Agent Assist through Conversation Analysis

Kshitij P. Fadnis[1], **Nathaniel Mills**[1], **Jatin Ganhotra**[1], **Haggai Roitman**[2*], **Gaurav Pandey**[1], **Doron Cohen**[1], **Yosi Mass**[1], **Shai Erera**[1], **Chulaka Gunasekara**[1], **Danish Contractor**[1], **Siva Sankalp Patel**[1], **Q. Vera Liao**[1], **Sachindra Joshi**[1], **Luis A. Lastras**[1], **David Konopnicki**[1]
[1]IBM Research, [2]eBay Research
{kpfadnis, wnm3, jatinganhotra, lastrasl}@us.ibm.com
{hroitman}@ebay.com, {doronc, yosimass, shaie, davidko}@il.ibm.com
{siva.sankalp.patel, chulaka.gunasekara, vera.liao}@ibm.com
{gpandey1, dcontrac, jsachind}@in.ibm.com,

Abstract

Customer support agents play a crucial role as an interface between an organization and its end-users. We propose **CAIRAA: Conversational Approach to Information Retrieval for Agent Assistance**, to reduce the cognitive workload of support agents who engage with users through conversation systems. CAIRAA monitors an evolving conversation and recommends both responses and URLs of documents the agent can use in replies to their client. We combine traditional information retrieval (IR) approaches with more recent Deep Learning (DL) models to ensure high accuracy and efficient run-time performance in the deployed system. Here, we describe the CAIRAA system and demonstrate its effectiveness in a pilot study via a short video[1].

1 Introduction

Customer care conversation systems have been used in a variety of domains, including technical support, reservation systems, and banking applications (Acomb et al., 2007). The majority of such systems provide a dashboard to customer support agents, so that they can interact with multiple end-users in parallel. The support agents use such dashboards to perform diverse tasks to address user requests, such as question-answering, conversational search, document passage extraction and transactions. When identifying the responses to user queries, the support agents either, (1) rely on their own domain knowledge and articulate such knowledge to be sent to the users or, (2) manually extract the keywords from the conversation and use search functions provided in their dashboards to identify the relevant knowledge contained in documents, and send URLs for these documents. The continuous process of identifying knowledge and responding becomes an immense cognitive workload for the customer support agents.

Dialogue	
A:	`Hello. Thank you for contacting Help@IBM. How may I help you?`
U:	Hi yes. I cannot connect to ibm connections cloud in ios
U:	yesterday my phone asked for the pw out of the blue and I clicked cancel bec on the road and now I have no connection to server
A:	No worries. First you will need to create a 16 digit password for for ibm connections cloud `https://w3.ibm.com/help/#/article/ios_create_16char_pass` then you will need to open on the iphone settings-accounts and passwords-ibm connections cloud click on your email address and in the password field enter this 16 digit password.
U:	I have that pw. Can I use my old one or better to create a new one?
A:	`Please always try the existing password first. If it doesn't work, then create a new password.`
U:	Worked. ;) Thanks

Table 1: Sample dialog from Help@IBM where Agent (**A**) utterance includes a URL to the User (**U**) query.

Although automated conversation systems have improved immensely in the last decade with advances in natural language processing, machine learning and dialog management (Wen et al. (2016); Li et al. (2016); Li et al. (2017)), these systems still fail to satisfy the sophisticated customer's needs in real-life scenarios. This leads to frustration (Weisz et al., 2019) and less engagement (Vtyurina et al., 2017). Therefore, having an empathetic human agent in-the-loop supported by efficient and accurate content retrieval, allows better coverage of customer needs and reduces customer frustration.

With CAIRAA, we propose and showcase a system that provides real-time assistance to the support agents and alleviates their cognitive workload. Our system provides two forms of real-time rec-

[*] Work done when author was at IBM Research
[1] `https://youtu.be/EeqMLLBWhxQ`

Proceedings of the 2020 EMNLP (Systems Demonstrations), pages 151–157
November 16-20, 2020.

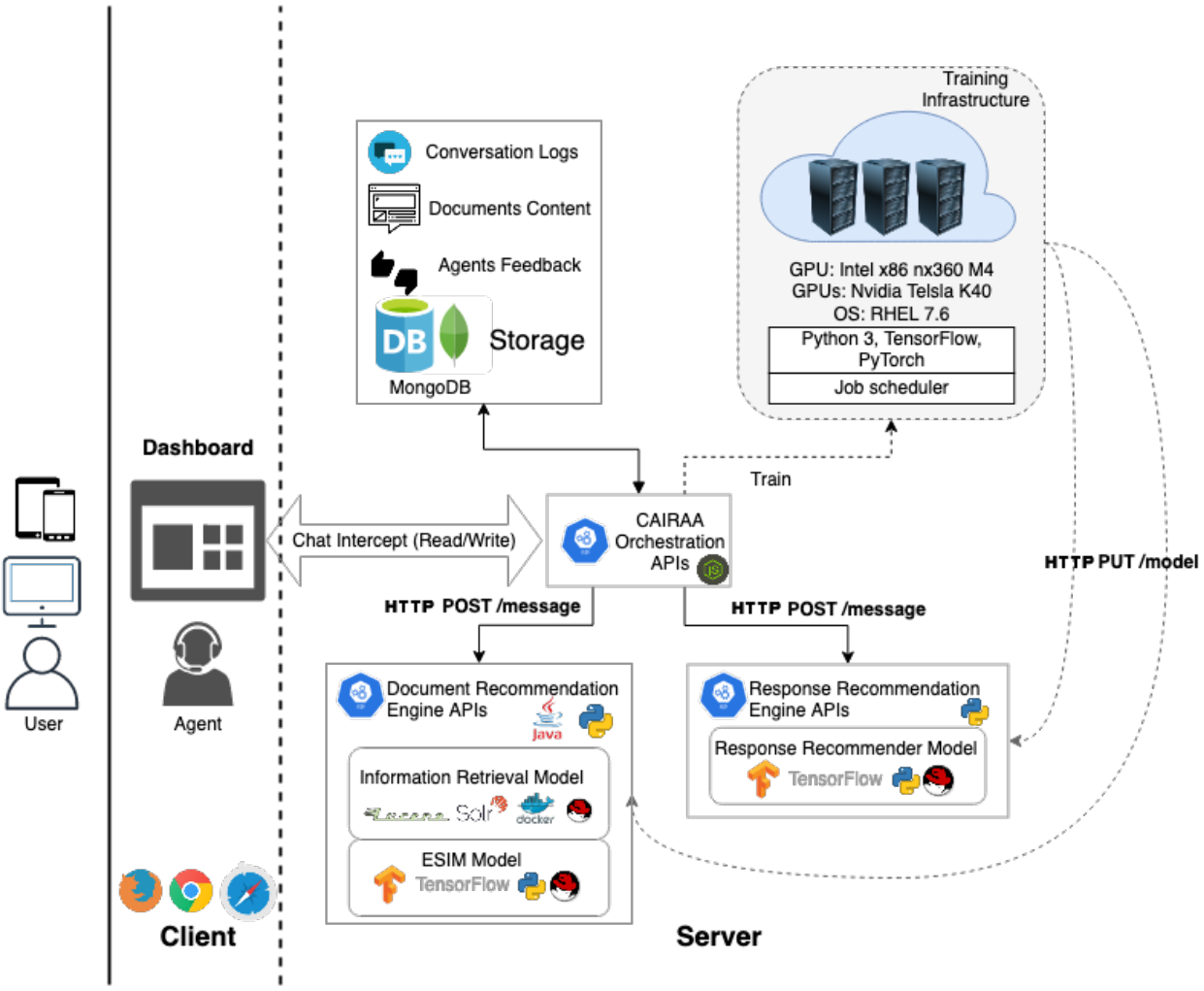

Figure 1: **CAIRAA System Architecture**: (a) Agent Dashboard - a web application (b) Orchestrator APIs - server side communications, controls data and process flow (c) Document Recommendation Engine - retrieves and recommends documents relevant to conversational context (d) Response Recommendation Engine - recommends agent responses based on conversational context (f) Storage - retains document content, conversation and system logs, agents feedback and activity (g) Training infrastructure - a dedicated cluster for deep learning models training (In development)

ommendations to the support agents: (1) URLs of the documents that contain information to resolve user issues, and (2) natural language responses that the agent can use to respond to their customer's queries. The operation of CAIRAA is illustrated in Table 1 with a sample conversation between a user and a support agent. In the example, the utterances made by the support agents are prefixed by **A** and the utterances made by the user are prefixed by **U**. The natural language utterances that CAIRAA predicted for the support agent are shown in different font (`monospace`) while the URLs of the related documents predicted by CAIRAA are shown in blue. In the following sections we describe the different components of our system and their implementation details.

2 System

The architecture of CAIRAA is illustrated in Figure 1. It consists of an Agent-dashboard that is used by customer-support agents to interact with customers (users). The agent is assisted by recommendations from two engines - Document Recommendation Engine and the Response Recommendation Engine. As their names suggest, these engines provide real-time recommendations for *documents* that could be relevant during the chat, as well as, *responses* to the agent. Figure 1 also depicts components for data storage as well as model training.

2.1 Web Content Extraction

Given a collection of human-to-human conversation logs, we extract the mentions of URLs (doc-

uments) from these conversations. We use Selenium[2] to render static as well as dynamic web-content. With our primary focus on text content, additional cleaning processes that filters selected HTML content (e.g., menus, search bars, side bars, headers and footers) and only preserves text content, along with embedded procedural and multimedia content references is implemented. The processed content is exported in two formats. a) markdown and b) formatted text.

2.2 Document Recommendation Engine

Given a set of conversational logs along with documents mentioned in those conversations, we train the Document Recommendation Engine using a pipeline consisting of an information-retrieval model followed by a deep-learning model to recommend URLs relevant to an evolving conversation. This Document Recommendation Engine is trained with the objective of predicting the most relevant web content for the conversation at hand. Once trained, this is provided as a real-time service to the agent dashboard to recommend the appropriate URLs to support agents while they converse.

2.2.1 Information Retrieval model

The Information Retrieval (IR) model is implemented using an Apache Lucene index, employed with English language analyzer and default BM25 similarity. Documents in the index are represented using two fields, (1) web page content, (2) document's representation augmented with the text of all historic conversations that link to it.

For a given (dialog) query d, matching documents are retrieved using four different ranking steps, which are combined using a cascade approach (Wang et al., 2011). Following (Van Gysel et al., 2016), we obtain an initial pool of candidate documents using a lexical query aggregation approach. To this end, each utterance $t_i \in d$ is represented as a separate weighted query-clause, having its weight assigned relatively to its sequence position in the dialog (Van Gysel et al., 2016). Various sub-queries are then combined using a single disjunctive query. The second ranker evaluates each document y obtained by the first ranker against an expanded query (applying relevance model (Lavrenko and Croft, 2001)). The third ranker applies a manifold-ranking approach (Xu et al., 2011), aiming to score content-similar documents (measured by Bhattacharyya language-model based similarity) with similar scores.

The last ranker in the cascade treats the dialog query d as a verbose query and applies the *Fixed-Point* (FP) method (Paik and Oard, 2014) for weighting its words. Yet, compared to "traditional" verbose queries, dialogs are further segmented into distinct utterances. Using this approach, we implement an *utterance-biased* extension for enhanced word-weighting. To this end, we first score the various utterances based on the initial FP weights of words they contain and their relative position. We then propagate utterance scores back to their associated words.

2.2.2 Deep Learning Model

We use the *Enhanced Sequential Inference Model* (ESIM) proposed by Chen et al. (2017) with the same goal as the IR model but it uses dense vectors to represent conversation-contexts and documents. The objective is to predict the relevant URL given the dialog history (context). The multi-turn dialog history is concatenated together to form the context of length m, represented as $C = (c_1, c_2, ..., c_i, ..., c_m)$, where c_i is the ith word in context. Given a web page content U as $U = (r_1, r_2, ..., r_j, ..., r_n)$, where r_j is the jth word in web page content, the web page is selected using the conditional probability $P(y = 1|C, U)$, which shows the confidence of selecting the web page U given context C.

We observe that the IR model is much faster than the neural ESIM model, but the ESIM model provides improved performance in comparison. We combine the ESIM model with the IR model using a re-ranking of latter's candidate pool, which provides a combination of both ranker models. For example, the IR model returns the top-k relevant web pages ($k = 20$) and then the ESIM model is used to re-rank them and show a subset to the agent based on their confidence scores. We refer to this two-stage pipeline as a *hybrid* approach, which combines the best of both worlds and deliver near real-time experience with better performance.

2.2.3 Rating and Confidence Estimation

In addition to providing a ranked list of webpages, the Document Recommendation Engine provides a rating and confidence estimate for each recommended web content, allowing to better guide the agent to the best solutions. The rating and confidence per each single recommended content URL

[2]http://www.seleniumhq.org

is estimated using a novel *query performance prediction* (QPP) model (Roitman et al., 2019), trained over a multitude of features obtained from the conversation context and recommended content analysis.

Besides, at every step of the conversation, a crucial decision that the Document Recommendation Engine has to make is to decide whether or not to present the recommendations to the agent. In case of a low confidence, the human agent may ask for further clarifications from the end-user. Such a decision is taken by training another confidence estimation model that considers the confidence of each individual recommended URL. Here, the system further exploits the interaction between the recommendations made by the IR and ESIM models, with the observation that higher agreement (measured by ranking-similarity) usually translates to higher overall confidence (Roitman and Kurland, 2019). We assessed the quality of our confidence estimation model by measuring its *accuracy* and *log-loss* per each task. For a single recommended URL, the model is trained to classify it as relevant or not. For a top-k recommended URLs list, the model is trained to determine whether it contains at least one relevant URL.

2.3 Response Recommendation Engine

As mentioned above, an agent is also shown recommended responses based on how other agents have responded to similar conversation contexts in the pasts. Thus, given the current dialog input context $C = (c_1, c_2, ..c_i..c_m)$, CAIRAA generates recommendations using a combination of *generative* as well as *retrieval* based methods.

2.3.1 Multi-task training

We use a hierarchical encoder (Serban et al., 2016) to encode conversation contexts. Specifically, the encoder first encodes each conversation turn and generates *turn-level* representations for the dialog. A secondary encoder then generates the overall context representation using the turn-level encodings.

We utilize this context encoding to generate response recommendations in three ways: (1) Using a vanilla decoder (Serban et al., 2016) (2) Using a decoder that additionally validates whether a subsequence at each time-step is likely to be relevant. (3) Using the encoded representations in a Siamese dual-encoder (Lowe et al., 2017) that also encodes the responses.

Vanilla Decoder: The decoder is initialized using the context encoding. The decoder generates the response autoregressively, that is, the token at each time-step is generated conditioned on the previous tokens of the response. The decoder is trained to minimize the log-perplexity of each word in the gold response.

Decoder with sub-sequence validation: When trained on actual conversation logs, vanilla decoders often resort to generic responses or responses that are irrelevant to the context. Hence, to enforce relevance, we enhance the decoder with a classifier for each time-step of decoding. At each time-step, the classifier predicts the relevance of the response so-far for the given conversational context. The classifier is trained to predict a relevance of 1 for a prefix of the gold response and 0 for a prefix of any other randomly sampled response at each time-step of decoding. Simultaneously, the decoder is also trained to minimize the word loss, that is, log-perplexity of each word in the gold response. For any response r, the relevance loss can be written as follows:

$$loss_r(r) = -\sum_{t=1}^{T} \log p(y_t | w_1, \ldots, w_t) \,, \qquad (1)$$

where $y_t = 1$ for the gold response and 0 for the randomly sampled response.

During inference, the token at each time-step is generated so as to maximize the sum of log-probability of the token and the log-relevance of the resultant partial response.

Siamese Dual-Encoder: Finally, the context encoding is also fed to a Siamese network. To train the Siamese network, we randomly sample $k-1$ negative responses for each conversation context. The negative responses as well as the gold response are fed to a recurrent encoder (bidirectional LSTM) to generate the corresponding response embeddings. The context embedding as well as the corresponding response embeddings are fed to a 1-in-k classifier, where the k labels correspond to the k responses. The classifier is trained to predict the class-label that corresponds to the gold response. If the gold response r has label ℓ, the Siamese loss can be computed as follows:

$$loss_s(r) = -\log p(\ell | r_1, \ldots, r_k) \,. \qquad (2)$$

Multi-task training Objective: The final loss is the sum of the loss for each of the above models. The model is trained until the loss on an independent validation set stops decreasing.

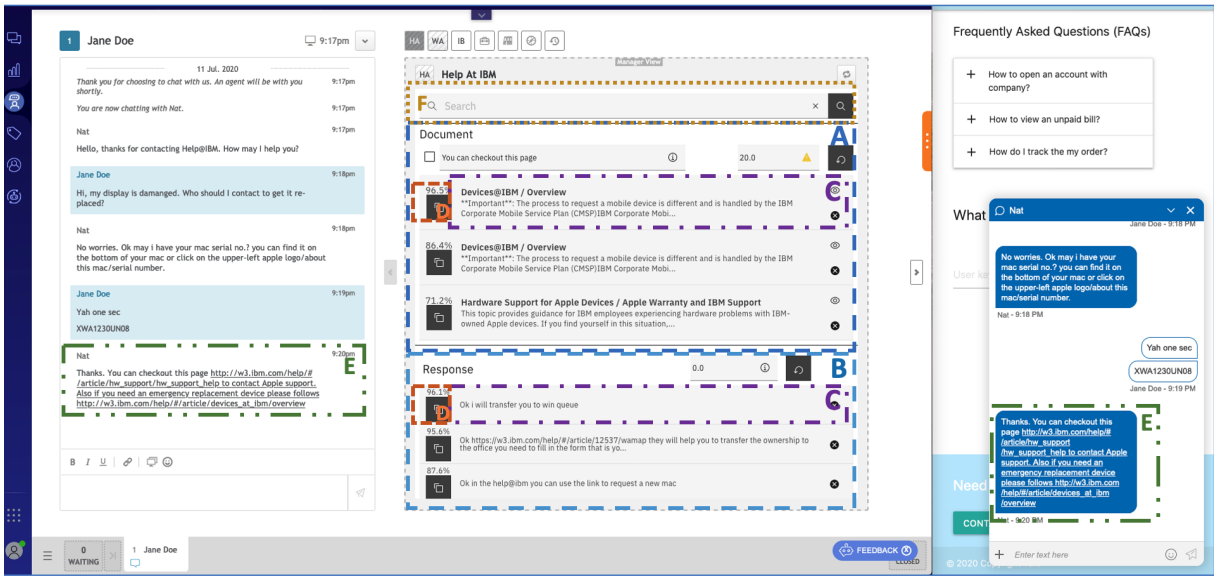

Figure 2: **CAIRAA In Action**; (A & B) Document and Response recommendations panel respectively; (C) Top Recommendation; (D) Confidence score of the recommendation and a button to copy URLs/responses into agent's chat with the end-user; (E) Agent's response; (F) Full-text search bar for agents to perform manual search.

2.3.2 Retrieval-based model

In order to build a retrieval based model, we encode the dialog contexts using the pre-trained Universal Sentence Encoder (USE) (Cer et al., 2018) as well as the context encoder trained using multi-task objective discussed above. For each encoder, we create an annoy index[3] which stores the context embeddings and the corresponding responses from the training data.

In order to return a response recommendation, a given dialog context is encoded either using USE or the context encoder. We fetch the responses of the k-nearest neighbours in the annoy index.

2.3.3 Scoring the responses

Before the retrieved and generated responses are presented to the user, they are scored from 0 to 1. We use a voting-based scoring mechanism, where each response votes for all the other responses (generated as well as retrieved). To achieve this, we encode each response using the pre-trained USE. The score of a response is the mean of the inner-product between the corresponding embedding and all the other response embeddings. Since USE embeddings are normalized, these inner products range between 0 and 1. Finally, the responses are sorted based on their scores and presented to the user.

[3]https://github.com/spotify/annoy

3 Deployment Details

Once the recommendation engines are trained, CAIRAA is tasked with the live operation of the agent dashboard. CAIRAA adopts optimized user interface design to deliver precise information with minimal agent interactions. Prioritizing scalability, each operational component is intentionally designed to be modular and stateless.

3.1 Agent Dashboard

The Agent dashboard (Figure 2) is an interactive web application that is integrated with customer support applications (e.g., LivePerson). Implemented using Javascript/ HTML5, it comprises of a chat interceptor; document and response recommendations panels; a full-text search; and a feedback facility runnable in an ES6 compliant browser. Chat interceptor monitors customer support applications (e.g., LivePerson) for conversation updates. Both document and response recommendations panels in their minimalist design limits information overload, where the former shows the title and a short description of the web content and the latter displays the recommended utterance. The agent interactivity with recommendations is limited to copy, view and reject. A full-text search allows the expert agents to perform simple keyword searches based on their domain knowledge.

While we have automated the agent assistance for recommending documents and responses, we

have retained the support agent in the loop to provide final edits and control over what is presented to the end-user. This allows a support agent to adjust the tone of responses, and to include documents outside recommended ones in their responses. Our framework captures these responses and their embedded recommendations for future retraining / retesting so that the system can self learn and automatically adjust to support agent activities.

4 Conclusion and Future Work

By providing real-time assistance to agents to support their clients, we leverage the speed and accuracy of automated recommendation engines while retaining the agents' expertise. Learning from conversation logs, CAIRAA promotes more uniform support by keeping all agents aware of the latest information to address current end-user needs. Combining traditional information retrieval approaches with modern deep learning models ensures high accuracy and efficient run-time performance in our deployed system.

References

Kate Acomb, Jonathan Bloom, Krishna Dayanidhi, Phillip Hunter, Peter Krogh, Esther Levin, and Roberto Pieraccini. 2007. Technical support dialog systems: issues, problems, and solutions. In Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies, pages 25–31. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. CoRR, abs/1803.11175.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 1657–1668.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01, page 120–127, New York, NY, USA. Association for Computing Machinery.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. arXiv preprint arXiv:1606.01541.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. arXiv preprint arXiv:1701.06547.

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. Dialogue & Discourse, 8(1):31–65.

Jiaul H. Paik and Douglas W. Oard. 2014. A fixed-point method for weighting terms in verbose informational queries. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM '14, page 131–140, New York, NY, USA. Association for Computing Machinery.

Haggai Roitman, Shai Erera, and Guy Feigenblat. 2019. A study of query performance prediction for answer quality determination. In Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR '19, page 43–46, New York, NY, USA. Association for Computing Machinery.

Haggai Roitman and Oren Kurland. 2019. Query performance prediction for pseudo-feedback-based retrieval. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'19, page 1261–1264, New York, NY, USA. Association for Computing Machinery.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In Thirtieth AAAI Conference on Artificial Intelligence.

Christophe Van Gysel, Evangelos Kanoulas, and Maarten de Rijke. 2016. Lexical query modeling in session search. In Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval, ICTIR '16, page 69–72, New York, NY, USA. Association for Computing Machinery.

Alexandra Vtyurina, Denis Savenkov, Eugene Agichtein, and Charles L. A. Clarke. 2017. Exploring conversational search with humans, assistants, and wizards. In Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA '17, page 2187–2193, New York, NY, USA. Association for Computing Machinery.

Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A cascade ranking model for efficient ranked retrieval. In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11, page 105–114, New York, NY, USA. Association for Computing Machinery.

Justin D Weisz, Mohit Jain, Narendra Nath Joshi, James Johnson, and Ingrid Lange. 2019. Bigbluebot: teaching strategies for successful human-agent interactions. In Proceedings of the 24th International Conference on Intelligent User Interfaces, pages 448–459.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. arXiv preprint arXiv:1604.04562.

Bin Xu, Jiajun Bu, Chun Chen, Deng Cai, Xiaofei He, Wei Liu, and Jiebo Luo. 2011. Efficient manifold ranking for image retrieval. In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11, page 525–534, New York, NY, USA. Association for Computing Machinery.