frontiers in Applied Mathematics and Statistics

METHODS
published: 26 August 2021
doi: 10.3389/fams.2021.604842

AVA: A Financial Service Chatbot Based on Deep Bidirectional Transformers

Shi Yu, Yuxin Chen and Hussain Zaidi*

The Vanguard Group, Malvern, PA, United States

We develop a chatbot using deep bidirectional transformer (BERT) models to handle client questions in financial investment customer service. The bot can recognize 381 intents, decides when to say I don't know, and escalate escalation/uncertain questions to human operators. Our main novel contribution is the discussion about the uncertainty measure for BERT, where three different approaches are systematically compared with real problems. We investigated two uncertainty metrics, information entropy and variance of dropout sampling, in BERT, followed by mixed-integer programming to optimize decision thresholds. Another novel contribution is the usage of BERT as a language model in automatic spelling correction. Inputs with accidental spelling errors can significantly decrease intent classification performance. The proposed approach combines probabilities from masked language model and word edit distances to find the best corrections for misspelled words. The chatbot and the entire conversational AI system are developed using open-source tools and deployed within our company's intranet. The proposed approach can be useful for industries seeking similar in-house solutions in their specific business domains. We share all our code and a sample chatbot built on a public data set on GitHub.

Keywords: chabot, BERT, rasa, bayesian learning, intent classification

OPEN ACCESS

Edited by:
Glenn Fung,
American Family Insurance,
United States

Reviewed by:
Qian You,
University of Wisconsin-Madison,
United States
Amin Salighehdar,
Stevens Institute of Technology,
United States

****Correspondence:***
Shi Yu
shi.yu@hotmail.com

Specialty section:
This article was submitted to Mathematical Finance, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *10 September 2020*
Accepted: *12 May 2021*
Published: *26 August 2021*

Citation:
Yu S, Chen Y and Zaidi H (2021) AVA: A Financial Service Chatbot Based on Deep Bidirectional Transformers. Front. Appl. Math. Stat. 7:604842. doi: 10.3389/fams.2021.604842

1 INTRODUCTION

Since their first appearances decades ago [1–3], chatbots have always been marking the apex of artificial intelligence as forefront of all major AI revolutions, such as human–computer interaction, knowledge engineering, expert system, natural language processing, natural language understanding, deep learning, and many others. Open-domain chatbots, also known as *chitchat* bots, can mimic human conversations to the greatest extent in topics of almost any kind, thus are widely engaged for socialization, entertainment, emotional companionship, and marketing. Earlier generations of open-domain bots, such as those mentioned in Ref [3, 4], relied heavily on hand-crafted rules and recursive symbolic evaluations to capture the key elements of human-like conversation. New advances in this field are mostly data-driven and end-to-end systems based on statistical models and neural conversational models [5] aim to achieve human-like conversations through a more scalable and adaptable learning process on free-form and large data sets [5], such as those given in Ref [6–9] and [10].

Unlike open-domain bots, closed-domain chatbots are designed to transform existing processes that rely on human agents. Their goals are to help users accomplish specific tasks, where typical examples range from order placement to customer support; therefore, they are also known as *task-oriented bots*

[5]. Many businesses are excited about the prospect of using closed-domain chatbots to interact directly with their customer base, which comes with many benefits such as cost reduction, zero downtime, or no prejudices. However, there will always be instances where a bot will need a human's input for new scenarios. This could be a customer presenting a problem it has never expected for [11], attempting to respond to a naughty input, or even something as simple as incorrect spelling. Under these scenarios, expected responses from open-domain and closed-domain chatbots can be very different: a successful open-domain bot should be "*knowledgeable, humorous, and addictive*," whereas a closed-domain chatbot ought to be "*accurate, reliable, and efficient*." One main difference is the way of handling unknown questions. A chitchat bot would respond with an adversarial question such as *Why do you ask this?* and keep the conversation going and deviate back to the topics under its coverage [12]. A user may find the chatbot is out-smarting, but not very helpful in solving problems. In contrast, a task-oriented bot is scoped to a specific domain of intents and should terminate out-of-scope conversations promptly and escalate them to human agents.

This article presents AVA (a Vanguard assistant), a task-oriented chatbot supporting phone call agents when they interact with clients on live calls. Traditionally, when phone agents need help, they put client calls on hold and consult experts in a support group. With a chatbot, our goal is to transform the consultation processes between phone agents and experts to an end-to-end conversational AI system. Our focus is to significantly reduce operating costs by reducing the call holding time and the need of experts, while transforming our client experience in a way that eventually promotes client self-provisioning in a controlled environment. Understanding intents correctly and escalating escalation intents promptly are key to its success. Recently, the NLP community has made many breakthroughs in context-dependent embeddings and bidirectional language models like ELMo, OpenAI, GPT, BERT, RoBERTa, DistilBERT, XLM, and XLNet [1, 13–21]. In particular, the BERT model [1] has become a new NLP baseline including sentence classification, question answering, named-entity recognition and many others. To our knowledge, there are few measures that address prediction uncertainties in these sophisticated deep learning structures, or explain how to achieve optimal decisions on observed uncertainty measures. The off-the-shelf softmax outputs of these models are predictive probabilities, and they are not a valid measure for the confidence in a network's predictions [22–25], which are important concerns in real-world applications [11].

Our main contribution in this study is applying advances in Bayesian deep learning to quantify uncertainties in BERT intent predictions. Formal methods like stochastic gradient (SG)-MCMC [23, 26–30] and variational inference (VI) [22, 31–33] extensively discussed in the literature may require modifying the network. In conventional neural networks, the parameters are estimated by a single point value obtained using backpropagation with stochastic gradient descent (SGD), whereas Bayesian deep learning assumes a prior over model parameters and then data are used to compute a distribution over each of these parameters. However, for BNNs with thousands of parameters, computing the posterior is intractable due to the complexity in computing the marginal likelihood [34]. SG-MCMC and VI methods propose two different solutions to address the aforementioned complexity. SG-MCMC mitigates the need to compute gradients on full data set by using mini-batches for gradient computation, which enables easier computation (with the same computational complexity as SGD), but still lacks the ability to capture complex distributions in the parameter space. VI performs Bayesian inference by using a computationally tractable "variational" distribution $q(\theta)$ to approximate the posterior, and the capacity of uncertainty representation is limited by the variational distribution. Re-implementation of the entire BERT model for Bayesian inference is a non-trivial task, so here we took the Monte Carlo dropout (MCD) approach [22] to approximate variational inference, whereby dropout is performed at training and test time, using multiple dropout masks. Our dropout experiments are compared with two other approaches (entropy and dummy class), and the final implementation is determined among the trade-off between accuracy and efficiency. Recently, similar MCD dropout approach has been proposed for transformer models to calibrate speech detection outcomes [35].

We also investigate the usage of BERT as a language model to decipher spelling errors. Most vendor-based chatbot solutions embed an additional layer of service, where device-dependent error models and N-gram language models [36] are utilized for spell checking and language interpretation. At the representation layer, WordPiece model [37] and byte pair rncoding (BPE) model [38, 39] are common techniques to segment words into smaller units; thus, similarities at the sub-word level can be captured by NLP models and generalized on out-of-vocabulary (OOV) words. Our approach combines efforts of both sides: words corrected by the proposed language model are further tokenized by the WordPiece model to match pretrained embeddings in BERT learning.

Despite all advances of chatbots, industries like finance and health care are concerned about cyber security because of the large amount of sensitive information entered during chatbot sessions. Task-oriented bots often require access to critical internal systems and confidential data to finish specific tasks. Therefore, 100% on-premise solutions that enable full customization, monitoring, and smooth integration are preferable than cloud solutions. In this study, the proposed chatbot is designed using RASA open-source version and deployed within our enterprise intranet. Using RASA's conversational design, we hybridize RASA's chitchat module with the proposed task-oriented conversational systems developed on Python, TensorFlow, and PyTorch. We believe our approach can provide some useful guidance for industries to contemplate adopting chatbot solutions in their business domains.

2 BACKGROUND

Recent breakthroughs in NLP research are driven by two intertwined directions: Advances in distributed representations, sparked by the success of word embeddings [40, 41], character embeddings [42–44], and contextualized word embeddings [1, 19, 45], have successfully tackled the curse of dimensionality in modeling complex language models. Advances of neural network architecture, represented by CNN [46–48], attention mechanism [49], and transformer as the seq2seq model with parallelized attentions [50], have defined the new state-of-the-art deep learning models for NLP.

Principled uncertainty estimation in regression [51], reinforcement learning [52], and classification [53] are active areas of research with a large volume of work. The theory of Bayesian neural networks [54, 55] provides the tools and techniques to understand model uncertainty, but these techniques come with significant computational costs as they double the number of parameters to be trained. The authors of Ref [22] showed that a neural network with dropout turned on at test time is equivalent to a deep Gaussian process, and we can obtain model uncertainty estimates from such a network by multiple-sampling the predictions of the network at test time. Non-Bayesian approaches to estimate the uncertainty are also shown to produce reliable uncertainty estimates [56]; our focus in this study is on Bayesian approaches. In classification tasks, the uncertainty obtained from multiple sampling at test time is an estimate of the confidence in the predictions similar to the entropy of the predictions. In this study, we compare the threshold for escalating a query to a human operator using model uncertainty obtained from dropout-based chatbot against setting the threshold using the entropy of the predictions. We choose dropout-based Bayesian approximation because it does not require changes to the model architecture, does not add parameters to train, and does not change the training process as compared to other Bayesian approaches. We minimize noise in the data by employing spelling correction models before classifying the input. Further, the labels for the user queries are human-curated with minimal error. Hence, our focus is on quantifying epistemic uncertainty in AVA, rather than aleatoric uncertainty [57]. We use mixed-integer optimization to find a threshold for human escalation of a user query based on the mean prediction and the uncertainty of the prediction. This optimization step, once again, does not require modifications to the network architecture and can be implemented separately from model training. In other contexts, it might be fruitful to have an integrated escalation option in the neural network [58], and we leave the trade-offs of integrated reject option and non-Bayesian approaches for future work.

Similar approaches in spelling correction, besides those mentioned in **Section 1**, are reported in Deep Text Corrector [59] that applies a seq2seq model to automatically correct small grammatical errors in conversational written English. Optimal decision threshold learning under uncertainty is studied in Ref [60] as reinforcement learning and iterative Bayesian optimization formulations.

3 SYSTEM OVERVIEW AND DATA SETS

3.1 Overview of the System

Figure 1 illustrates system overview of AVA. The proposed conversational AI will gradually replace the traditional human–human interactions between phone agents and internal experts and eventually allow clients self-provisioning interaction directly to the AI system. Now, phone agents interact with AVA chatbots deployed on Microsoft Teams in our company intranet, and their questions are preprocessed by a sentence completion model (introduced in **Section 6**) to correct misspellings. Then, inputs are classified by an intent classification model (**Sections 4**, **Sections 5**), where relevant questions are assigned predicted intent labels, and downstream information retrieval and questioning answering modules are triggered to extract answers from a document repository. Escalation questions are escalated to human experts following the decision thresholds optimized using methods introduced in **Section 5**. This article only discusses the intent classification model and the sentence completion model.

3.2 Data for Intent Classification Model

Training data for AVA's intent classification model is collected, curated, and generated by a dedicated business team from interaction logs between phone agents and the expert team. The whole process takes about one year to finish. In total, 22,630 questions are selected and classified to 381 intents, which compose the *relevant question set* for the intent classification model. Additionally, 17,395 questions are manually synthesized as *escalation questions*, and none of them belongs to any of the aforementioned 381 intents. Each *relevant question* is hierarchically assigned with three labels from Tier 1 to Tier 3. In this hierarchy, there are five unique Tier-1 labels, 107 Tier-2 labels, and 381 Tier-3 labels. Our intent classification model is designed to classify relevant input questions into 381 Tier-3 intents and then trigger downstream models to extract appropriate responses. The five Tier-1 labels and the numbers of intents included in each label are *account maintenance* (9,074), *account permissions* (2,961), *transfer of assets* (2,838), *banking* (4,788), *tax FAQ* (2,969). At Tier-1, general business issues across intents are very different, but at the Tier-3 level, questions are quite similar to each other, where differences are merely at the specific responses. Escalation questions, compared to relevant questions, have two main characteristics:

- Some questions are relevant to business intents but unsuitable to be processed by conversational AI. For example, in **Table 1**, question "*How can we get into an account with only one security question?*" is related to *call authentication* in *account permission*, but its response needs further human diagnosis to collect more information. These types of questions should be escalated to human experts.
- Out-of-scope questions. For example, questions like "*What is the best place to learn about Vanguard's investment philosophy?*" or "*What is a hippopotamus?*" are totally outside the scope of our training data, but they may still occur in real-world interactions.

3.3 Textual Data for Pretrained Embeddings and Sentence Completion Model

Inspired by the progress in computer vision, transfer learning has been very successful in NLP community and has become a common practice. Initializing deep neural network with pretrained embeddings and fine-tuning the models toward

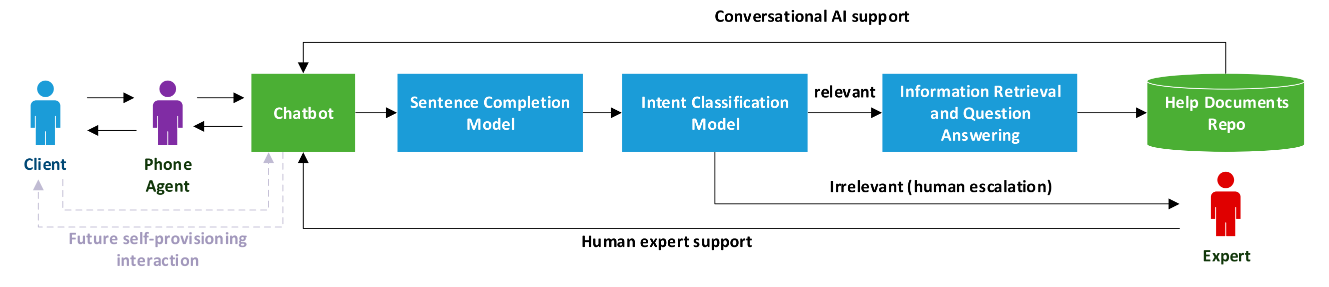
Conversational AI support
Client
Phone Agent
Chatbot
Sentence Completion Model
Intent Classification Model
relevant
Information Retrieval and Question Answering
Help Documents Repo
Irrelevant (human escalation)
Expert
Human expert support
Future self-provisioning interaction

FIGURE 1 | End-to-end conceptual diagram of AVA.

TABLE 1 | Example questions used in AVA intent classification model training.

T1 label	T2 label	T3 label	Questions
Account maintenance	Call authentication	Type 2	Am I allowed to give the client their social security number?
	Call authentication	Type 5	Do the web security questions need to be reset by the client if their web access is blocked?
	Web reset	Type 1	How many security questions are required to be asked to reset a client's web security questions?
Account permission	Call authentication	Type 2	How are the web security questions used to authenticate a client?
	Agent incapactiated	Type 3	Is it possible to set up agent certification for an incapacitated person on an individual Roth 401 k?
TAX FAQ	Miscellaneous	What is	Do I need my social security number on the 1099MISC form?
Transfer of asset	Unlike registrations	Type 2	Does the client need to provide special documentation if they want to transfer from one account to another account?
	Brokerage transfer	Type 3	Is there a list of items that need to be included on a statement to transfer an account?
Banking	Add owner	Type 4	Once a bank has been declined how can we authorize it?
	Add/change/delete	Type 3	Does a limited agent have authorization to adjust bank info?
Escalation	–	–	How can we get into an account with only one security question?
	–	–	Am I able to use my Roth IRA to set up a margin account?
	–	–	What is the best place to learn about Vanguard's investment philosophy?

TABLE 2 | Comparison of intent classification performance. BERT and XLNet models were all trained for 30 epochs using batch size 16.

Model	Performance
BERT small + SharePoint embeddings	0.944
BERT small + Google embeddings	0.949
BERT large + Google embeddings	0.954
XLNet large + Google embeddings	0.927
LSTM with attention + Word2Vec	0.913
LSTM + Word2Vec	0.892
Logistic regression + TFIDF	0.820
Xgboost + TFIDF	0.760
Naive Bayes + TFIDF	0.661

task-specific data are proven methods in multitask NLP learning. In our approach, besides applying off-the-shelf embeddings from Google BERT and XLNet, we also pretrain BERT embeddings using our company's proprietary text to capture special semantic meanings of words in the financial domain. Three types of textual data sets are used for embeddings training:

- SharePoint text: About 3.2G bytes of corpora scraped from our company's internal SharePoint websites, including Web pages, Word documents, ppt slides, pdf documents, and notes from internal CRM systems.
- Emails: About 8G bytes of customer service emails are extracted.
- Phone call transcriptions: We apply AWS to transcribe 500 K client service phone calls, and the transcription text is used for training.

All embeddings are trained in case-insensitive settings. Attention and hidden layer dropout probabilities are set to 0.1, hidden size is 768, attention heads and hidden layers are set to 12, and vocabulary size is 32,000 using SentencePiece tokenizer. On AWS P3.2xlarge instance, each embeddings is trained for one million iterations and takes about one week CPU time to finish. More details about parameter selection for pretraining are available in the GitHub code. The same pretrained embeddings are used to initialize BERT model training in intent classification and also used as language models in sentence completion.

4 INTENT CLASSIFICATION PERFORMANCE ON RELEVANT QUESTIONS

Using only relevant questions, we compare various popular model architectures to find one with the best performance on 5-fold validation. Not surprisingly, BERT models generally

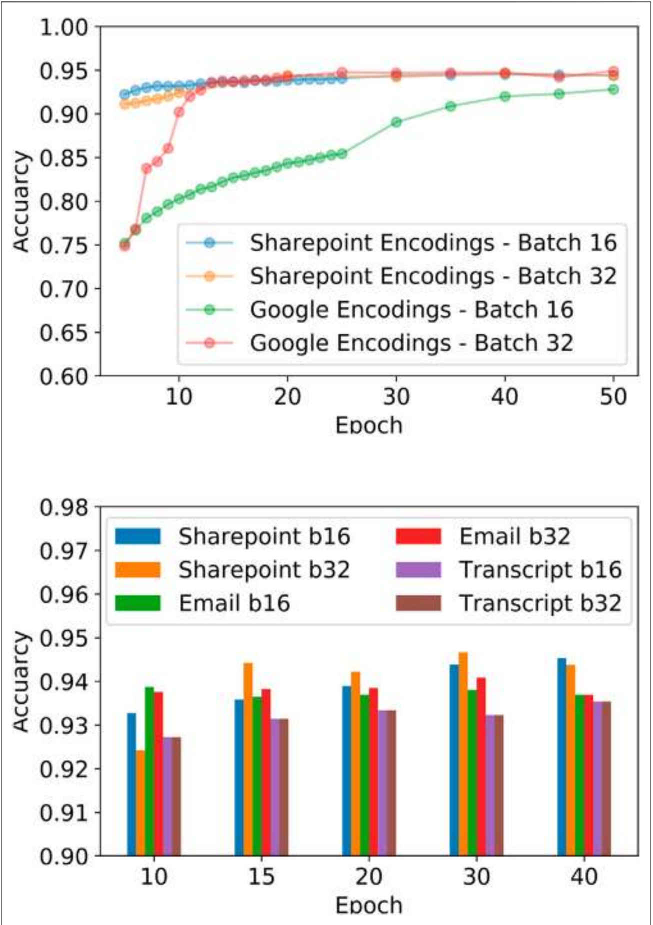

FIGURE 2 | Comparison of test set accuracy using different embeddings and batch sizes.

produce much better performance than other models (**Table 2**). Large BERT (24-layer, 1024-hidden, and 16-heads) has a slight improvement over small BERT (12-layer, 768-hidden, and 12-heads) but less preferred because of expensive computations. To our surprise, XLNet, a model reported outperforming BERT in multitask NLP, performs 2 percent lower on our data.

BERT models initialized by proprietary embeddings converge faster than those initialized by off-the-shelf embeddings (**Figure 2A**). And embeddings trained on company's SharePoint text perform better than those built on Emails and phone call transcriptions (**Figure 2B**). Using larger batch size 32) enables models to converge faster and leads to better performance.

5 INTENT CLASSIFICATION PERFORMANCE INCLUDING ESCALATION QUESTIONS

We have shown how the BERT model outperforms other models on real data sets that only contain relevant questions. The capability to handle 381 intents simultaneously at 94.5% accuracy makes it an ideal intent classifier candidate in a chatbot. This section describes how we quantify uncertainties on BERT predictions and enable the bot to detect escalation questions. Three approaches are compared:

- Predictive entropy: We measure uncertainty of predictions using Shannon entropy $H = -\sum_{k=1}^{K} p_{ik} \log p_{ik}$, where p_{ik} is the prediction probability of ith sample to kth class. Here, p_{ik} is softmax output of the BERT network [56]. A higher predictive entropy corresponds to a greater degree of uncertainty. Then, an optimally chosen cutoff threshold applied on entropies should be able to separate the majority of in-sample questions and escalation questions.
- Dropout: We apply Monte Carlo (MC) dropout by doing 100 Monte Carlo samples. At each inference iteration, a certain percent of the set of units drop out. This generates random predictions, which are interpreted as samples from a probabilistic distribution [22]. Since we do not employ regularization in our network, τ^{-1} in Eq. 7 in Ref [22] is effectively zero and the predictive variance is equal to the sample variance from stochastic passes. We could then investigate the distributions and interpret model uncertainty as mean probabilities and variances.
- Dummy class: We simply treat escalation questions as a dummy class to distinguish them from original questions. Unlike entropy and dropout, this approach requires retraining of BERT models on the expanded data set including dummy class questions.

5.1 Experimental Setup

All results mentioned in this section are obtained using BERT small + SharePoint embeddings (batch size 16). In entropy and dropout approaches, both relevant questions and escalation questions are split into five folds, where four folds (80%) of relevant questions are used to train the BERT model. Then, among that 20% held-out relevant questions, we further split them into five folds, where 80% of them (equal to 16% of the entire relevant question set) are combined with four folds of escalation questions to learn the optimal decision variables. The learned decision variables are applied on BERT predictions of the remaining 20% (906) of held-out relevant questions and held-out escalation questions (4,000), to obtain the test performance. In the dummy class approach, the BERT model is trained using four folds of relevant questions plus four folds of escalation questions and tested on the same amount of test questions as entropy and dropout approaches.

5.2 Optimizing Entropy Decision Threshold

To find the optimal threshold cutoff b, we consider the following quadratic mixed-integer programming problem

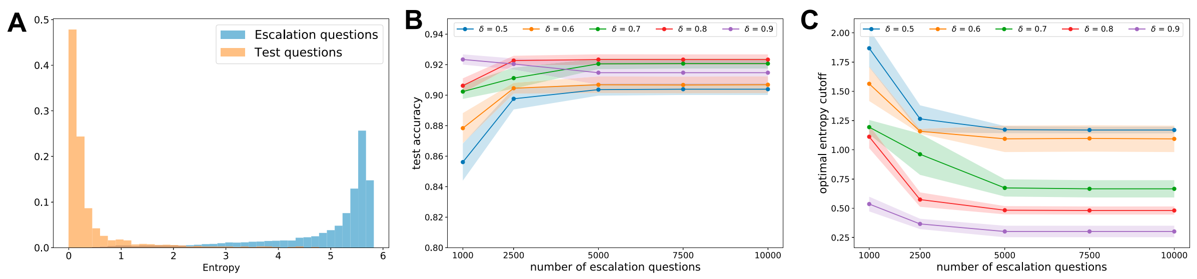

FIGURE 3 | Optimizing the entropy threshold to detect escalation questions. As shown in **(A)**, in-sample test questions and escalation questions have very different distributions of predictive entropies. Subfigure **(B)** shows how test accuracies, evaluated using decision variables b solved by **(1)** on BERT predictions on test data, change when different numbers of escalation questions are involved in training. Subfigure **(C)** shows the impact of δ on the optimized thresholds when the number of escalation questions increase optimization. Usually, to safeguard making wrong predictions in client-facing applications, δ is set to a value smaller than 1 because 1 means the cost of making wrong predictions is the same as spending human effort on a question. In contrast, a value 0.5 means the cost of wrong predictions is two times larger than human answering cost. Such a cost is guided by business reasons, and different δ could lead to different optimal thresholds.

$$\begin{array}{lll}\min\limits_{x,b} & \sum_{i,k}(x_{ik}-l_{ik})^2 & \\ s.t. & x_{ik}=0 & \textit{if } E_i\geq b, \text{ for } k \text{ in } 1,\ldots,K \\ & x_{ik}=1 & \textit{if } E_i\geq b, \text{ for } k=K+1 \\ & x_{ik}\in\{0,1\} & \\ & \sum_{k=1}^{K+1}x_{ik}=1 & \forall i \text{ in } 1,\ldots,N \\ & b\geq 0 & \end{array} \quad . \tag{1}$$

to minimize the quadratic loss between the predictive assignments x_{ik} and true labels l_{ik}. In **Eq. 1**, i is the sample index, k is class (intent) indices, x_{ik} is $N\times(K+1)$ binary matrix, and l_{ik} is also $N\times(K+1)$, where the first K columns are binary values and the last column is a uniform vector δ, which represents the cost of escalating questions. Normally, δ is a constant value smaller than 1, which encourages the bot to escalate questions, rather than making mistaken predictions. The first and second constraints of **Eq. 1** force an escalation label when entropy $E_i\geq b$. The third and fourth constraints restrict x_{ik} as binary variables and ensure the sum for each sample is 1. Experimental results (**Figure 3**) indicate that **Eq. 1** needs more than 5,000 escalation questions to learn a stabilized b. The value of escalation cost δ has a significant impact on the optimal b value and in our implementation is set to 0.5.

5.3 Monte Carlo Dropout

In the BERT model, dropout ratios can be customized at encoding, decoding, attention, and output layers. A combinatorial search for optimal dropout ratios is computationally challenging. Results reported in the article are obtained through simplifications with the same dropout ratio assigned and varied on all layers. Our MC dropout experiments are conducted as follows:

1. Change dropout ratios in encoding/decoding/attention/output layer of BERT
2. Train the BERT model on 80% of relevant questions for 10 or 30 epochs
3. Export and serve the trained model by TensorFlow serving
4. Repeat inference 100 times on questions, average the results per each question to obtain mean probabilities and standard deviations, and then average the deviations for a set of questions.

According to the experimental results illustrated in **Figure 4**, we make three conclusions: 1) Epistemic uncertainty estimated by MCD reflects question relevance: when inputs are similar to the training data, there will be low uncertainty, while data are different from the original, training data should have higher epistemic uncertainty. 2) Converged models (more training epochs) should have similar uncertainty and accuracy no matter what drop ratio is used. 3) The number of epochs and dropout ratios are important hyper-parameters that have substantial impacts on uncertainty measure and predictive accuracy and should be cross-validated in real applications.

$$\begin{array}{ll}\min\limits_{x,c,d} & \sum_{i,k}(x_{ik}-l_{ik})^2 \\ \text{s.t.} & \alpha_{ik}=\begin{cases}0 & \text{if } P_{ik}\leq c, \text{ for } k \text{ in } 1,\ldots,K \\ 1 & \text{if otherwise}\end{cases} \\ & \beta_{ik}=\begin{cases}0 & \text{if } V_{ik}\geq d, \text{ for } k \text{ in } 1,\ldots,K \\ 1 & \text{if otherwise}\end{cases} \\ & x_{ik}=0 \quad \text{if } \ \alpha_{ik}=0 \text{ OR } \beta_{ik}=0 \\ & x_{ik}=1 \quad \text{if } \alpha_{ik}=1 \text{ AND } \beta_{ik}=1 \\ & \sum_{k}^{K+1}x_{ik}=1 \quad \forall i \text{ in } 1,\ldots,N \\ & 1\geq c\geq 0 \\ & 1\geq d\geq 0\end{array} \quad . \tag{2}$$

We use mean probabilities and standard deviations obtained from models where dropout ratios are set to 10% after 30 epochs of training to learn optimal decision thresholds. Our goal is to optimize lower bound c and upper bound d and designate a question as relevant only when the mean predictive probability P_{ik} is larger than c and standard deviation V_{ik} is lower than d. Optimizing c and d, on a 381-class problem, is much more computationally challenging than learning entropy threshold because the number of constraints is proportional to class number. As shown in **Eq. 2**, we introduce two variables α and β to indicate the status of mean probability and deviation conditions, and the final assignment variable x is the logical AND of α and β. Solving 2) with more than 10 k samples is very slow (shown in **Supplementary Appendix**), so we use 1,500 original relevant questions and increase the number of escalation questions from 100 to 3,000. For performance testing, the optimized c and d are applied as decision variables

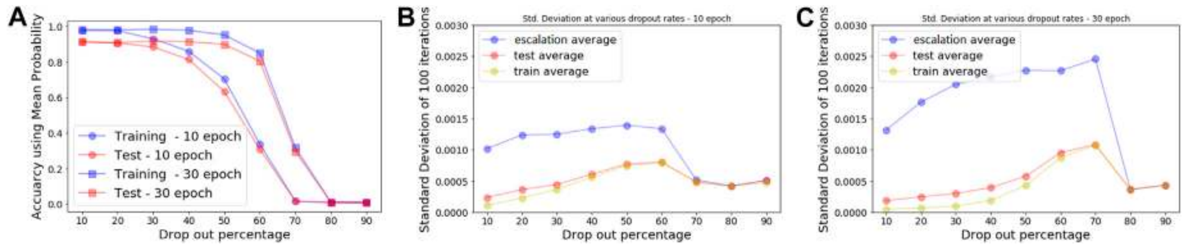

FIGURE 4 | Classification accuracy and uncertainties obtained from Monte Carlo dropout.

TABLE 3 | Performance of cross-comparison of three approaches evaluated on test data of the same size (906 relevant questions plus 4,000 escalation questions). Precision/recall/F1 scores were calculated assuming relevant questions are true positives. In entropy and dropout optimization processes, δ is set to 0.5. Other delta values for the dropout approach are listed in **Supplementary Appendix**.

Number of escalation questions in training	**Entropy**				**Dropout**				**Dummy class**			
	1,000	**5,000**	**8,000**	**10,000**	**100**	**1,000**	**2000**	**3,000**	**1,000**	**5,000**	**8,000**	**10,000**
Optimal entropy cutoff *b*	2.36	1.13	0.85	0.55	–	–	–	–	–	–	–	–
Optimal mean probability cutoff *c*	–	–	–	–	0.8172	0.6654	0.7921	0.0459	–	–	–	–
Optimal standard cutoff *d*	–	–	–	–	0.1533	0.0250	0.0261	0.0132	–	–	–	–
Mean accuracy in 381 classes	91.9%	88.3%	85.6%	81.7%	88.41%	80.13%	80.24%	74.72%	94.2%	**93.7%**	87.7%	82%
Accuracy of the dummy class	79.25%	91.2%	93.25%	95.2%	86.69%	91.83%	91.95%	92.57%	73.6%	**94.5%**	99.4%	99.6%
Precision (binary classification)	51.4%	70.2%	74.7%	79.8%	90.7%	68.8%	68.9%	63.7%	81%	**95.3%**	99.5%	99.6%
Recall (binary classification)	96.7%	91.3%	88.1%	83.5%	93.9%	82.7%	83.2%	84.7%	99.7%	**98.7%**	92.6%	86%
F1 score (binary classification)	0.671	0.794	0.808	0.816	0.738	0.751	0.754	0.727	0.894	**0.967**	0.959	0.923

Bold values represent best performance.

on samples of BERT predictions on test data. Performance from dropout is presented in **Table 3** and **Supplementary Appendix**. Our results showed a decision threshold optimized from **Eq. 2** involving 2000 escalation questions and gave the best F1 score (0.754), and we validated it using the grid search and confirmed its optimality (shown in **Supplementary Appendix**).

5.4 Dummy Class Classification

Our third approach is to train a binary classifier using both relevant questions and escalation questions in the BERT model. We use a dummy class to represent those 17,395 escalation questions and split the entire data sets, including relevant and escalation, into five folds for training and test.

Performance of the dummy class approach is compared with entropy and dropout approaches (**Table 3**). Deciding an optimal number of escalation questions involved in threshold learning is non-trivial, especially for entropy and dummy class approaches. Dropout does not need as many escalation questions as entropy does to learn the optimal threshold mainly because the number of constraints in **Eq. 2** is proportional to the class number (381), so the number of constraints is large enough to learn a suitable threshold on small samples. (To support this conclusion, we present extensive studies in **Supplementary Appendix** on a 5-class classifier using Tier one intents.) The dummy class approach obtains the best performance, but its success assumes the learned decision boundary can be generalized well to any new escalation questions, which is often not valid in real applications. In contrast, entropy and dropout approaches only need to treat a binary problem in the optimization and leave the intent classification model intact. The optimization problem for entropy approach can be solved much more efficiently and is selected as the solution for our final implementation.

It is certainly possible to combine dropout and entropy approach, for example, to optimize thresholds on entropy calculated from the average mean of MCD dropout predictions. Furthermore, it is possible that the problem defined in **Eq. 2** can be simplified by proper reformulation and can be solved more efficiently, which will be explored in our future works.

6 SENTENCE COMPLETION USING LANGUAGE MODEL

6.1 Algorithm

We assume misspelled words are all OOV words, and we can transform them as [MASK] tokens and use bidirectional language models to predict them. Predicting masked word within sentences is an inherent objective of a pretrained bidirectional model, and we utilize the masked language model API in the Transformer package [61] to generate the ranked list of candidate words for each [MASK] position. The sentence completion algorithm is illustrated in Algorithm 1.

6.2 Experimental Setup

For each question, we randomly permutate two characters in the longest word, the next longest word, and so on. In this way, we generate one to three synthetic misspellings in each question. We investigate intent classification accuracy changes on these questions, and how our sentence completion model can prevent performance changes. All models are trained using relevant data (80%) without misspellings and validated on synthetic misspelled test data. Five settings are compared: 1) no correction: classification performance without applying any autocorrection; 2) no LM: autocorrections made only by word edit distance without using masked language model; 3) BERT SharePoint: autocorrections made by masked LM using pretrained SharePoint embeddings together with word edit distance; 4) BERT Email: autocorrections using pretrained email embeddings together with word edit distance; and 5) BERT Google: autocorrections using pretrained Google small uncased embedding data together with word edit distance.

Algorithm 1 Auto-correction of Multiple OOV tokens. Assuming there are d OOV tokens, we first find top B combinations using beam search that maximizes joint probability $\prod_{i=1}^{d} P_i$, then among them select the combination that has the smallest accumulated Levenshtein distances to thoses OOV tokens.

```
1:  procedure AUTO-CORRECT($< t_0, .., t_N >, I$)    ▷ $I$ is the list of OOV indices
2:      initialize $P \in D_{M \times d}$    ▷ Prob. Matrix of candidate tokens
3:      initialize $\hat{P} \in D_{B \times d}$    ▷ Optimal prob. Matrix
4:      for $i$ in $I_{1 \times d}$ do
5:          $oov_i \leftarrow t_i$
6:          $t_i \leftarrow$ [MASK]
7:      end for
8:      $W, P \leftarrow$ MaskedLM($t_0, .., t_N, M$)
9:      for $w_{j,-}$ in $\hat{W}$ do    ▷ iterate top M candidates
10:         $d_{j,-} = \sum_i$ WordEditDistance($w_{ji}, oov_i$)
11:     end for
12:     $\hat{j} = \arg\min_j d_{j,-}$    ▷ The candidate with shortest edit distance
13:     for $i$ in $I_{1 \times d}$ do
14:         $t_i = w_{\hat{j}i}$    ▷ Autocorrect the oov token
15:     end for
16:     return $< t_0, .., t_N >$ ▷ Return the autocompleted sentence
17: end procedure
```

We also need to decide what is an OOV or what should be included in our vocabulary. After experiments, we set our vocabulary as words from four categories: 1) All words in the pretrained embeddings; 2) all words that appear in training questions; 3) words that are all capitalized because they are likely to be proper nouns, fund tickers, or service products; 4) all words start with numbers because they can be tax forms or specific products (e.g., 1099b and 401 k). The purposes of including 3) and 4) are to avoid autocorrection on those keywords that may represent significant intents. Any word falls outside these four groups is considered as an OOV. During our implementation, we keep monitoring the OOV rate, defined as the ratio of OOV occurrences to total word counts in recent 24 h. When it is higher than 1%, we apply manual intervention to check chatbot log data.

We also need to determine two additional parameters M, the number of candidate tokens prioritized by masked language model, and B, the beam size in our sentence completion model. In our approach, we set M and B to the same value, and it is benchmarked from 1 to 10 k by test sample accuracy. Notice that when M and B are large, and when there are more than two OOVs, beam search becomes very inefficient in Algorithm 1. To simplify this, instead of finding the optimal combinations of candidate tokens that maximize the joint probability argmax$\prod_{i=1}^{d} p_i$, we assume they are independent and apply a simplified algorithm (shown in **Supplementary Appendix**) on single OOV separately.

TABLE 4 | Comparison of intent classification accuracy using the best BERT model vs. conventional n-gram models.

Model	Single OOV	Two OOVs	Three OOVs
BERT SharePoint	0.934	0.882	0.849
5 G	0.755	0.719	0.622
5-4 G	0.817	0.731	0.643
5-4-3 G	0.823	0.752	0.646
5-4-3-2 G	0.826	0.755	0.643

TABLE 5 | Benchmark of intent classification API performance across different models in real-time application. Each model is tested using 10 threads, simulating 10 concurrent users, for a duration of 10 min. In this test, models are not served as Monte Carlo sampling, so the inference is done only once. All models are hosted on identical AWS m5.4xlarge CPU instances. As seen, the simplest model (6A-6H, six attention layers and six hidden layers) can have a double throughput rate and half latency than the original BERT small model, and the accuracy performance only drops 1.6%. The performance is evaluated using JMeter at the client side, and APIs are served using Domino Lab 3.6.17 Model API. Throughput indicates how many API responses being made per second. Latency is measured as time elapse between request sent till response received at client side.

Model	Performance	Throughput	Average latency (ms)
12A-12H	0.944	8.9/s	1,117
6A-12H	0.941	9.0/s	1,108
12A-9H	0.934	11.8/s	843
3A-9H	0.933	12.0/s	831
3A-12H	0.930	9.1/s	1,097
6A-6H	0.928	18.1/s	552

In additional to BERT, we also implemented a conventional spelling correction algorithm using Google Web 1 T n-gram [62]. We used the longest common subsequence (LCS) string matching algorithm [63] and compared a variety of best combinations of n-grams report in the article. The experimental setting is identical as the one we set up for BERT models: We apply auto-spelling correction algorithms on synthetic misspelled test data (20%), and then the intent classification accuracy performance is evaluated using the BERT SharePoint model trained on 80% relevant data without misspellings for 10 epochs. As shown in **Table 4**, n-gram models do not provide comparable performance as BERT language models, and the most complicated hybrid n-gram models (5-4-3 g and 5-4-3-2 g) [63] is not comparable to Google BERT model and far worse than BERT SharePoint model.

An further improved version of sentence completion algorithm to maximize joint probability is our future research. In this article, we have not considered situations when misspellings are not OOV. Detecting improper words or improper grammar in a sentence may need evaluation of metrics such as perplexity or sensibleness and specificity average (SSA) [10], and the simple word matching algorithm can be much generalized as reinforcement learning–based approach [64].

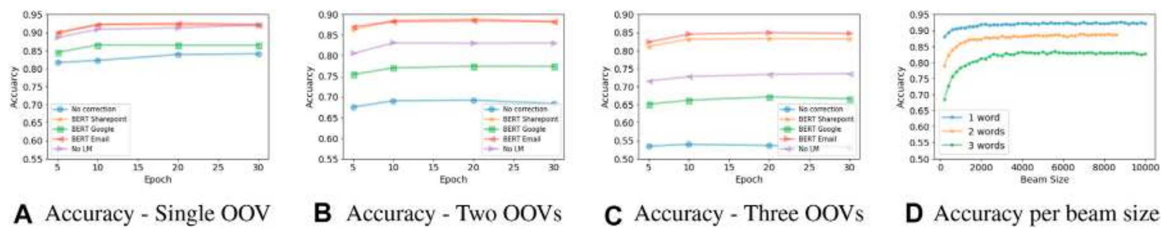

FIGURE 5 | As expected, misspelled words can significantly decrease intent classification performance. The same BERT model that achieved 94% on clean data, dropped to 83.5% when a single OOV occurred in each question. It further dropped to 68 and 52%, respectively, when two and three OOVs occurred. In all experiments, LM models proved being useful to help correcting words and reduce performance drop, while domain-specific embeddings trained on Vanguard SharePoint and Email text outperform off-the-shelf Google embeddings. The beam size *B* (*M*) was benchmarked as results shown in subfigure **(D)** and was set to 4,000 to generate results in subfigure **(A–C)**.

6.3 RESULTS

According to the experimental results illustrated in **Figure 5**, pretrained embeddings are useful to increase the robustness of intent prediction on noisy inputs. Domain-specific embeddings contain much richer context-dependent semantics that helps OOVs get properly corrected and leads to better task-oriented intent classification performance. Benchmark shows B $\geq$ 4000 leads to the best performance for our problem. Based on this, we apply SharePoint embeddings as the language model in our sentence completion module.

7 IMPLEMENTATION

The chatbot has been implemented fully inside our company network using open-source tools including RASA [65], TensorFlow, and PyTorch in Python environment. All backend models (sentence completion model, intent classification model, and others) are deployed as RESTful APIs in AWS SageMaker. The front end of chatbot is launched on Microsoft Teams, powered by Microsoft Bot Framework and Microsoft Azure Directory, and connected to backend APIs in AWS environment. All our BERT model trainings, including embeddings pretraining, are based on BERT TensorFlow running on AWS P3.2xlarge instance. The optimization procedure uses Gurobi 8.1 running on AWS C5.18xlarge instance. The BERT language model API in the sentence completion model is developed using Transformer 2.1.1 package on PyTorch 1.2 and TensorFlow 2.0.

During our implementation, we further explore how the intent classification model API can be served in real applications under budget. We gradually reduce the numbers of attention layer and hidden layer in the original BERT small model (12 hidden layers and 12 attention heads) and create several smaller models. By reducing the number of hidden layers and attention layers in half, we see a remarkable 100% increase in performance (double the throughput and half the latency) with the cost of only 1.6% drop in intent classification performance (**Table 5**).

8 CONCLUSION

Our results demonstrate that optimized uncertainty thresholds applied on BERT model predictions are promising to escalate escalation questions in task-oriented chatbot implementation, meanwhile the state-of-the-art deep learning architecture provides high accuracy on classifying into a large number of intents. Another feature we contribute is the application of BERT embeddings as the language model to automatically correct small spelling errors in noisy inputs, and we show its effectiveness in reducing intent classification errors. The entire end-to-end conversational AI system, including two machine learning models presented in this article, is developed using open-source tools and deployed as in-house solution. We believe those discussions provide useful guidance to companies that are motivated to reduce dependency on vendors by leveraging state-of-the-art open-source AI solutions in their business.

We will continue our explorations in this direction, with particular focuses on the following issues: 1) Current fine-tuning and decision threshold learning are two separate parts, and we will explore the possibility to combine them as a new cost function in BERT model optimization. 2) Dropout methodology applied in our article belongs to approximated inference methods, which is a crude approximation to the exact posterior learning in parameter space. We are interested in a Bayesian version of BERT, which requires a new architecture based on variational inference using tools like TFP TensorFlow Probability. 3) Maintaining chatbot production system would need a complex pipeline to continuously transfer and integrate features from deployed model to new versions for new business needs, which is an uncharted territory for all of us. 4) Hybridizing "chitchat" bots, using state-of-the-art progresses in deep neural models, with task-oriented machine learning models is important for our preparation of client self-provisioning service.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusion of this article is available at https://github.com/cyberyu/ava.

AUTHOR CONTRIBUTIONS

SY: main author, deployed the model and wrote the paper. Corresponding author YC: deployed the model and wrote the paper HZ: deployed the model and wrote the paper.

ACKNOWLEDGMENTS

We thank our colleagues in Vanguard CAI (ML-DS team and IT team) for their seamless collaboration and support. We thank colleagues in Vanguard Retail Group (IT/Digital, Customer Care) for their pioneering effort collecting and curating all the data used in our approach. We thank Robert Fieldhouse, Sean Carpenter, Ken Reeser and Brain Heckman for the fruitful discussions and experiments.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fams.2021.604842/full#supplementary-material

REFERENCES

1. Devlin J, Chang M-W, Lee K, and Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the NACL, Vol 1 (2019). p. 4171–86.
2. Colby KM, Weber S, and Hilf FD. Artificial Paranoia. *Artif Intelligence* (1971). 2(1):1–25. ISSN. doi:10.1016/0004-3702(71)90002-6
3. Weizenbaum J. ELIZA-a Computer Program for the Study of Natural Language Communication between Man and Machine. *Commun ACM* (1966). 9(1):36–45. ISSN. doi:10.1145/365153.365168
4. Worswick S. Mitsuku (2019). Available at: http://www.mitsuku.com.
5. Gao J, Galley M, and Li L. *Neural Approaches to Conversational Ai*. SIGIR '18 (2018).
6. Fedorenko DG, Smetanin N, and Rodichev A (2017). Avoiding echo-responses in a Retrieval-Based Conversation System. In: Conference on Artificial Intelligence and Natural Language. p. 91–7.
7. Microsoft (2019). Microsoft. Zo. Available at: https://www.zo.ai.
8. Serban IV, Sankar C, Germain M, Zhang S, Lin Z, Subramanian S, et al. (2017). A Deep Reinforcement Learning Chatbot. *CoRR* Available at: http://arxiv.org/abs/1709.02349.
9. Zhou L, Gao J, Li D, and Shum H (2018). The Design and Implementation of Xiaoice, an Empathetic Social Chatbot. *CoRR, abs/1812.08989*.
10. Adiwardana D, Luong M-T, So DR, Hall J, Fiedel N, Thoppilan R, et al. (2020). *Towards a Human-like Open-Domain Chatbot*. Available at: https://arxiv.org/abs/2001.09977
11. Larson S, Mahendran A, Peper JJ, Clarke C, Lee A, Hill P, et al. (2019). 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). Association for Computational Linguistics.
12. Sethi S (2019). The State of Chatbots in 2019. Available at: https://hackernoon.com/the-state-of-chatbots-in-2019-d97f85f2294b.
13. Dai AM, and Le QV (2015). Semi-supervised Sequence Learning. In: In Proceedings of Advances in Neural Information Processing Systems 28. p. 3079–87.
14. Howard J, and Ruder S (2018). Universal Language Model fine-tuning for Text Classification. In: Proceedings of the 56th Annual Meeting of the ACL, Melbourne, Australia, July 2018. p. 328–39.
15. Lample G, and Conneau A (2019). Cross-lingual Language Model Pretraining. *CoRR, abs/1901.07291*.
16. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, et al. (2019). Roberta: A Robustly Optimized BERT Pretraining Approach. *CoRR, abs/1907.11692*.
17. Peters ME, Neumann M, Zettlemoyer L, and Yih W (2018). Dissecting Contextual Word Embeddings: Architecture and Representation. *CoRR, abs/1808.08949*.
18. Peters M, Ammar W, Bhagavatula C, and Power R. Semi-supervised Sequence Tagging with Bidirectional Language Models. In: Proceedings of the 55th ACL. Vancouver, Canada (2017). p. 1756–65.
19. Peters M, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, et al. (2018). Deep Contextualized Word Representations. In: Proceedings of the 2018 Conference of the NAACL. New Orleans, Louisiana. p. 2227–37.
20. Tang R, Lu Y, Liu L, Mou L, Vechtomova O, and Lin J (2019). Distilling Task-specific Knowledge from BERT into Simple Neural Networks. *CoRR, abs/1903.12136*.
21. Yang Z, Dai Z, Yang Y, Carbonell JG, Salakhutdinov R, and Le QV (2019). Xlnet: Generalized Autoregressive Pretraining for Language Understanding. *CoRR, abs/1906.08237*.
22. Gal Y, and Ghahramani Z (2016). Dropout as Bayesian Approximation: Representing Model Uncertainty in Deep Learning 48:1050.
23. Maddox WJ, Garipov T, Izmailov P, Vetrov DP, and Wilson AG (2019). A Simple Baseline for Bayesian Uncertainty in Deep Learning. In: H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett. editors. *Advances in Neural Information Processing Systems*. Red Hook, NYCurran Associates, Inc.
24. Pearce T, Zaki M, Brintrup A, and Neely A (2018). Uncertainty in Neural Networks: Bayesian Ensembling. *ArXiv, abs/1810.05546*.
25. Shridhar K, Laumann F, and Liwicki M (2019). A Comprehensive Guide to Bayesian Convolutional Neural Network with Variational Inference. *CoRR, abs/1901.02731*.
26. Li C, Stevens A, Chen C, Pu Y, Gan Z, and Carin L (2016). Learning Weight Uncertainty with Stochastic Gradient Mcmc for Shape Classification. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). p. 5666–75.
27. Park C, Kim J, Ha SH, and Lee J (2018). Sampling-based Bayesian Inference with Gradient Uncertainty. *CoRR, abs/1812.03285*.
28. Rao Q, and Frtunikj J (2018). Deep Learning for Self-Driving Cars: Chances and Challenges. In: 2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS). Los Alamitos, CA: IEEE Computer Society. p. 35–8.
29. Seedat N, and Kanan C (2019). Towards Calibrated and Scalable Uncertainty Representations for Neural Networks. *ArXiv, abs/1911.00104*.
30. Welling M, and Teh YW (2011). Bayesian Learning via Stochastic Gradient Langevin Dynamics. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11, ISBN 9781450306195. Madison, WI, USA: Omnipress. p. 681–8.
31. Blundell C, Cornebise J, Kavukcuoglu K, and Wierstra D (2015). Weight Uncertainty in Neural Network. In: B. Francis and B. David. editors. Proceedings of the 32nd International Conference on Machine Learning. PMLR p. 1613–22.
32. Graves A (2011). Practical Variational Inference for Neural Networks. *Adv Neural Inf Process Syst* 24:2348–56. doi:10.1016/s0893-6080(10)00238-8
33. Hernández-Lobato JM, and Adams RP (2015). Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In: Proceedings of the 32nd ICML, Vol 37, ICML'15, Lille, France. p. 1861–9.
34. Ye N, and Zhu Z (2019). Functional Bayesian Neural Networks for Model Uncertainty Quantification. Available at: https://openreview.net/forum?id=SJxFN3RcFX.
35. Miok K, Skrlj B, Zaharie D, and Robnik-Sikonja M (2020). To Ban or Not to Ban: Bayesian Attention Networks for Reliable Hate Speech Detection. *arXiv: Appl*.
36. Lin Y, Michel J-B, Aiden EL, Orwant J, Brockman W, and Petrov S (2012). Index. In: Proceedings of the ACL 2012 System Demonstrations. USA. p. 169–74. doi:10.2307/j.ctv18pgr3b.13
37. Schuster M, and Nakajima K (2012). Japanese and Korean Voice Search. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). p. 5149–52. doi:10.1109/ICASSP.2012.6289079
38. Gage P (1994). A New Algorithm for Data Compression. *C Users J* 12(2): 23–38.
39. Sennrich R, Haddow B, and Birch A. Neural Machine Translation of Rare Words with Subword Units. In: Proceedings of the 54th ACL, Berlin, Germany. Stroudsburg, PA: Association for Computational Linguistics (2016). p. 1715–25.
40. Mikolov T, Sutskever I, Chen K, Corrado G, and Dean J (2013). Distributed Representations of Words and Phrases and Their Compositionality. *CoRR, abs/1310.4546*.
41. Mikolová T, Karafit M, Burget L, Cernocký J, and Khudanpur S (2010). Recurrent Neural Network Based Language Model. In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*. Makuhari, Japan 2:1045–8.
42. dos Santos C, and Gatti M (2014). Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In: Proceedings of the 25th International Conference on Computational Linguistics. Dublin, Ireland. p. 69–78.

43. Dos Santos CN, and Zadrozny B (2014). Learning Character-Level Representations for Part-Of-Speech Tagging. In: Proceedings of the 31st International Conference on Machine Learning - Volume 32, ICML'14, Beijing, China. p. II–1818–II–1826.
44. Kim Y, Jernite Y, Sontag DA, and Rush AM (2015). Character-aware Neural Language Models. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 2741–9.
45. Radford A, and Sutskever I (2018). Improving Language Understanding by Generative Pre-training. Available at: https://openai.com/blog/language-unsupervised/.
46. Collobert R, and Weston J. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In: Proceedings of the 25th ICML, Helsinki, Finland. ACM Press (2008). p. 160–7. doi:10.1145/1390156.1390177
47. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, and Kuksa PP (2011). Natural Language Processing (Almost) from Scratch. *J Mach Learn Res* 12:2493–537.
48. Elman JL (1990). Finding Structure in Time. *Cognitive Science* 14(2):179–211. doi:10.1207/s15516709cog1402_1
49. Bahdanau D, Cho K, and Bengio Y (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In: 3rd International Conference on Learning Representations. San Diego, CA, USA: ICLR 2015.
50. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. (2017). Attention Is All You Need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY: Curran Associates Inc. P. 6000–10.
51. Kuleshov V, and Ermon S (2018). Accurate Uncertainties for Deep Learning Using Calibrated Regression. *Proceedings of the 35th International Conference on Machine Learning*. PMLR 80:2796–804.
52. Ghavamzadeh M, Mannor S, Pineau J, and Tamar A (2016). Bayesian Reinforcement Learning: A Survey. *Found Trends Mach Learn* 8(5-6): 359–483. doi:10.1561/2200000049
53. Guo C, Pleiss G, Sun Y, and Weinberger KQ (2017). On Calibration of Modern Neural Networks. In: P. Doina and T. Yee Whye. editors. Proceedings of the 34th International Conference on Machine Learning 70:1321–30. . PMLR.
54. MacKay D (1992). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation* 4:448–72. doi:10.1162/neco.1992.4.3.448
55. Neal R (1995). Bayesian Learning for Neural Networks. Berlin, Heidelberg: Springer-Verlag.
56. Lakshminarayanan B, Prtizel A, and Blundell C (2017). *Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles*. p. 6405.
57. Kendall A, and Gal Y (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?. In: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, and S. Vishwanathan. editors. *Advances in Neural Information Processing Systems*. Curran Associates, Inc. 30.
58. Geifman Y (2019). Selective Net: A Deep Neural Network with an Integrated Reject Option. In: K. Chaudhuri and R. Salakhutdinov. editors. *Proceedings of the 36th International Conference on Machine Learning*. PMLR. p. 2151–2159. Available at: http://proceedings.mlr.press/v97/geifman19a/geifman19a.pdf.
59. Atpaino (2017). Selective Net: A Deep Neural Network with an Integrated Reject Option. Available at: https://github.com/atpaino/deep-text-corrector.
60. Lepora NF (2016). Threshold Learning for Optimal Decision Making. *Adv Neural Inf Process Syst* 29:3763–71.
61. HuggingFace (2017). Transformers. Available at: https://github.com/huggingface/transformers.
62. Brants T, and Franz A (2006). *Web 1T 5-gram Version 1*. Philadelphia: Linguistic Data Consortium. p. LDC2006T13.
63. Islam A, and Inkpen D (2009). Real-word Spelling Correction Using Google Web it 3-grams. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3, EMNLP '09. USA: Association for Computational Linguistics. p. 1241–9.
64. Chen JZ, Yu S, and Wang H (2020). Exploring Fluent Query Reformulations with Text-To-Text Transformers and Reinforcement Learning. *ArXiv, abs/2012.10033*.
65. Bocklisch T, Faulkner J, Pawlowski N, and Nichol A (2017). Rasa: Open Source Language Understanding and Dialogue Management. *CoRR, abs/1712.05181*. Available at: http://arxiv.org/abs/1712.05181.

Conflict of Interest: All authors were employed by the company The Vanguard Group.