers, which makes understanding the origins of the observed differences in their performance challenging. This work is not intended to compare different encoding methods, and so uses a consistent approach using simple TF-IDF vectors. This bag of words approach has been selected due to its use as a baseline in other task-oriented dialogue classification work using neural models.[9] Further, a decision was taken to only use unigrams rather than larger n-grams was taken to minimise the pre-processing that would be required upon receiving a user query in a real deployment situation. For the same reason, no further text processing was performed, including lemmatisation or stop-word removal.

The vocabulary was generated from the 15100 training examples, including 100 out of scope queries, and consisted of 5146 words. While a larger vocabulary could have been used this would actually be detrimental to performance as each of these features would not exist in the training set, but would add computational time to consider the zero-weighted feature. As can be seen in Table 3, this vocabulary includes numbers which can indicate times of the day or other information that may be of use for classifying intent. A TF-vectorizer was fitted to the training data, before being used to separately transform each of the datasets into a sparse matrix with 5146 features.

00	000	005	00am	00pm	[…]	ziti	zombie	zone	zoo	zulu

Table 3. The first and last 5 words in the vocabulary for this model.

<u>Brief summary of the two neural network models with their pros and cons</u>

<u>Multilayer Perceptron:</u>

The multilayer perceptron is a supervised learning model, consisting of a minimum of three layers of neurons: an input layer, at least one hidden layer, and an output layer. The neurons receive weighted inputs from the prior layer, and in the event that this is greater than their bias will fire forward into the next layer. Activation functions can be applied to each layer of neurons to adjust behaviour, making MLP suitable for both classification and regression tasks through applying different activation functions to the output layer. The MLP is trained through backpropagation of errors, estimating the gradient of the error surface and using an algorithm to descend the surface towards a minimum, representing maximum accuracy. Each iteration of this optimization adjusts the weights of the perceptron until a minimum is reached. Because the MLP is a universal approximator, there is a high risk of overfitting to the training data.

Pros	Cons
- Universal approximator, so very wide range of applications - Overfitting can be mitigated with dropouts, early stopping etc. - Local minima can be avoided by using optimizer with momentum - No need to perform feature selection on the data prior to model creation - Can be run on GPUs for faster calculations	- High risk of overfitting to training data. - Mitigating overfitting increases computational demand - Can find a local, not global, minimum - Difficult to impossible to explain what is going on in the MLP 'black box' - Vast number of parameters to tune - Slow on CPUs - Need a lot of training data to do well

<u>Support Vector Machines</u>

Support vector machines are also a supervised learning algorithm, whereby the aim is to maximise the margin of the hyperplane between classes. As this margin depends upon the points closest to the classification boundary, these are the ones considered in optimisation (the support vectors). Optimization therefore depends upon a smaller subset of data points and so is less costly to solve. As the aim is to maximise the margin, the optimal solution is found rather than any solution. While this approach would, in theory, limit SVMs to only be able to solve linearly separable problems, this can be solved by use of the 'kernel trick'. This involves mapping the points into higher dimensional space, separating the points with a hyperplane and then re-mapping this back to the original space. This allows for non-linear boundaries, and vastly increases the cases that SVMs can be used in. SVMs can be used for either regression or classification, with the approach outlined here only covering binary classification. This can be generalised to multiclass classification through either a one-vs-one approach, where a classification boundary is drawn between each pair of classes, or one-vs-rest where a boundary is drawn between each class and all of the others. The former of these results in far more boundaries, and thus requires many more calculations.

Pros	Cons
- Finds optimal solution not local minimum - Can solve non-linearly separable problems - Good generalisability - Wide range of basis functions allows for many problems to be solved - Works well with high-dimensional data	- Non-probabilitistic predictions limit interpretability - Can be costly for large datasets and multi-class problems where the number of classes is high - Risk of overfitting to training data - Use of non-linear basis functions can significantly increase computational demand

Hypothesis statement

Both MLP and SVM will produce high classification accuracy on the test set. Both models are powerful neural computing models, that have achieved high accuracy with this dataset [5] with different encoding techniques. We have used TF-IDF for the encodings, which is slightly more powerful than Bag-of-Words used for SVM, so we anticipate an increase in accuracy for this; conversely USE (which was used for the MLP) is more powerful than TF-IDF[9], [10] so we would expect a decrease in performance. Classification accuracy should therefore be similar for both models.

Performance on out-of-scope data will be much weaker than in-scope data. The challenges of this task are well documented[4]–[6], and we anticipate this performance drop will be considerable.

SVM will be considerably more expensive than MLP. With 151 classes, many decision boundaries need to be computed (151 for one-vs-rest, 11325 for one-vs-one) each dependent upon multiple vectors, which will be particularly costly for non-linear kernels. For the MLP, cost will increase with complexity of the perceptron but based upon our reference paper this can be limited to a single layer of 400 hidden neurons [5].

Description of choice of training and evaluation methodology

We retain the training/validation/testing split provided with the data. Beyond being for comparability purposes, this split partitions queries by a particular author into one of the sets to limit similarities between them[5]. For SVM, the model is fitted to the entirety of the training data, while for MLP 3-fold cross validation of the training set is performed to enable Early Stopping. 3-fold has been chosen to limit computation. In both cases, the fitted model is then used to predict the validation intents (both in scope and out of scope). Classification accuracy and prediction time on this validation data are used to evaluate model performance during tuning.

For the multilayer perceptron the parameters tuned are: number of hidden layers, number of neurons in the hidden layer(s), hidden layer activation function(s), dropout rate(s), learning rate, early stopping patience, optimisation algorithm, and momentum (where appropriate).

For the support vector machines, the parameters tuned are: kernel, loss function, regularization constant and implementation (one vs one in libsvm or one vs rest in liblinear[11]).

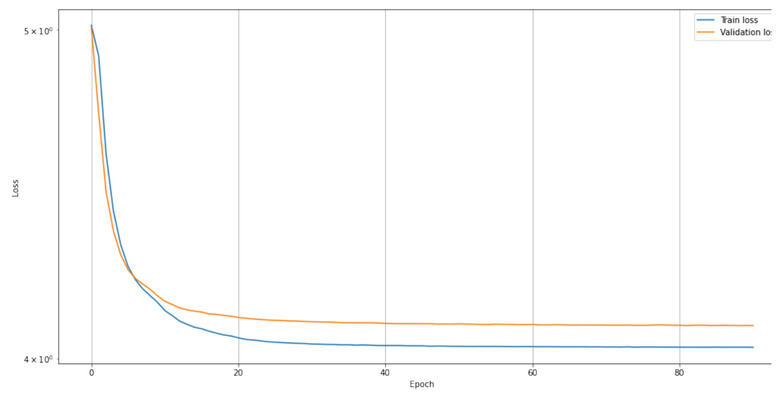


Fig 1. Training & validation loss for Multilayer Perceptron.

Due to the number of parameters selected for tuning (8 for MLP), and the resources available (a single CPU), a grid search is not practical. Common alternatives include random search and Bayesian optimisation, but an interesting recent work framed it as an RL problem[12]. Here an agent maximises accuracy for a parameter at a time, with their order not significant. This repeated linear search considerably cuts down on dimensionality, and so drastically reduces computation. This approach will be employed in this work, but with a human, not artificial, agent, allowing for simultaneous optimisation of both evaluation metrics - classification accuracy and time. Iterative optimization continues until no improvement is seen on the evaluation metrics. Drawbacks of this approach include increased user input and the risk of finding a local rather than global maximum.

Choice of parameters and experimental results:

Multilayer Perceptron

Optimizer	Hidden Layers	Hidden Layer Size	Hidden Layer Activation	Dropout	Learning Rate	Early Stopping Patience
Adam	1	800	ReLU	0.75	0.001	10

Table 4. Optimal hyperparameters for Multilayer Perceptron.

The initial configuration of the MLP was chosen to be that found to be optimal in our reference paper[5], despite their initial query encoding with USE embeddings[10]. This consisted of a single hidden layer of 400 neurons with a tanh activation function and a dropout of 0. As this is a multiclass classification problem, softmax was used on the output layer and cross-entropy as the loss function in all cases. Stochastic gradient descent was the initially chosen optimizer but failed to achieve acceptable results (<1% accuracy) after varying the learning rate, momentum and dropout rate. We next changed Adam, chosen because it performs well with large datasets[13], and obtained an initial classification accuracy of 87%. From here we iteratively tuned the parameters outlined above, with intermediate results given in the implementation details.

SVM

The initial work was to select the basis function, with linear performing the best. Accuracy of LinearSVC was very similar to that of SVC using a linear kernel (~0.1%) but offered much quicker classification (~4000x faster). Both of these were tuned to provide their optimal classification, with LinearSVC deemed to perform best within this application domain due to its superior speed.

Implementation	Loss Function	Regularization Constant
LinearSVC (one vs rest)	Hinge Squared	2.5

Table 6. Optimal hyperparameters for Support Vector Machine.

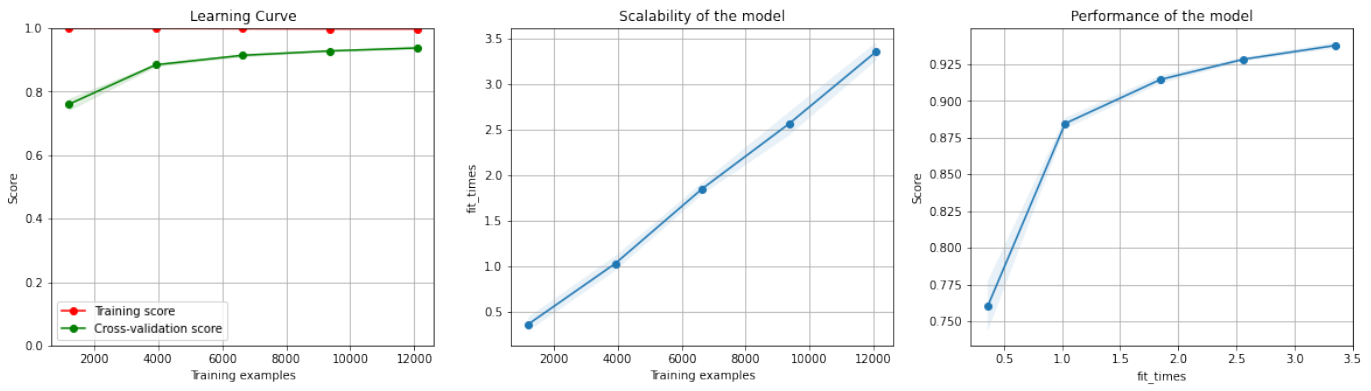


Fig 2. Learning curves for Support Vector Machines.

Analysis and critical evaluation of results:

As can be seen from Fig.3, the two best models performed very similarly on all datasets, with the greatest discrepancy in performance being 3% for the out-of-scope validation data. Applying McNemar's test to the queries that have been misclassified by each of the models, Table 7 & 8, we find that the difference in classification accuracy of the two models is not significant (p=0.646 for in scope and p=0.488 for out of scope). Some of the mutually misclassified queries are given in Fig. 4. Some queries, such as 'Who set up the numbers for it" (intent: timer) would be difficult for a human to classify. Others may represent a high weight being given to particular words, e.g. "what veggies can I pair with mushrooms" (intent: out of scope) was classified as sync_device by both classifiers, one may think due to 'pair with'. This could potentially be rectified with a larger training corpus and vocabulary, as even the use of bigrams in this case may not have rectified the classification. Both models have struggled on similar intents, Fig. 5, with common misclassifications including change_user_name to user_name or change_ai_name, and distance to timer or directions. MLP has misclassified many OOS queries as greeting, while both have misassigned many queries to w2 and calculator. The black-box nature of these models makes understanding the origin of the issues difficult, and therefore also to fix them. The application domain does value user-interpretability of reasoning, so the extent of this problem is limited to the programmer.

In Scope Queries	Incorrect (SVM)	Correct (SVM)
Incorrect (MLP)	308	82
Correct (MLP)	89	4021

Table 7. Classification matrix for test data for in scope queries.

Out of Scope Queries	Incorrect (SVM)	Correct (SVM)
Incorrect (MLP)	797	47
Correct (MLP)	55	101

Table 8. Classification matrix for test data for out-of-scope queries.

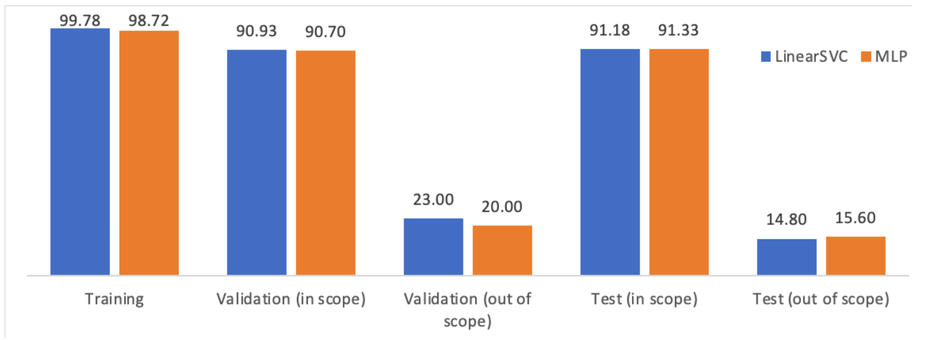
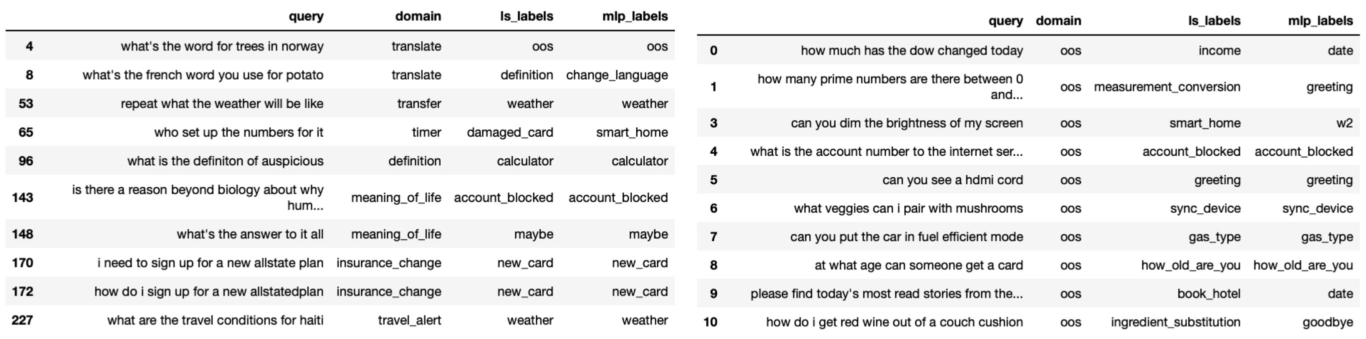


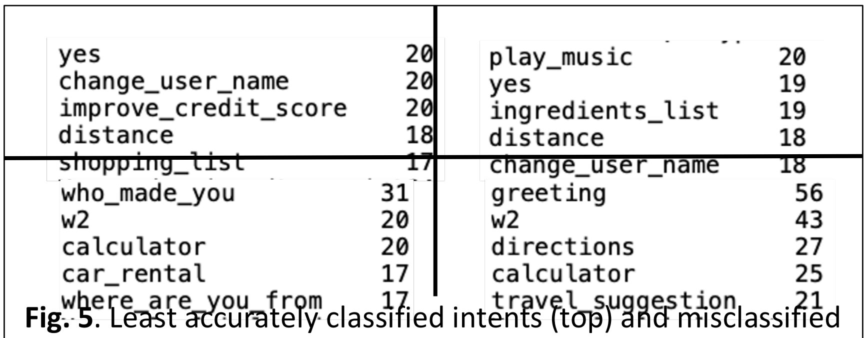
Fig 3. Classification accuracy for best two models.



	query	domain	ls_labels	mlp_labels
4	what's the word for trees in norway	translate	oos	oos
8	what's the french word you use for potato	translate	definition	change_language
53	repeat what the weather will be like	transfer	weather	weather
65	who set up the numbers for it	timer	damaged_card	smart_home
96	what is the definiton of auspicious	definition	calculator	calculator
143	is there a reason beyond biology about why hum...	meaning_of_life	account_blocked	account_blocked
148	what's the answer to it all	meaning_of_life	maybe	maybe
170	i need to sign up for a new allstate plan	insurance_change	new_card	new_card
172	how do i sign up for a new allstatedplan	insurance_change	new_card	new_card
227	what are the travel conditions for haiti	travel_alert	weather	weather

	query	domain	ls_labels	mlp_labels
0	how much has the dow changed today	oos	income	date
1	how many prime numbers are there between 0 and...	oos	measurement_conversion	greeting
3	can you dim the brightness of my screen	oos	smart_home	w2
4	what is the account number to the internet ser...	oos	account_blocked	account_blocked
5	can you see a hdmi cord	oos	greeting	greeting
6	what veggies can i pair with mushrooms	oos	sync_device	sync_device
7	can you put the car in fuel efficient mode	oos	gas_type	gas_type
8	at what age can someone get a card	oos	how_old_are_you	how_old_are_you
9	please find today's most read stories from the...	oos	book_hotel	date
10	how do i get red wine out of a couch cushion	oos	ingredient_substitution	goodbye

Fig. 4. A selection of queries incorrectly classified by both classifiers for in/out of scope.



```
yes                     20
change_user_name        20
improve_credit_score    20
distance                18
shopping_list           17
```

```
play_music          20
yes                 19
ingredients_list    19
distance            18
change_user_name    18
```

```
who_made_you          31
w2                    20
calculator            20
car_rental            17
where_are_you_from    17
```

```
greeting             56
w2                   43
directions           27
calculator           25
travel_suggestion    21
```

Fig. 5. Least accurately classified intents (top) and misclassified oos intents (bottom) by SVC (left) and MLP (right).

Both models performed notably better on the in-scope then the out-of-scope data, with both achieving ~91% accuracy on the in-scope and ~15% on the out-of-scope test data. Our reference paper[5] also saw a similar decrease in performance with 91.0% using SVM and 93.5% using MLP on the in scope data, and 14.5% and 47.4% on the out of scope data, respectively. As noted previously, the different encoding means their MLP results are not directly comparable to ours. Our optimization methodology may have only found a local maximum, but the similar accuracy of both models suggests they have reached the global maximum for this encoding approach, and that improved performance would require more sophisticated text transformations.

It is perhaps unsurprising that models struggle to correctly classify queries as "out-of-scope". This class represents everything that does not fit into one of the other 150 categories but has also been trained as a class consisting of only 100 examples. This means that out of scope examples that closely resemble the out-of-scope training examples are likely to be correctly classified, but any other examples will not. It is possible that a query would contain very few, if any, words from within our training vocabulary that are uncommon, and so be classified solely on the basis of stop words.

	SVM	MLP
Training Time (s)	3.71	573
Validation Prediction Time (s)	0.109	0.378
Test Prediction Time (s)	0.213	0.674

Table 9. Time taken to execute various operations.

Performance of the two models in terms of time, however, is very different, as seen in Table 9. To achieve a more accurate value for these the average fitting time was taken over 1000 attempts (SVM), and 10 full optimizations with early stopping (MLP); prediction time is also given averaged over 1000 attempts. Training time for the SVM was around 200x faster than for the MLP, and also required significantly less tuning to arrive at the final model. It is worth noting that this does not hold true for the one-vs-one implementation of SVM, where training took approximately 800s and validation prediction around 300s for the best model. The performance increase from these two different SVM implementations is very impressive. The problem here was linearly separable, which allowed for this impressive linearly scalable performance. If the problem had required a radial or polynomial basis function, or a one-vs-one implementation had been required, this result would not have been possible. It is also interesting that one-vs-one performs much approximately twice as well on the out-of-scope data suggesting the added complexity can pay off within this particular task.

Conclusions, lessons learned, references and future work:

As hypothesised, we have found that both models perform with a similar, high classification accuracy on in-scope data and that this decreases substantially for both models on out-of-scope data. Differences in their accuracy were not found to be significant. Contrary to our hypothesis, SVM was much faster than MLP both to train and to test, due to the linear separability of classes and application of a one-vs-rest approach.

The speed and accuracy of one-vs-rest SVM is impressive and suggests high scalability, with linear complexity. From these results, SVM is superior to MLP for this application. Given the poor performance on out-of-scope data, increasing the training examples to reflect the potential extent of real-world queries would be likely to yield improved accuracy with relatively small increase in cost. Inverting the testing and training sets for out-of-scope data could facilitate exploratory work before embarking upon more substantial work.

Treating optimization as an RL problem does appear to have provided a suitable, computational resource-efficient grid-search alternative. The number of tuneable parameters means optimizing an MLP will always be resource intensive, be it human or computational. Optimizing the MLP required days of work from the human agent, adjusting parameters for small improvements in performance. Random starting weights mean the observed increases in performance can be random, and so repetition is necessary to ensure that these persist. Optimizing SVM is a much simpler process for the user, as well as taking less computational time. The RL approach provides an interesting alternative to other approaches in resource-constrained situations.

The performance stated in our reference paper suggests that improved performance is possible with more complex pre-processing[5], so investigation of more complex word embeddings, from use of n-grams up to state-of-the-art approaches like BERT provides a rich seam for future work. Given there were queries that were only misclassified by one of our models, it may be possible to obtain improved results through an ensemble approach, although this would come at increased computational expense.

References:

[1] P. Budzianowski *et al.*, ‘MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling’, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018, pp. 5016–5026, doi: 10.18653/v1/D18-1547.

[2] V. Zahariev, D. Shunkevich, S. Nikiforov, and E. Azarov, ‘Intelligent Voice Assistant Based on Open Semantic Technology’, in *Open Semantic Technologies for Intelligent System*, vol. 1282, V. Golenkov, V. Krasnoproshin, V. Golovko, and E. Azarov, Eds. Cham: Springer International Publishing, 2020, pp. 121–145.

[3] D. Braun, A. Hernandez-Mendez, F. Matthes, and M. Langen, ‘Evaluating Natural Language Understanding Services for Conversational Question Answering Systems’, in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, Saarbrücken, Germany, 2017, pp. 174–185, doi: 10.18653/v1/W17-5522.

[4] Y. Zheng, G. Chen, and M. Huang, ‘Out-of-Domain Detection for Natural Language Understanding in Dialog Systems’, *IEEEACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 1198–1209, 2020, doi: 10.1109/TASLP.2020.2983593.

[5] S. Larson *et al.*, ‘An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction’, 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)*, Hong Kong, China, 2019, pp. 1311–1316, doi: 10.18653/v1/D19-1131.

[6] S. Ryu, S. Kim, J. Choi, H. Yu, and G. G. Lee, ‘Neural sentence embedding using only in-domain sentences for out-of-domain sentence detection in dialog systems’, *Pattern Recognit. Lett.*, vol. 88, pp. 26–32, Mar. 2017, doi: 10.1016/j.patrec.2017.01.008.

[7] H. Taniguchi, H. Sato, and T. Shirakawa, ‘A machine learning model with human cognitive biases capable of learning from small and biased datasets’, *Sci. Rep.*, vol. 8, no. 1, p. 7397, Dec. 2018, doi: 10.1038/s41598-018-25679-z.

[8] E. Cambria, S. Poria, A. Gelbukh, and M. Thelwall, ‘Sentiment Analysis Is a Big Suitcase’, *IEEE Intell. Syst.*, vol. 32, no. 6, pp. 74–80, Nov. 2017, doi: 10.1109/MIS.2017.4531228.

[9] M. Henderson *et al.*, ‘Training Neural Response Selection for Task-Oriented Dialogue Systems’, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, Aug. 2019, pp. 5392–5404, Accessed: Feb. 25, 2021. [Online]. Available: https://www.aclweb.org/anthology/P19-1536.pdf.

[10] D. Cer *et al.*, ‘Universal Sentence Encoder for English’, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Brussels, Belgium, 2018, pp. 169–174, doi: 10.18653/v1/D18-2029.

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, ‘LIBLINEAR: A Library for Large Linear Classification’, *J. Mach. Learn. Res.*, vol. 9, no. 61, pp. 1871–1874, 2008.

[12] J. Wu, S. Chen, and X. Liu, ‘Efficient hyperparameter optimization through model-based reinforcement learning’, *Neurocomputing*, vol. 409, pp. 381–393, Oct. 2020, doi: 10.1016/j.neucom.2020.06.064.

[13] D. P. Kingma and J. Ba, ‘Adam: A Method for Stochastic Optimization’, *ArXiv14126980 Cs*, Jan. 2017, Accessed: Feb. 25, 2021. [Online]. Available: http://arxiv.org/abs/1412.6980.

Glossary

Word	Definition
Activation function	A function that is applied to the output of neurons
Adam	Adaptive movement estimation. A gradient descent algorithm that adjusts learning rates for each parameter individually, and so can converge much quicker than SGD
Agent	An entity (often computational) that takes actions based upon its observations
Backpropagation	Through taking derivatives of the error surface and passing this backwards through the MLP, it is possible to adjust the weights so as to reduce the error – resulting in a better solution.
Classification	Choosing a group from a defined number of groups
Classification accuracy	The number of examples assigned to the correct class divided by the total number of examples. A measure of the proportion classified correctly.
Corpus	Collection of all of the texts
Cross-entropy loss	A loss function that is commonly used in multiclass classification tasks. This involves the sum of the losses, taking into account both the number of examples assigned to a particular class and how many instances should be assigned to that class.
Greedy approach	Making the decision that maximises a particular objective at each step
Intent	The purpose for which the user initiated the query
Learning Rate	The speed at which the model improves: a large learning rate means overshooting the minimum is likely, a small learning rate will take longer to reach the optimal solution
Lemmatization	Combining multiple forms of the same word, e.g. walk & walking
Momentum	Uses the error from previous epochs to adjust the weight changes for this epoch. This is implemented to help avoid finding local minima as the momentum from previous iterations is likely to ‘push’ you above the saddle of a local minimum.
n-grams	Combinations of multiple words that appear consecutively within a text. A bigram would consist of two adjacent words e.g.: “two adjacent” & “adjacent words”
Neuron	A simple processing unit that takes a weighted input and produces an output, depending on if that weighted input is greater than its threshold (bias)
NLP	Natural language processing, modelling that involves the use of normal text inputs
Overfitting	This occurs when a model too closely resembles the training data, leading to very high performance on this data but poor performance on unfamiliar data.
Regression	Predicting a value
ReLU	A function that is zero for negative numbers, and the number itself for positive numbers
RL	Reinforcement Learning. Where the objective is to maximise a ‘reward’ observed for the outcome of a particular action in repeating cycles
SGD	Stochastic Gradient Descent. This approach takes a random subset of the data, calculates the gradient and moves in the direction of greatest gradient for that subset. The subset is chosen to limit computational complexity, but this does mean that the direction chosen is not necessarily the best direction for the whole dataset.
Sigmoid	An s-shaped activation function ranging from 0 to 1
Softmax	A function that only selects the maximum value from its inputs; other inputs are ignored
Stop-words	Very common words, such as ‘the’, ‘it’, and ‘and’.
Supervised learning	Modelling where target values are known
Tanh	Hyperbolic tangent, a s-shaped activation function ranging from -1 to 1
TF-IDF	A process whereby each word within the vocabulary is given a unique number. Each time that word appears within a text(query in this case) it is counted (Text Frequency). This is then multiplied measure of its rarity within the corpus (Inverse Document Frequency)

Implementation details & intermediate results:

All times are given as executed on a 2.3 GHz 8-Core Intel Core i9 CPU

Green highlighting represents results taken forward to next stage of optimization.

MLP key results:

Optimized parameter & values attempted	Validation Accuracy
1. SGD Optimizer, 1 hidden layer, 400 neurons, tanh activation: varied dropout rate 0, 0.1, 0.5, 1; learning rates 10, 5, 1, 0.5, 0.1, 0.01, 0.001, 0.0001; momentum 10, 5, 1, 0.5, 0.1	All <= 1%
2. Adam Optimizer (other parameters as above), dropout rate 1, 0.5, 0.1, 0.01, 0.001, 0	87.77%
3. Adam, dropout 0.5, learning rates: 10, 5, 1, 0.5, 0.1, 0.01, 0.001, 1e-4, 1e-5, 1e-6	89.63%
4. Adam, dropout 0.5, learning rate 0.001, hidden layer size: 100, 200, 400, 800, 1600	90.1%
5. Adam, dropout 0.5, learning rate 0.001, 2 hidden layers both same size of: 100, 200, 400, 800	Best for 800 at 88.7%. Worse than 1 layer, discounted.
6. Adam, dropout 0.5, learning rate 0.001, hidden layer size: 25, 50, 75, 151	All worse than 800
7. Adam, dropout 0.5, learning rate 0.001, hidden layer size: 800, activation functions for hidden layer: relu, linear, sigmoid	90.4%
8. Adam, learning rate 0.001, hidden layer size: 800, activation functions for hidden layer, dropout: 1, 0.9, 0.8, 0.75, 0.5, 0.25, 0.1, 0.01, 0	90.7%
9. Adam, hidden layer size: 800, activation functions for hidden layer, dropout 0.75, learning rate: 0.0001, 0.01	Worse than 0.001
10. Hidden layer size: 800, activation functions for hidden layer, dropout 0.75, learning rate: 0.001, SGD, momentum 10, 1, 0.1, 0.01, 0.001, 0.0001	Worse than Adam
11. Adam, activation functions for hidden layer, dropout 0.75, learning rate: 0.001, hidden layer size: 50, 100, 200, 400, 800, 1600, 3200, 6400, 12800	Remains optimal
12. Adam, activation functions for hidden layer, dropout 0.75, learning rate: 0.001, two hidden layers of size:100, 200, 400, 800, 1600	Worse than one layer
13. Adam, activation functions for hidden layer, dropout 0.75, learning rate: 0.001, no hidden layers	Worse than one layer
14. Adam, activation functions for hidden layer, dropout 0.75, learning rate: 0.001, hidden layer size 800, early stopping patience: 5, 10, 20, 50 epochs	90.9%
15. Adam, activation functions for hidden layer, dropout 0.75, learning rate: 0.001, hidden layer size 800, no early stopping	Worse (90.1%)
16. Optimal values adopted as no further improvement seen since iteration 14.	

SVM key results:

Optimizing parameter	Parameter value Training time (s) / validation prediction time (s) / Out of Scope prediction time (s) Classification Accuracy: Training %/ In Scope % / Out Of Scope %				
Optimization cycle 1: kernel	Linear (one-v-rest): 2.7 / 0.05 / 0.03 9.6% 90.9% / 23%	Linear (one-v-one): 851 / 219 / 8 99.2% / 90.8% / 45%	Radial: 1329 / 236 / 8 99.7% / 89.4% / 64%	Polynomial: 1761 / 1320 / 33.8 99.7% / 78.1% 78%	Sigmoid: 10707 / 3652 / 313 98.0% / 90.3% / 45%
Optimization cycle 2: regularization & loss function	Linear (one-v-rest) Hinge Squared Loss, C=2.5: 3.2 / 0.06 / 0.03 99.8% / 90.9% / 23%	Linear (one-v-one), C=5: 4183 / 4175 / 7.6 99.9% / 91.5% / 45%	Linear (one-v-rest), Hinge Loss, C=10: 3.07 / 0.05 / 0.02 99.8% / 90.3% / 22%		

For deciding between the One-V-Rest and One-V-One models, the small increase in classification accuracy for One-Vs-One was deemed not worth the significantly increased computational cost, even with its superior performance on the out-of-scope examples, due to the response speed necessitated by the task in question.