“Where Is My Parcel?” Fast and Efficient Classifiers to Detect User Intent in Natural Language

Constantina Nicolaou[‡], Amal Vaidya[‡], Fabon Dzogang[†], David Wardrope[‡] and Nikos Konstantinidis[‡]
[‡]Department of Physics and Astronomy, University College London, London, UK
Emails: constantina.nicolaou.17@ucl.ac.uk
amal.vaidya.15@ucl.ac.uk
d.wardrope@ucl.ac.uk
n.konstantinidis@ucl.ac.uk
[†]ConversationalAI Team, ASOS AI, London, UK
Email: fabon.dzogang@asos.com

***Abstract*—We study the performance of customer intent classifiers designed to predict the most popular intent received through ASOS.com Customer Care Department, namely “Where is my order?”. These queries are characterised by the use of colloquialism, label noise and short message length. We conduct extensive experiments with two well established classification models: logistic regression via n-grams to account for sequences in the data and recurrent neural networks that perform the extraction of these sequential patterns automatically. Maintaining the embedding layer fixed to GloVe coordinates, a Mann-Whitney U test indicated that the F1 score on a held out set of messages was lower for recurrent neural network classifiers than for linear n-grams classifiers (M1=0.828, M2=0.815; U=1,196, P=1.46e-20), unless all layers were jointly trained with all other network parameters (M1=0.831, M2=0.828, U=4,280, P=8.24e-4). This plain neural network produced top performance on a denoised set of labels (0.887 F1) matching with Human annotators (0.889 F1) and superior to linear classifiers (0.865 F1). Calibrating these models to achieve precision levels above Human performance (0.93 Precision), our results indicate a small difference in Recall of 0.05 for the plain neural networks (training under 1hr), and 0.07 for the linear n-grams (training under 10min), revealing the latter as a judicious choice of model architecture in modern AI production systems.**

***Index Terms*—text classification, customer intent, n-grams, conversational-AI, recurrent neural networks, word-embedding.**

I. Introduction

Over the past couple of decades, there have been significant advancements in the scientific research of computational linguistics and natural language processing (NLP) which led to many practical technological applications in consumer products. These advancements were facilitated by the vast increase of available data used to train Machine Learning models, the development of highly efficient learning algorithms and the substantial increase in computational power. NLP is a sub-field of Artificial Intelligence that is focused on using computational techniques to understand, manipulate and produce natural language. Natural language, whether spoken or written, is the most natural means of communication between Humans with the associated challenges including being able to handle colloquialism, variability, ambiguity and context-dependent interpretation of Human languages.

With online fashion sales increasing year-on-year, one of the challenges retailers face is the ability to handle customer care queries efficiently. ASOS.com is a global e-commerce fashion company with the entirety of their sales originating online. With millions of customers across the globe their customer care advisors handle large numbers of contacts every week through live chat, email and dedicated social media. The time of a customer care advisor is extremely valuable and as such making their jobs as easy and efficient as possible is of great importance to the company. Customer care is the front face of any company and hence has to be operating flawlessly as a smooth customer experience is crucial. AI-powered chatbots working alongside Human agents can help accelerate customer service and provide increased functionality.

The top most frequent customer query received through ASOS.com customer care is *“Where is my order?”* (WISMO). These queries can often be resolved by simply communicating the order status to the customer or advising them of the expected delivery date. In a typical scenario, advisors have to process long message queues manually in a two step process: they first identify a correct intent for the customer and then initiate a conversation to seek query resolution. In this work we study two different approaches to build an efficient customer intent classifier. This will improve cost-efficiency in customer care as once the correct intent is automatically identified, the relevant information can be served to the advisor, or used directly by the Conversational AI agent to assist the customer when the request lends itself to automated resolution (e.g customers contacting to inquire about their parcel before the expected delivery date).

Also, the recent successes of large transformer models [1]–[3] has led to the standardisation of transfer learning techniques such as model pre-training and embedding of words and sentences in modern AI production systems. However it is not clear to what extent does pre-training and embedding help with training faster and more efficient classifiers on a typical downstream classification task. This work is aimed at shedding light on the cost-benefits trade-off of deploying machine learning models in an AI production system using accelerated computing resources [4].

We compare the efficiency of a simple logistic regression classifier to that of a more computationally intensive recurrent neural network classifier. We describe our procedure for selecting optimal configurations in each case, and discuss the statistical robustness of our results. Finally, we introduce a comparison with Human performance.

II. Anonymised customer care queries

The dataset used in this study is composed of anonymised customer messages received through the ASOS.com online chat at the time a new query is opened. Each message is associated with a binary label where 1 represents a *"Where is my order?"* query and 0 represents any other intent. Both the message and the label have been provided by the customers when opening a query with customer care service. Our aim is to study how efficient a fast classifier can be at predicting WISMO queries by training on a vast sample of labelled customer queries data.

A. Collection and data preparation

The majority of the messages are short with 2% of all messages containing more than 100 words in the training set and 3% in the test set as shown in Figure 1. The median length of messages in the training dataset is 21 words and in the test set 33 words. All messages have been cleaned from personally identifiable information including customer identifiers, order identifiers, email addresses, names and postcodes, each time replacing the personally identifiable content with a tag indicating which type of information was removed. Non-English messages have been removed from the collection by applying an automatic language detection tool [5]. We then manually reviewed a sample of the data to check that only English messages were left for analysis. To ensure that all messages provide sufficient information to convey an intent clearly, we also removed every message shorter than three words from the collection. After removing duplicates in the data by indexing each message by the conversation identifier, residual duplicate content was found to occur only to a negligible extent, with largest repeated occurrence amounting to 0.1% of all messages.

B. Data splits and label cleaning

Following anonymisation and data preparation steps, the training set was composed of 91,189 messages and the test set was composed of 1,138 messages held out for evaluation and not seen during training. The number of WISMO queries in the data amounts to 32% of all messages in the training set and 20% of all messages in the test set. Since we expect the customer labels to contain substantial level of noise due to their nature, we also asked a group of expert advisers to carefully re-label our test sample to compare with the customer annotations. We found that original labels provided by the customers predict the set of denoised labels from the group of expert advisers with 93% Precision, and 85% Recall giving us an estimate of Human annotator performance of 0.89 F1 on this task.

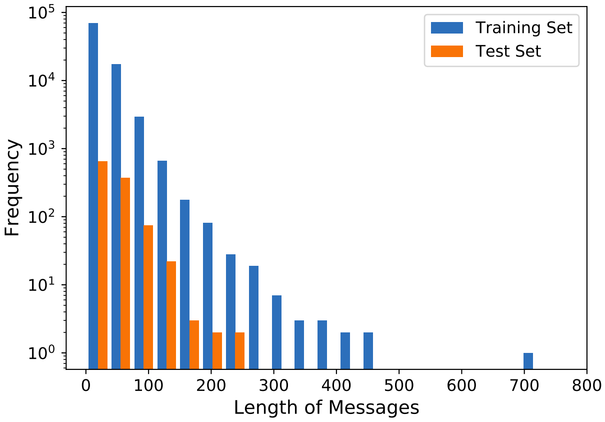

Fig. 1. Distribution of message length in the training set (blue) and in the test set (orange) where length is measured as the sum of the number of words in a message.

III. Linear n-grams classifiers

Linear n-grams models are widely employed for their effectiveness in many NLP tasks such as detecting the language of a document [6], as well as identifying the topic of that document [7], or sentiment analysis [8].

N-grams of higher orders allow the classification model to account for fine sequential patterns in each input message. In our experiments we considered n-gram sequences up to order seven, see Section V-B. All sentences in the training set are first tokenised and a word index is built by extracting all sequences at different n-gram order. Using the binary encoding scheme we represent each sentence as a vector of length $|V|$ where we set to one each entry in the index corresponding to a word found in the sentence, and to zero every other entry index. On Figure 2 we have plotted the distribution of words and n-grams in the training data and in the test data. We observe that only a few words/n-grams appear very often whereas most words/n-grams appear very few times as one should expect since the distribution of word frequencies in a corpus of documents is known to follow Zipf's law [9]. In solid agreement in both sets we notice that unigram frequencies follow an equivalent Zipf-Mandelbrot distribution that accounts for the peculiarities of function words in the low ranks of the distribution [10]. In the training data we observed that function words and action verbs used in the past tense dominate top frequent unigrams: in this corpus of transcript conversational data we expect function words to occur naturally as the authors relate the chains of events that led them to open a new query. As observed in the literature, n-grams of higher orders better align with the straight line pattern one should expect under the original Zipf law [11].

Given a binary encoded sentence on the n-gram index, the logistic regression classifier outputs a probability used to threshold the decisions of the classifier between the signal class *("Where is my order?")* and the background class *(Every other intent)*. In this study the threshold was taken to be 0.5 during development. We then calibrate all final models to

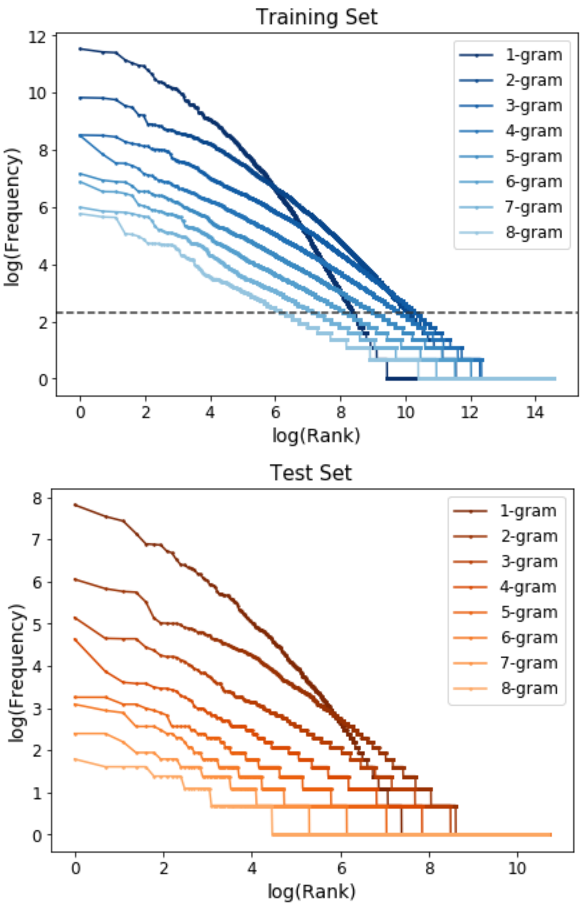

Fig. 2. Log of word frequency against log of word rank for all words appearing in the training set (blue shades - top panel) and the test set (orange shades - bottom panel). The x-axis is sorted in increasing order of word rank. The grey dashed line in the top panel represents the frequency cutoff applied set to 10.

control for a given precision level, see Section VI-C for a discussion. The total number of parameters in the linear model, n_{linear}, scales with the total size of the word index, accounting for the bias term this amounts to:

$$n_{\text{linear}} = |V| + 1.$$

IV. Recurrent neural network classifiers

A recurrent neural network learns to extract n-grams while allowing skips when a positional word is not informative when forming a discriminative sequential pattern. In this work we make use of lstm (Long short-term memory) cells which have the capacity to handle long-term dependencies and provide a solution to the vanishing gradient problem [12]. This is achieved by a gating mechanism that allows the neural networks to learn sequential patterns while also learning strategical skips when forming a hidden representation of each training message. An lstm cell is composed of four gates updated recursively across the sequence of words, with the update equations at word position t being:

$$\begin{aligned}
i_t &= \sigma[W_{xi}x_t + W_{hi}h_{t-1} + b_i] \\
g_t &= \tanh[W_{xg}x_t + W_{hh}h_{t-1} + b_g] \\
f_t &= \sigma[W_{xf}x_t + W_{hf}h_{t-1} + b_f] \\
o_t &= f_t \cdot c_{t-1} + i_t \cdot g_t \\
h_t &= o_t \cdot \tanh[c_t]
\end{aligned}$$

where σ denotes the sigmoid function, $\cdot$ denotes the Hadamard product, i_t is the input gate, g_t the input modulation gate, f_t the forget gate, o_t the output gate, c_t the memory state of each cell, x_t are the coordinates of the embedded word at the t^{th} position of the sentence, and h_t is the final memory state of the lstm cell at that position. W are matrices of internal weights and b are trainable bias parameters specific for each gate. The word embedding layer is a lookup table mapping a word index entry to the corresponding word vector representation (x_t as above). Each input token is associated with a set of coordinates in a latent vector space resolved by predicting word contexts from vast amounts of pre-training data, or resolved by training them jointly with all other network parameters on the given task. Pre-trained coordinates are often used to avoid the substantial overhead in number of model parameters (e.g millions of model parameters) and corresponding model complexity. The GloVe coordinates (Global Vectors for Word Representation) [13] are pre-trained on corpora larger in volume than any typical labelled dataset available on a downstream task. As a result of pre-training at scale, they are expected to correctly associate words with similar meanings [13] and help the classifier with encoded common sense information. The benefit of using pre-trained vectors is evident if the information they contain can successfully transfer to the task at hand. In this study we experiment with GloVe vectors pre-trained on the 2014 Wikipedia snapshot for different sizes of word vectors ranging between 50 and 300 dimensions. The total number of parameters in glove-lstm is given by:

$$n_{\text{lstm}} = 4(n_c n_d + n_d^2 + n_d) + n_d + 1$$

where n_c denotes the number of coordinates used to encode each word and n_d the size of the lstm memory. Now jointly training the continuous word representation layer incurs a $|V|n_c$ additional cost to n_{lstm}. Denoting by T the maximum length of the input message, this set of parameters is shared across all T word positions neural network, being updated recursively from left to right at each word step.

V. Comparative experiments

We conduct an extensive grid search across the hyperparameter space, where two classes of lstm classifiers (glove-lstm and plain-lstm) are compared with a faster linear n-grams classifiers. The glove-lstm model uses GloVe coordinates while the plain-lstm trains randomly initialised coordinates on the classification task.

A. Evaluation metrics

We monitor both the test performance of a classifier and the time it takes to train the model. We train all models on the same training set and evaluate their performance on the same test set to ensure that our comparison highlights the difference in model architecture and training configuration. Reported times are indicative and dependent on the implementation settings. In our comparisons we make use of Tensorflow 1.12 [14], Python 3.6 [15], and the multiclass softmax function to implement the logistic regression step in all model architectures. Using the equivalent binary sigmoid function reduces by half the number of parameters and processing times for the n-grams models, but not for the lstm models (see Section III and Section IV). We emphasise the difference in architecture by reporting the number of parameters and training times per epoch when using a binary sigmoid activation function. We also want to stress that training times for the lstm can be improved by using a faster implementation on accelerated computing resources which makes an extensive use of parallel computing on the GPU [16]. Each tested model was designed to complete training in a budget of 20 epochs under reasonable times.

B. Grid search settings

In all experiments, we use simple pre-processing steps that are easily applicable to other languages. We used the Tweet tokenizer provided in the NLTK Natural Language Toolkit [17], delimiting words on every punctuation character, and discarding every word occurring less than 10 times in the training set. All models are tested against a batch size in the set $\{32, 64, 128, 256, 512, 1024\}$. We conduct an extensive search in the hyperparameter space as described in the remainder of this section.

a) Linear n-grams classifiers: The optimiser used to train the n-gram model is a simple mini-batch gradient descent solver with a decay factor of 0.01. Each model is trained for 20 epochs on a given configuration of the following grid search: learning-rate in $\{0.05, 0.10, 0.50, 1.00\}$, regularisation-value in $\{1\text{e}{-6}, 1\text{e}{-5}, 1\text{e}{-4}, 1\text{e}{-3}\}$ and a maximum n-grams order between one to seven.

b) Lstm classifiers: We train two lstm models with their difference being that one makes use of fixed pre-trained word coordinates (glove-lstm), while the other jointly trains all neural layers on the classification task (plain-lstm). Both models are trained using Adam optimiser, a learning rate of 0.001 and a regularisation value of 0.001. A decay factor was used for glove-lstm as its performance on the train set and on the test were not found to strongly associate ($\rho = 0.59$) across configurations. After applying a decay factor of 0.01 we observed improved association between train and test performance ($\rho = 0.99$). Each glove-lstm model is train in a given configuration of the following grid search: memory-size in $\{1, 16, 32, 64, 128, 256\}$, embedding-size in $\{50\text{d}, 100\text{d}, 200\text{d}, 300\text{d}\}$ and max-input-length in $\{20, 30, 50, 100, 200\}$. The plain-lstm models with a fixed input length of 100 word positions, a memory size in $\{1, 16\}$, and an embedding size in $\{1, 16, 32, 64, 128, 256\}$.

C. Model selection procedure

Since any best performing configuration on the test set could be a random fluctuation due to the stochastic nature of the descent algorithm used to train these models, before selecting a top performing model we conducted replication trials by re-training the same model on the same data 100 times and re-calculating their performance after each trial. A top performing configuration was discarded if the replication trials did not yield similar or greater test F1 levels as observed during the grid search. This situation can lead to critical bottlenecks in a production environment where these models are re-trained regularly (e.g weekly) on recent data. Our procedure ensures that each selected model can be reproduced to similar levels of performance in a reasonable number of trials. We also leverage the trials statistics to conduct a proper statistical test for comparing between n-grams, glove-lstm and plain-lstm. Finally we introduce and discuss the best realised performance of these models on our denoised set of labels.

VI. Results and discussion

A. Grid search results

a) Comparison of selected configurations: The top performing plain-lstm configuration was reproduced 5% of the times in the replication trials while for the glove-lstm this was 2% of the times. For the n-grams the configuration that passed the replication trials was the second top performing model with 2% of success. Interestingly, the only difference between the two top performing n-grams configurations was the regularisation value which was found to be $1\text{e}{-5}$ and $1\text{e}{-4}$ respectively. We therefore safely selected the second n-gram configuration with the understanding that higher regularisation improves the generalisation ability of a classifier.

In summary all identified optimal classifiers were as follow: the n-grams classifier used a maximum of 3-grams, with a learning rate of 1.0, a regularisation value of $1\text{e}{-4}$ and a batch size of 32. The plain-lstm classifier used embedding vectors of dimension 256, with a memory size of 16 and batch size of 128. The glove-lstm classifier used GloVe coordinates of dimension 200, with a memory size of 256, a batch size of 512 and an maximum message length of 200 word positions.

We indicate the performance of each top performing classifier in Table I and provide the summary statistics of the replication trials in Table II. First, we observe that the n-grams model consistently outperformed the glove-lstm model in both the grid search runs and in the replication trials. Since the GloVe vectors were found to cover 94% of all words in the training set, it is unlikely that the better performance of the n-grams model is solely due to the fact that it fully accounts for arbitrary vocabulary. It is also unlikely that glove-lstm is limited by only processing the first 200 word positions in a message, as most messages in the test set are shorter than 100 words (see Figure 1). We rather observed that most of the signal was concentrated in the first word positions of each

	Precision	**Recall**	**F1**
plain-lstm	**0.817**	**0.882**	**0.848**
n-grams	0.816	0.855	0.835
glove-lstm	0.789	0.868	0.827

TABLE I
TEST PERFORMANCE FOR THE OPTIMAL CONFIGURATIONS FOUND BY GRID SEARCH.

	$\mathbf{F1}^{\mathbf{Med}}_{\mathbf{test}}$	$\mathbf{F1}^{\mathbf{IQR}}_{\mathbf{test}}$	$\mathbf{F1}^{\mathbf{Min}}_{\mathbf{test}}$	$\mathbf{F1}^{\mathbf{Max}}_{\mathbf{test}}$	$\mathbf{F1}^{\mathbf{Max}}_{\mathbf{denoised}}$
Human	n/a	n/a	n/a	n/a	0.889
plain-lstm	**0.831**	0.010	0.811	**0.854**	**0.887**
n-grams	0.828	**0.006**	**0.817**	0.835	0.865
glove-lstm	0.815	0.012	0.784	0.834	0.856

TABLE II
F1 SCORES FOR THE OPTIMAL CONFIGURATIONS IDENTIFIED BY GRID SEARCH. THE MEDIAN, INTERQUARTILE RANGE (IQR), MIN AND MAX QUANTITIES CORRESPOND WITH RE-TRAINING EACH CONFIGURATION IN 100 TRIALS. THE LAST COLUMN IS THE MAXIMUM F1 SCORE OBTAINED ON A VERSION OF THE TEST SET RE-LABELLED BY A GROUP OF EXPERT ANNOTATORS.

message, with top performing glove-lstm configurations also showing shorter input lengths of 100 positions or even 50 positions. The statistical robustness of the above observations are tested in Section VI-B.

b) Trade-off between accuracy and processing times: The information contained in pre-trained GloVe vectors does not seem amenable to training a top performing classifier on this downstream task, whereas the word coordinates resolved from the training data perform the best in conjunction with the lstm layer. Interestingly, glove-lstm processes nearly three times as much data as the plain-lstm configuration at equivalent input length. This difference in speed is directly associated with the batch size, suggesting that maintaining a fixed word representation layer set to GloVe coordinates may help glove-lstm to generalise better at larger batch size.

Linear n-grams were the second top performing model. They showed the highest worst-case performance and lowest IQR across replication trials. This model is relatively simple compared to the top performing plain-lstm model, fitting two orders of magnitude less weights on the training data. This could explain the remarkable performance stability observed during the trials, it could also be related to the nature of the discriminative features used to support a class decision: extracted at the global sentence level for the n-grams, as opposed to read off sequentially from left to right for the lstm.

We observe that n-grams models train under 10 minutes (26s per epoch) owing to their small size overall (see Figure 3). While it takes almost five times longer to train the top performing recurrent neural model on a GPU, whether the gain in F1 performance is worth the extra costs of training a recurrent neural classifier depends on the availability of accelerated computing resources to train and deploy these classifiers.

B. Statistical significance

We tested whether the linear n-grams classifier tends to outperform glove-lstm, while being inexpensive to train. A

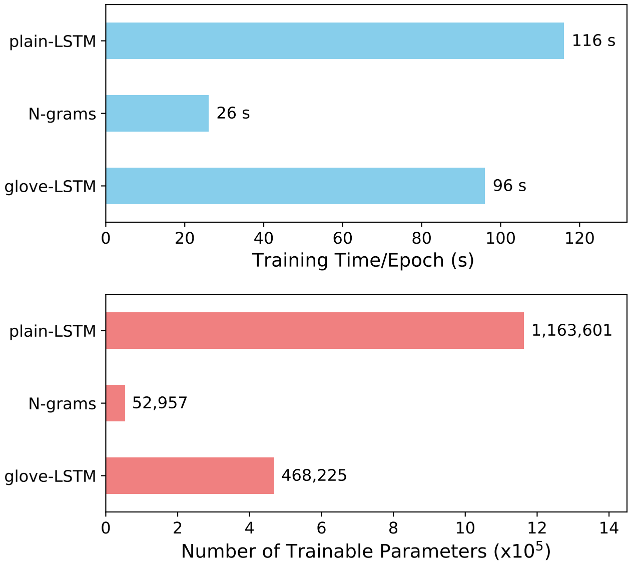

Fig. 3. Training time per epoch in seconds (top panel) and number of trainable parameters (bottom panel) for the three models.

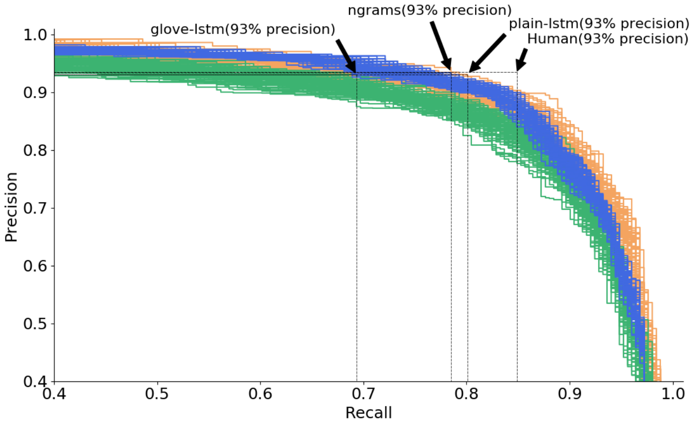

Fig. 4. Precision-recall curves associated with each replication run of all optimal configurations: glove-lstm (green), n-grams (blue), plain-lstm(orange).

Mann-Whitney U test indicated that test F1 for linear n-grams classifiers was statistically greater than for the glove-lstm classifier (M1$=0.828$, M2$=0.815$; U$=1,196$, $P=1.46\mathrm{e}{-20}$) where M denotes the median F1 score across 100 trials for each model and P the test's p-value). Now training all neural layers including the continuous word representation layer (i.e. the plain-lstm model), a Mann-Whitney U test indicated superior performance for the neural network model (M1$=0.831$, M2$=0.828$; U$=4,280.0$, $P=8.24\mathrm{e}{-4}$).

C. Calibration analysis on denoised labels

Before deploying these models in an AI production environment, we wish to control their level of precision to limit the number of irrelevant messages identified. We first asked a group of expert advisers to review the 1,138 labels initially provided in the test data, by showing them both the message and the corresponding label assigned by the author.

This procedure allowed us to access Human performance on the task of annotating the correct intent, see last column of Table II. The discrepancies in the original labels could be attributed to the users being pressed when opening a new query. Also expert advisers received extensive training on identifying the correct intent from short user messages. The original labels were found to predict the new set of denoised labels provided by the group of expert advisers with 0.93 Precision and 0.85 Recall giving us an estimate of Human performance of 0.889 F1 on this task. Remarkably, the plain-lstm achieved similar level of performance with 0.887 F1, followed by the n-grams model with 0.865 F1, and the glove-lstm model 0.856 F1. To control for precision level above that of users', we then conducted a systematic analysis as follows: the classification layer in each of the three model architectures has parameters $w \in \mathbb{R}^{|V|}$ and $b \in \mathbb{R}$, such that $P(y = 1|x) = \sigma[wx + b] = [1 + \exp{(-wx - b)}]^{-1}$. Given a decision threshold ϵ the classifier returns a binary prediction $\hat{y} = \{P(y = 1|x) > \epsilon\}_{1,0} \in \{0, 1\}$ for a training example $x \in \mathbb{R}^{|V|}$ associated with a label $y \in \{1, 0\}$. Here we calibrate the predictions $\hat{y}$'s of a classifier by tuning the threshold function for different values of ϵ in the unit interval, and re-calculating the performance in each individual replication trial, see Figure 4. Following our procedure we identified the trials which maximise recall for all three model architectures after controlling for precision level above Human performance. This was achieved with corresponding 0.80 Recall for the plain-lstm, 0.78 Recall for the n-grams, and 0.69 Recall for glove-lstm, meaning plain-lstm can address 80% of incoming *"Where is my Order?"* queries for a corresponding error rate below the level of noise produced by Human authors.

VII. Conclusion

We studied the design of a user intent classifier aimed at detecting the top most popular intent received in a customer care service, namely *"Where is my order?"*. Two different model architectures were compared, a logistic regression classifier combined with n-grams, and more computationally intensive recurrent neural network classifiers, using either fixed pre-trained word coordinates (glove-lstm) or training these coordinates jointly with all other network parameters (plain-lstm). A grid search was performed to identify a top performing configuration in each three cases, accompanied by replication trials to assess their robustness. While the linear n-grams model was found to outperform glove-lstm in a Mann-Whitney U test to test for significant difference in medians (M1= 0.828, M2= 0.815; U= $1,196$, $P = 1.46\text{e}{-20}$) the plain-lstm was found as the best performing classifier overall (M1= 0.831, M2= 0.828, U= $4,280.0$, P= 8.24e−4). On a set of denoised labels, this classifier was found to match Human performance (0.89 F1). A calibration analysis revealed that linear n-grams achieved interesting levels of Recall at equivalent Human Precision, with a 0.07 difference while for plain-lstm this was 0.05. Our findings indicate that linear n-grams models offer appealing levels of accuracy for user intent classification while being inexpensive to train. If the necessity for higher accuracy significantly outweighs the cost of longer training times, then in order to observe the expected gain it was necessary to update all neural layers including re-training the word representation layer on GPU. Future works will consist in exploring larger models, comparing with our current set of observations both in terms of accuracy and processing times.

VIII. Acknowledgements

We are thankful to Adam Jelley and Hamzah Hussian for their contributions to the preliminary experiments and useful discussions. We thank Naeem Khedarun, lead Engineer of ASOS.com ConversationalAI for his assistance and expertise in deploying the grid search experimentation on GPU servers. We also thank Ben Chamberlain for managing ASOS.com CDT program with UCL and Gabriel Facini for his insightful comments on an earlier version of this document. CN was supported by the STFC UCL Centre for Doctoral Training in Data Intensive Science (grant number ST/P006736/1).

References

[1] A. Vaswani *et al.*, "Attention Is All You Need," *arXiv e-prints*, p. arXiv:1706.03762, Jun 2017.

[2] A. Radford *et al.*, "Language models are unsupervised multitask learners," 2019.

[3] J. Devlin *et al.*, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv e-prints*, Oct 2018.

[4] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," 2019.

[5] N. Shuyo, "Language detection library for java," 2010. [Online]. Available: http://code.google.com/p/language-detection/

[6] H. Sababa and A. Stassopoulou, "A classifier to distinguish between cypriot greek and standard modern greek," in *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE, 2018, pp. 251–255.

[7] Y. Shao *et al.*, "Clinical text classification with word embedding features vs. bag-of-words features," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 2874–2878.

[8] N. Cummins *et al.*, "Multimodal bag-of-words for cross domains sentiment analysis."

[9] G. K. Zipf, *Selected Studies of the Principle of Relative Frequency in Language*. Harvard University Press, 1932.

[10] M. A. Montemurro, "Beyond the zipf–mandelbrot law in quantitative linguistics," *Physica A: Statistical Mechanics and its Applications*, vol. 300, no. 3-4, pp. 567–578, 2001.

[11] L. Q. Ha *et al.*, "Extension of Zipf's law to word and character n-grams for English and Chinese."

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[13] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[14] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[15] "Python software foundation. python language reference." [Online]. Available: http://www.python.org

[16] J. Appleyard, T. Kocisky, and P. Blunsom, "Optimizing performance of recurrent neural networks on gpus," *arXiv preprint*, 2016.

[17] E. Loper and S. Bird, "Nltk: the natural language toolkit," *arXiv preprint cs/0205028*, 2002.