Reproduction Study: Controlling the Risk of Conversational Search via Reinforcement Learning

Niek IJzerman 11318740
Jeroen van Wely 11289988
Jochem Soons 11327030

Abstract

This reproduction study aims to reproduce recent work by Wang and Ai. Their paper proposes a novel risk-aware conversational search agent that incorporates the risk of asking wrong clarification questions to users. The proposed model contains a decision-maker module that is trained using reinforcement learning. This study assesses the claimed benefits of using such a framework by reproducing the original paper's results. We find that our study offers two main strengths: 1) by fixing errors and bugs in the original authors' code, we provide a repository that could be a starting point for future research, and 2) our results show that the results obtained by Wang and Ai are reproducible, although we find that results are not convincing enough to support all claims made in the paper. We conclude the study by discussing some recommendations for future research.

1 Introduction

An important reason for the retrieval of unreliable results within modern IR is the vague or immature formulation of queries by users. Several techniques have been proposed to address this problem. Examples are query auto-complete (Hu et al., 2015) and query suggestion (Sordoni et al., 2015). In particular, conversational retrieval in which a system asks clarifying questions to the user is believed to be a key technique for future IR (Radlinski and Craswell, 2017). However, previous work on conversational retrieval often assumes a dedicated user that would provide responses to any question asked by a retrieval system. This assumption is over-optimistic as it neglects the fact that bad questions could be asked, which could drive the user away from using the IR system again.

To address this problem, Wang and Ai (Wang and Ai, 2021) propose a risk-aware conversational retrieval system that is able to control and balance the risk of retrieval reliability and user engagement. More precisely, the authors introduce a conversational retrieval system consisting of three components: (1) A result retrieval module that retrieves and ranks candidate results according to the current information requests and corresponding context, (2) A question retrieval module that finds or generates clarifying questions for users and (3) a risk decision module that decides whether the system should ask the question provided by the question retrieval module or directly show the documents provided by the result retrieval module. This risk decision module is trained using reinforcement learning and is the key to controlling and balancing the risk of asking poor questions or providing low-quality results to users.

The contributions of the authors are stated as following: (1) They propose a new risk-aware conversational agent which takes the risks into consideration when deciding to ask clarifying questions or answer the query given conversation context. (2) They introduce a Reinforcement Learning approach to train proposed agent without having annotated data for when to ask clarifying question and when to give answer. (3) They show that their risk-aware conversational search agent could improve both answer quality and user experience in interacting with the retrieval system.

This study aims to critically assess and reproduce the paper of Wang and Ai, and is further divided into six sections. In section 2, we explain the main concept of the proposed risk-aware agent. In section 3, we discuss the methodology of reproducing results. In section 4, we provide and analyze the results of our study. In section 5, we briefly discuss an extension made to the framework. Finally, in section 6 and 7, we conclude and discuss our reproduction study, and we provide some recommendations for future research.

2 Risk-aware user-agent model

In this section, we will discuss the interactive user-agent model proposed by Wang and Ai. The risk-aware conversation agent aims to answer a user's query correctly. It will do so by entering a conversation with the user. A conversation is composed of turns. Each turn starts with the information provided by the user and ends with a reaction by the

agent. The start of a conversation, turn one, is always the user's initial query or question. From that point on, the agent can choose one out of two actions. Either asking a clarifying question to ensure the agent "understands" the question better or answering the query question with the current context information already available. Either one of these actions will end the current turn. When the agent asks a question, the user can react by answering the question or leaving the conversation. When the agent provides an answer, the conversation ends. Each turn will thus start with new context information, which is either the user's query or the user's answer to the clarifying question asked by the agent in the previous turn.

The framework proposed by Wang and Ai essentially consists of three components: the question reranker, the answer reranker, and the risk-aware decision module. The rerankers (section 2.1) will first rank the answers and clarifying questions based on the current context information. The decision-maker (section 2.2) then uses the conversational context and rankings scores to decide whether to ask a clarifying question or answer the user's query question. In the following sections, We will go into more detail for each of the individual components of the framework.

2.1 Answer and Question Reranker

As stated above, the goal of the answer and (clarifying) question reranker is to retrieve the best answer and question given the current context information. To demonstrate that the risk-aware decision-maker (section 2.2) can work with any answer and (clarifying) question reranker, Wang and Ai present the model's performance using two different rerankers. These are the ParlAI bi- and poly-encoder rerankers (Humeau et al., 2019). For both encoder implementations of the model, the same reranker structure is used with different sets of parameters. Which parameter settings have been used will be discussed in section 3.2.

If both encoders are compared, the poly-encoder tends to have the highest performance while maintaining reasonable efficiency. It uses a similar structure to the bi-encoder, where both the context and question/answer are encoded separately. However, it extends this system with an additional mechanism where the input context is jointly encoded with the question/answer encoding. Therefore, the bi-encoder is slightly faster. However, the performance is noticeably lower in both the original paper (Humeau et al., 2019), and the paper by Wang and Ai (Wang and Ai, 2021).

2.2 Decision making model

The risk-aware decision-maker module is essentially the core component of the framework proposed by Wang and Ai. The module uses a Deep Q Network (DQN) to decide whether to answer the user's query question or ask a clarifying question. The DQN first uses a BERT encoder to encode the query question (q), context information (h), and the top-k ranked clarifying questions ($cq^1, ..., cq^k$) and answers ($a^1, ..., a^k$). Together with the top-k clarifying questions and answers scores ($s_{cq}^{1:k}$, and $s_{ans}^{1:k}$) respectively), these encodings are concatenated and given as input to a two-layer feed-forward neural network. We define the input by:

$$S = (q, h, cq^1, ..., cq^k, a^1, ..., a^k, s_{cq}^{1:k}, s_{ans}^{1:k}) \quad (1)$$

And subsequently we can represent the two-layer DQN network as:

$$DQN(S) = W_2 \cdot \phi\left(W_1 \cdot S + b_1\right) + b_2 \quad (2)$$

Where ϕ is a ReLU activation function. The network will output a prediction of the rewards for asking a clarifying question and answering the query question in the form of a 2×1 vector $y_{\text{pred}} = (r_{\text{ans}}, r_{cq})$. The agent's action to take in the conversation is subsequently chosen using an epsilon-greedy policy over these expected rewards of either asking a clarification question or providing an answer.

2.3 Training procedure

Rerankers. Both the question- and answer-reranker are pretrained on the Reddit dataset **[bron]**. They are then finetuned on the answer and question datasets, respectively. During tuning, the training set is divided into batches of 100 conversation-response pairs, of which only the true response is used as a positive sample. All other responses are used as negative samples. Both rerankers are trained to minimize the cross-entropy loss between the batch relevance label vector and the batch similarity logits vector. The batch relevance label vector will have a single one corresponding to the true response. All other values will be zero. The batch similarity vector is a vector containing the scores

for each response. Such a score is computed by the Bi- and Poly-encoders.

Decision Maker. The DQN is trained via Reinforcement Learning by simulating the interactions between user and agent and defining a reward (r) for each action (a) made by the DQN. Note that a reward can also be negative and that an action is either answering the user's query question or asking a clarifying question. Additionally, the user can be modeled with different bad question tolerance levels. The authors have modeled users with 0-tolerance (user leaves after a single bad question), 1-tolerance (user tolerates one bad question but leaves if a second bad question is asked), and 2-tolerance (user tolerates two bad questions). This means that the agent can only ask so many bad questions in each turn before the user leaves and thus ends the conversation. The rewards for each action are defined as follows:

- When action a is to answer the user's query question, the conversation ends, and the DQN receives a reward r equal to the Mean Reciprocal Rank (MRR) of the top-k reranked answers by the answer-reranker. Because there is only one correct answer in the top-k reranking, the reward will therefore always be equal to $\frac{1}{r}$ where r is the rank of the true answer within the reranking. For example, with a batch size of 10, the maximum reward for answering will be 1, and the minimum will be $\frac{1}{10}$.
- When action a is to ask a clarifying question, and the correct question's rank r is lower than or equal to the bad question tolerance of the user, a reward will be given of r_{cq}. This value is a hyperparameter set by the authors to $r_{cq} = 0.21$
- When action a is to ask a clarifying question, and the correct question's rank r is higher than the bad question tolerance of the user, a penalty will be given of p_{cq}. This value is again a hyperparameter that is set by the authors to $p_{cq} = r_{cq} - 1 = -0.79$.

The goal of the DQN is to predict the reward r of taking action a given its input state S where S is previously defined in equation 1. It does this by minimizing the mean squared error (MSE) loss between the rewards prediction $y_{\text{pred}}\,(S)_a$ and its target $y_{\text{target}}\,(S)_a$ during training. The loss that is minimized is therefore defined as:

$$L = \text{MSE}\left(y_{\text{target}}\,(S)_a, y_{\text{pred}}\,(S)_a\right) \tag{3}$$

The predicted reward is defined by the output of the DQN:

$$y_{\text{pred}}\,(S)_{\text{a}} = DQN(S)_{\text{a}} \tag{4}$$

Note that the output of the DQN is a 2×1 vector $y_{\text{pred}} = (r_{\text{ans}}\,, r_{cq})$. The target reward is defined by:

$$y_{\text{target}}(S)_a = r_a + \gamma \cdot \max_{a'=ans,cq} DQN(S')_{a'} \tag{5}$$

Where r_a is the obtained reward by taking action a in state S, γ is a discount factor which controls how much we care about future actions, S' is the new state we are in after taking action a in the original state S. $\max_{a'=ans,cq} DQN(S')_{a'}$ is the maximum predicted reward by our DQN given the new state S' for any action a that leads to the highest predicted reward. Note that if the current action a ends the conversation, S' is a terminal state. The predicted reward of a terminal state is always 0 and therefore if S' is a terminal state: $\max_{a'=ans,cq} DQN(S')_{a'} = 0$. This would be the case for the actions that are answering the user's query question or exceeding the bad question tolerance of the user. In those cases, the target reward becomes equal to just the reward obtained by taking action a in state S:

$$y_{\text{target}}\,(S)_{\text{ans}} = r_{\text{ans}} \tag{6}$$

$$y_{\text{target}}\,(S)_{cq} = p_{cq} \tag{7}$$

Only taking the action of asking a clarifying question and the correct questions rank not exceeding the user's bad question tolerance will not lead to a terminal state and therefore:

$$y_{\text{target}}(S)_{cq} = r_{cq} + \gamma \cdot \max_{a'=ans,cq} DQN(S')_{a'} \tag{8}$$

The DQN will start with a random exploration policy. Meaning it chooses its actions randomly given any state. Gradually the DQN will start choosing its actions not randomly anymore but start choosing them based on the highest predicted reward. The DQN is trained until it has converged.

3 Methodology: reproducing the results

Within this section, we will address each part of the reproduction process that has been conducted. For

each part, we will describe details and instructions from the paper and point out inconsistencies and mistakes we encountered when reproducing the research.

3.1 Datasets

Wang and Ai (Wang and Ai, 2021) use the MSDialog dataset (Qu et al., 2018), (Qu et al., 2019), (Yang et al., 2018) for their experiments. This dataset consists of question-answer conversations from an online forum for Microsoft products. This data is preprocessed as following:

1. The MSDialog data contains discussions where multiple agents are involved. Therefore, an assumption is made that all agents share the goal of correctly answering the user's question(s). Hence, the agents are merged into a single agent to ensure that each conversation only has two participants (i.e., one user and one agent).
2. All consecutive responses from a user are merged into a single response within a conversation to ensure that all conversations are conducted in turns.
3. After the above preprocessing steps, a filtering step is performed to ensure that all conversations are between four and ten turns long.
4. In the MSDialog dataset, each turn has a binary label indicating if this reply is voted the community's definitive answer. These labels are used in the final filter step that ensures all conversations end in a correct final answer.

All the conversations filtered out within the preprocessing steps are used to train and test the question rerankers. These are conversations either not having a correct final answer or conversations that are too long or too short. We ran the preprocessing steps and found our datasets statistics to be comparable to that of the original paper, although there is a slight difference in the number of conversations and average conversation length:

Table 1: MSDialog and our dataset statistics

Item	MSDialog	Orig. & Repr.
# conversations	35000	3762 / 3894
Max. turns	1700	10 / 10
Min. turns	3	4 / 4
Avg. turns	8.94	4.70 / 4.92

Additionally, we would like to make two points of critique concerning the handling of the data. Firstly, during training of the DQN, batch selection is performed deterministically. Each epoch uses the exact same batches in the exact same order. The authors may have implemented the training this way to reduce the computational load as making the batch selection deterministic allows reusing the question and answering embeddings. After epoch one, all embeddings have been computed and stored in memory. Each epoch after this can then reuse these embeddings by loading them from memory. However, it is doubtful whether deterministic batches in combination with a smaller batch size (e.g., a batch size of 10) improve training and subsequently the quality of the model as it results in a low diversity of negative samples. We will address this issue in section 5 as well.

Another point of concern is that only the original answer from the conversation during training is considered correct. However, it might be the case that answers from other conversations could also be considered correct, especially because most conversations seem to be centered around the same general topic: Microsoft. Additionally, some answers might not be entirely correct but at least "more correct" than others.

3.2 Baselines

In the original paper, five models are used as baselines. The first three are **Q0A**, **Q1A**, and **Q2A**, which are models that always answer the user's query question after 0, 1 and 2 question-answer turns, respectively. Additionally, **CtxPred** is used as a baseline as well. This is a risk-unaware model that first employs a binary classifier to predict whether to answer the query or ask a clarifying question given the current information context. It then uses these classification results to rank and select the top response by either the question- or answer-reranker. The code for CtxPred was not provided in the Github repository made publicly available by the authors. Therefore, we have not been able to reproduce the results for the CtxPred model. Finally, an **oracle model** is included that always knows when to answer the user's query question or ask a clarifying question. For all of these models, the same reranker and hyperparameter settings have been used.

3.3 Evaluation metrics

The evaluations metrics used by the authors are recall@1, Mean Reciprocal Rank (MRR), and the decision-making accuracy. We will now briefly

explain their definitions within the context of this paper.

The **recall@1** metric is defined as the frequency of conversations where the agent eventually gives the ground truth answer regardless of how many clarifying questions it has asked. Note that if the agent exceeds the bad-question-tolerance of the user, the conversation is terminated, and thus the agent can not answer the user's query question leading to a recall@1 score of 0. recall@1 is considered as the direct measure of the answer quality.

The **MRR** metric also does not consider the amount of clarifying questions proceedings the agent's answer. Note here that recall@1 evaluates the top answer, while MRR evaluates the entire reranking of length k. The ground truth is always present in this reranking, and therefore, the minimum MRR score for answering is $\frac{1}{k}$. However, suppose the agent exceeds the bad-question-tolerance of the user. In that case, the conversation is terminated, and thus the agent can not answer the user's query, which leads to an MRR score of 0. MRR, therefore, punishes the asking of bad questions more severely compared to answering the user's query with a wrong answer. Thus MRR should not be used as a primary evaluation metric because it is biased to answer the user's query instead of asking more questions.

The **decision error** measures the frequency of the agent making the wrong decision, where a wrong decision can either be exceeding the bad question tolerance of the user or answer the user's query question with the wrong answer. It, therefore, mainly measures the ability of the model to avoid risks.

3.4 Hyperparameters

The parameters used to obtain our results are presented in table 2. We used the same hyperparameters settings as in the original paper. The authors did not provide information about the number of epochs completed when training the decision-making module, but after some initial empirical investigation, we noticed fairly quick convergence during training and therefore set num_epochs$= 5$.

Table 2: Hyperparameter settings

Hyperparameter	Abbr.	Value
Good question reward	r_{cq}	0.21
Bad question penalty	p_{cq}	-0.79
Learning rate	lr	10^{-4}
Discount factor	γ	0.79
Regularization weight	λ	10^{-2}
Batch size	batch_size	10/100

We want to comment here about an inconsistency we encountered regarding the batch size used by the authors. In the original paper, the authors mention the usage of a batch size of 100, meaning that in each iteration, the question and answer candidate pool consists of one ground truth answer (the target) and 99 randomly sampled negatives. However, after our initial reproduction experiments using a batch size of 100, it became clear that these results were not in line with those reported in the original paper. After contact with the authors, it became apparent that the authors made a consistency error in their paper, as they obtained a large part of their results using a batch size of 10 instead of 100 (only their ablation study was performed using batch size 100). To further analyze this inconsistency and the effect of batch size on performance, we decided to reproduce results using batch size 10 and 100.

3.5 Technical details

The code of the authors was publicly available through GitHub[1]. The implementations for the bi- and poly encoder are from the ParlAI framework (Miller et al., 2017). The risk-aware Agent module (DQN) and User are written by the authors from scratch using Pytorch. We ran the code on the Lisa GPU server provided by the University of Amsterdam with 3 CPU cores, 64GB RAM and 1 GeForce 1080Ti, 11 GB GDDR5x GPU (see `https://userinfo.surfsara.nl/systems/lisa` for more information). Fine-tuning the poly- and bi-encoder rerankers took approximately 6 hours to complete while training the decision-maker (i.e., the main experiment) took approximately 5 hours.

3.6 Running the code

When we tried to run the code, we immediately experienced quite some errors. Firstly, to fine-tune the encoders, dependencies psutil, typing-extensions and egg-subword had to be installed additionally to the packages provided by the authors in their GitHub repository. Secondly, we had

[1] https://github.com/zhenduow/conversationalQA

to adjust the code in metric.py to avoid None value errors. Thirdly, in train_model.py within the directory conversationalQA-master/ParlAI/examples it was necessary to add the path to the ParlAI directory, which was not clear from the authors' instructions. Fourthly, we ran into dictionary size mismatch errors when the pretrained models were loaded. Although the authors provided a fix for this issue within their GitHub readme, it did not provide a full solution. We did notice via a GitHub forum that other researchers encountered the same issue and that the authors did provide the full solution within that thread. This fix, however, seems not to be updated in the authors' GitHub repository.

Training and evaluating the actual models by calling run_sampling.py resulted in some complications as well. Firstly, we noticed memory errors arising after training for some time, indicating a problem of memory leaks. After discussing this error with the authors of the original paper, they uploaded an updated version of their code that eliminated the memory errors. There were still some other minor errors that we had to fix before we could get the code working. First, we ran into a type error in agent.py as the targets for the model were ints while floats were expected. Secondly, the path to the ParlAI directory needed to be added run_sampling.py. We solved this in a flexible manner so that it does not need hard-coded path appends. Thirdly, the weight decay of the score agent was set to zero while it needed to be 0.01 to be in line with the paper. Fourthly, we encountered index out-of-range errors while updating the memory within the code. Lastly, we experienced that the code was not compatible to run on CPU. Therefore, we adapted the code to be compatible to run on both CPU and GPU (using CUDA).

3.7 Inspection of code pipeline

While the fixes mentioned in section 3.6 allowed us to train and evaluate the model, we also inspected the code pipeline more closely to determine if it was aligned with the theoretical pipeline as stated in the paper. This led to three interesting insights, which we will discuss in this section.

Firstly, the tolerance parameter was being initialized but not used within the code. Therefore, adjusting the tolerance parameter did not have any effect on the model training. Instead, all experiments were effectively ran with a tolerance of 0. We fixed this issue by adapting the code so that user tolerance could be modeled variably throughout the code.

Secondly, all models (i.e., the risk-aware conversational agent and baseline models Q0A, Q1A, Q2A) were evaluated in parallel. At the same time, if the authors' risk-aware conversational agent had finished a conversation (by either giving back the results or exceeding tolerance/patience) the conversation of the baseline models was also ended, and a penalty was given to these models. Because most conversations naturally ended in less than two turns, the evaluation scores of the Q1A and particularly Q2A were poor (see results of the paper in table 3). However, as this approach of evaluation and comparison of the baseline models against the proposed risk-aware model is unfair, we chose to adjust the pipeline such that baseline models could continue with the conversation, even if the risk-aware agent had already provided an answer earlier. Subsequently, this made the comparison of results between models fair.

Thirdly, concerning the ablation study of the authors where different variants of the risk-aware agent are compared (each agent has a different set of input features), we noticed that different agent variants would continue to learn from conversations even if they had already ended the conversation. For example, if an agent decided to answer the user query immediately at the first turn, but another agent continued the conversation by asking one or more clarification questions, all agents would continue to learn from this continuing experience. This creates a discrepancy between actual actions taken and experience from which is learned. Therefore, we decided to adjust the pipeline such that agent models stopped learning from a conversation when they ended it.

Altogether, we can conclude that the authors' original code was not immediately usable to reproduce results. After fixing the numerous errors reported in section 3.6 and the adjustments made after close inspection of the code as reported in this section, we have made our updated version of the repository also publicly available on GitHub[2].

4 Results and Analysis

In this section, we will discuss and analyze the reproduction results obtained in this study. We will discuss the results of four experiments following the set-up of the original paper, where we

[2] https://github.com/jochemsoons/risk_aware_conversationalQA

will closely compare results to those reported by Wang and Ai. For their experiments, the authors use 5-fold cross-validation to obtain their final results. However, because we ran into a scarcity of computational resources at the final stages of our study, we only managed to complete one run of each experiment.

4.1 Experiment 1: using poly-encoder as reranker

The results for the three different user tolerance levels using the poly-encoder as reranker, compared to the original results, are displayed in table 3. We can condense three interesting results from the table. First, we can notice that for the risk-aware agent, performance increases when user tolerance increases. This is in line with the result from the paper, and this indicates that the agent benefits from taking more risks, yielding higher performance. To further investigate if this increased performance actually comes from more questions being asked by the agent, we have plotted the average number of questions asked by the agent in figure 1. The figure confirms our intuition: for a higher user tolerance, the agent takes more risk and asks more questions, leading to higher performance.

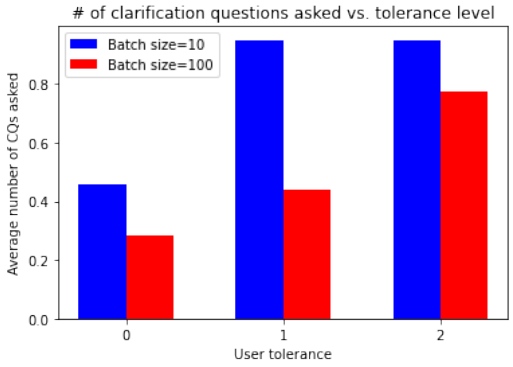

Figure 1: Number of clarification questions asked for different levels of user tolerance.

Second, we notice that as the tolerance level increases, the performance of the agent converges to the performance of the Q1A baseline. This is in line with what the authors also reported in the original paper. However, in our results, the Q1A baseline outperforms the agent for a tolerance of 2, while in the original paper the agent outperforms all baselines with a (significantly) small margin.

Third, it is also interesting to notice that the performance of the Q2A baseline model is significantly better than in the original paper, although it is still by far the worst-performing model. The gap in performance difference can be explained by the modifications we made to the training and evaluation pipeline as discussed in section 3.7.

Overall, we can conclude that results are comparable to those reported by Wang and Ai. However, where Wang and Ai state that their proposed risk-aware model outperforms the simple Q0A, Q1A, and Q2A baseline model, we feel that the results are not that convincing. In our result, the Q1A baseline model outperforms the proposed model multiple times. Additionally, when analyzing the results reported in the original paper, we notice that the performance difference between the proposed risk-aware model and the baseline counterparts is insufficient to conclude that the proposed model outperforms the other models (this is also supported by the statistical insignificance as reported by the authors). We also find it notable that the results of the CtxPred model obtained by the authors for tolerance levels 1 and 2 are identical to those of the risk-aware agent. The authors provide no explanation for why this happens, which further reinforces our doubts about the impressiveness of the performance of the risk-aware agent.

4.2 Experiment 2: varying batch size using poly-encoder

As discussed earlier in section 3.4, we also ran experiments using the poly-encoder using a batch size of 100, as this was the mentioned batch size throughout the original paper. The results of this experiment can be found in table 4. We can observe two interesting things from the table. First, it is noticeable that performance is significantly lower when compared to the results obtained using a batch size of 10 in table 3. This is not unsurprising, as using a larger batch size means that the rerankers have a larger pool of question and answer candidates that they have to rank. For a batch size of 10, we have 1 positive and 9 negatives, whereas with batch size 100, we have 1 positive and 99 negatives. Therefore, a larger batch size inevitably leads to a significant performance drop. We do find it important to show this effect, however, as it clarifies that the choice of batch size (a hyperparameter) in this experimental set-up has a crucial effect on performance. Therefore, results could be misleading when a small batch size is chosen, such as the batch size of 10 chosen by the authors in their paper. In any case, this result indicates the importance of

Table 3: Comparison of all models using poly-encoder with **batch size 10**. The original paper's results are presented in red. The reproduced results are presented in blue. Numbers in bold mean the results is the best excluding the oracle. † indicates $p < 0.01$ statistical significance over the best among baseline models.

Users	0 -tolerance			1-tolerance			2 -tolerance		
Models	R@1/10	MRR	Dec. err	R@1/10	MRR	Dec. err	R@1/10	MRR	Dec. err
Q0A	0.775 / 0.719	**0.840** / 0.823	0.198 / 0.224	0.745 / 0.719	0.834 / 0.823	0.198 / 0.262	0.748 / 0.719	0.840 / 0.823	0.198 / 0.272
Q1A	0.753 / 0.718	0.804 / 0.784	0.123 / 0.176	0.785 / 0.774	0.849 / 0.851	0.065 / **0.101**	0.820 / **0.788**	0.872 / **0.868**	0.225 / **0.096**
Q2A	0.008 / 0.102	0.008 / 0.213	0.993 / 0.776	0.008 / 0.219	0.008 / 0.245	0.992 / 0.743	0.008 / 0.221	0.008 / 0.248	0.993 / 0.742
CtxPred	0.740 / -	0.796 / -	0.128 / -	0.785 / -	0.853 / -	0.058 / -	0.820 / -	0.872 / -	0.023 / -
Agent	**0.778** / **0.764**	0.831 / **0.843**	**0.098**† / **0.144**	**0.788** / **0.775**	**0.853** / **0.854**	**0.058**† / 0.103	**0.820**† / **0.788**	**0.872**† / 0.863	**0.023**† / 0.100
Oracle	0.858 / 0.821	0.914 / 0.893	0 / 0	0.865 / 0.843	0.917 / 0.909	0 / 0	0.893 / 0.849	0.932 / 0.913	0 / 0

choosing the same batch size for different models to ensure a fair comparison.

Second, if we analyze the performance similar to the previous section, we notice the same effects. Performance of the agent increases when user tolerance increases, and performance also converges to performance of the Q1A model for higher tolerance levels. However, in this specific experiment, the Q1A model does not outperform the proposed model, although performance differences are again significantly small. Overall, we find that the results of this experiment lead to the same conclusion as the previous one: results of the proposed risk-aware agent are not convincing enough to conclude that the model clearly outperforms the simple baselines.

4.3 Experiment 3: using bi-encoder as reranker

In the original paper, the authors state that their decision-maker module can be extended to work with any reranking-module. To illustrate this, experiments have also been performed using the bi-encoder as the question and answer reranker. The result of this experiment, including our reproduced results, can be found in table 5. From the table, we can see that our obtained results are comparable to those of the original paper. Additionally, We can see similar patterns in results as earlier when analyzing the results of the poly-encoder. Once again, the performance of the agent increases when tolerance increases, but for a higher tolerance, the Q1A baseline outperforms the proposed model. Interestingly, our results obtained using the bi-encoder are comparable to the results using the poly-encoder in table 3, whereas the results of the authors are overall lower for the bi-encoder.

Overall, we can draw the same conclusion from table 5 as we could from the results of the earlier experiments: the proposed risk-aware agent does not clearly outperform the simple baselines. However, these results do indicate the capability of the risk-aware module to be used with different types of rerankers, which was the authors' primary motivation for adding the bi-encoder experiment.

4.4 Experiment 4: ablation study of input features

The decision-making deep Q network takes multiple features as input that can be separated into two types: 1) all encoded text features for the query, context, and question/answer candidates, and 2) the ranking scores from the question and answer rerankers. To investigate the effect of adding these different features, the authors have included an ablation study. For this ablation study, two variants of the DQN-based agent are created: one agent only takes the encoded text features as input, while the other only takes the ranking score features as input. Subsequently, we can compare results against the agent model that takes both types of features as input (i.e., the risk-aware agent model that we have analyzed until now). We have reproduced this ablation study with the results shown in table 6.

From the table, we see comparable results to those reported in the original paper. For almost all tolerance levels and metrics, we see that the text + score agent outperforms the agents that take only one feature type as input. This is in line with the result in the paper, and it implies that all features play important roles in the decision-making module. This result further illustrates that the decision-making module design is non-trivial, in the sense that a reinforcement learning-based approach benefits from combinations of features and that, for example, only using ranking scores is insufficient for obtaining maximum performance.

5 Extension: BM25 negative sampling

While the authors state that conversations to sample negative answers and questions from are chosen randomly from the dataset, these conversations are picked by deterministically looping over the train

Table 4: Comparison of all models using poly-encoder with **batch size 100**. Numbers in bold mean the results is the best excluding the oracle.

Users	0 -tolerance			1-tolerance			2 -tolerance		
Models	R@1/100	MRR	Dec. err	R@1/100	MRR	Dec. err	R@1/100	MRR	Dec. err
Q0A	0.418	0.543	0.357	0.418	0.542	0.440	418	0.543	0.478
Q1A	0.376	0.453	0.404	0.440	0.533	0.301	0.464	0.565	**0.256**
Q2A	0.078	0.094	0.876	0.094	0.118	0.838	0.104	0.130	0.826
Agent	**0.434**	**0.553**	**0.282**	**0.467**	**0.575**	**0.286**	**0.474**	**0.573**	**0.256**
Oracle	0.528	0.640	0	0.552	0.664	0	0.564	0.674	0

Table 5: Comparison of all models using bi-encoder with **batch size 10**. The original paper's results are presented in red. The reproduced results are presented in blue. Numbers in bold mean the results is the best excluding the oracle. † indicates $p < 0.01$ statistical significance over the best among baseline models.

Users	0 -tolerance			1-tolerance		
Models	R@1/10	MRR	Dec. err	R@1/10	MRR	Dec. err
Q0A	0.673 / 0.713	**0.798** / 0.820	0.243 / 0.238	0.673 / 0.704	0.798 / 0.816	0.243 / 0.235
Q1A	0.720 / 0.729	0.778 / 0.820	0.140 / 0.175	**0.770** / **0.781**	**0.835** / **0.868**	0.073 / **0.065**
Q2A	0.008 / 0.208	0.008 / 0.227	0.993 / 0.763	0.008 / 0.231	0.008 / 0.244	0.993 / 0.722
CtxPred	0.720 / -	0.778 / -	0.135 / -	0.768 / -	0.833 / -	0.070 / -
Agent	**0.738**† / **0.776**	**0.798** / **0.852**	**0.123**† / **0.146**	0.760 / 0.776	0.834 / 0.865	**0.065**† / 0.083
Oracle	0.820 / 0.836	0.890 / 0.904	0 / 0	0.860 / 0.850	0.916 / 0.913	0 / 0

and test set. Furthermore, the procedure to sample negative answers and questions from these conversations is deterministic as well. Subsequently, the batches used for training are the same for each epoch. To account for a possibly better training pipeline, we have implemented a sampling framework that utilizes the BM25 algorithm to sample informative negatives. Using this framework, we no longer sample negative questions and answers from conversations for training locally, but construct our negatives globally by taking the best matching BM25 conversations compared to the true positive conversation. Subsequently, we subtract negative questions and answers from these best-matched conversations to create batches.

Subsequently, the created batches contain more informative negatives that are semantically closer related to the true positive conversation. We expect that this change of sampling negatives improves the distinctiveness of the model for positive and negative instances and therefore leads to better scores.

As our reproduction timeline and computational scarcity at the end of our project did not allow us to obtain all results using the BM25 pipeline, we could not include the results of applying this extension that would allow a fair comparison of the models with and without BM25 negative sampling. We did, however, finish the code to produce results using BM25 sampled negatives, and we have included this in our GitHub repository. This way, follow-up research might easily leverage our code to run and analyze the BM25 model's results.

6 Conclusion

This reproduction study analyzed whether the risk-aware conversational search agent proposed by Wang Ai brings the benefits as claimed in the original paper. To achieve this, we have reproduced nearly all the experiments that the authors have performed. We can draw several main conclusions from this work. Firstly, We can conclude that it was a significant challenge to produce results that follow the theoretical framework as explained in the original paper. This challenge involved solving numerous code errors and bugs, but also adapting the code so that the training procedure matches that of the paper and comparison against baselines was done fairly. Additionally, by thoroughly investigating the authors' code and the paper, we have identified some inconsistencies throughout the original paper that have been brought to the authors' attention.

Secondly, throughout our results section, we could conclude that results were comparable to those reported in the original paper. However, we find that our results are slightly less convincing than the original results, as, in multiple experiments, relatively simple baseline models managed to outperform the proposed model. Altogether, we feel that this reproduction study highlights that there is too little evidence to support the claim that the risk-aware agent proposed by Wang and Ai brings sub-

Table 6: Ablation study results using poly-encoder as reranker with **batch size 100**. The original paper's results are presented in red. The reproduced results are presented in blue. Numbers in bold mean the result is the best performing. † and ‡ indicates $p < 0.1$ and $p < 0.01$ statistical significance over the best of encoded text and score models.

Users	0 -tolerance			1-tolerance			2-tolerance		
Models	R@1/100	MRR	Dec. err	R@1/100	MRR	Dec. err	R@1/100	MRR	Dec. err
Encoded text	0.392 / 0.414	0.488 / 0.529	0.324 / 0.328	0.438 / 0.439	0.528 / 0.531	0.206 / 0.303	0.461 / 0.464	0.559 / 0.565	0.163 / **0.256**
Score	0.442 / **0.441**	0.543 / 0.551	0.246 / 0.292	0.452 / 0.447	0.552 / 0.552	0.235 / 0.330	0.448 / 0.458	0.551 / 0.560	0.226 / 0.353
Encoded text + Score	**0.446** / 0.434	**0.544** / **0.556**	**0.236**† / **0.282**	**0.466**‡ / **0.467**	**0.562**‡ / **0.575**	**0.183**‡ / **0.286**	**0.479**‡ / **0.474**	**0.578**‡ / **0.573**	**0.152**‡ / **0.256**

stantial benefits over simpler methods that do not explicitly take risk into account. However, we think the proposed method is an interesting and original idea that can still inspire follow-up research, which we will discuss further in the following section.

7 Discussion & Further Research

We feel that this reproduction study has two main strengths. First, by fixing errors in the original authors' code and publishing our repository online, we have provided a starting point for future research that might follow up on the model proposed by Wang and Ai. Second, our results show that the results obtained in the original paper are reproducible, although we conclude that they are not convincing enough to support all claims made by the authors.

We would also like to point out some weaknesses in our work and include some recommendations for future research. First, we would have liked to perform the same 5-fold cross-validation scheme as in the original paper to ensure that our results are reliable. This was unfortunately not possible due to computational and temporal constraints. Second, our work reproduced the experiments reported in the original paper but did not include results of extensions on the paper. Overall, we think that incorporating risk using Reinforcement Learning (RL) within conversational search is a very intriguing idea. We think there are many possibilities of extending Wang Ai's research. It would be interesting to analyze the performance of this model on other datasets. Currently, the MSDialog dataset is limited to the subject of Microsoft products only. Using a dataset that contains multiple different topics might be interesting. Additionally, it might be interesting to compare and contrast performance when different RL-based algorithms are used to form a decision-maker module. Finally, as discussed in section 5, we have initialized an extension on the research by incorporating BM25 negative sampling that is readily available on our GitHub repository. However, this method of sampling is based on sparse information. We think the incorporation of negative sampling using dense retrieval could improve the performance of the model.

8 Acknowledgments

We want to thank Mohammad Aliannejandi and Evangelos Kanoulas for the opportunity, guidance, and supervision during this reproducibility work. We are especially thankful for the extension granted to complete our reproduction study. We would also like to extend our gratitude to Zhenduo Wang and Qingyao Ai. By quickly responding to our questions, they have been very helpful during the process of conducting this study.

References

Wenyan Hu, Xiaodi Zhang, Alvaro Bolivar, and Randall Scott Shoup. 2015. Predictive algorithm for search box auto-complete. US Patent 8,990,240.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*.

A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. 2017. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*.

Chen Qu, Liu Yang, W Bruce Croft, Johanne R Trippas, Yongfeng Zhang, and Minghui Qiu. 2018. Analyzing and characterizing user intent in information-seeking conversations. In *The 41st international acm sigir conference on research & development in information retrieval*, pages 989–992.

Chen Qu, Liu Yang, W Bruce Croft, Yongfeng Zhang, Johanne R Trippas, and Minghui Qiu. 2019. User intent prediction in information-seeking conversations. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, pages 25–33.

Filip Radlinski and Nick Craswell. 2017. A theoretical framework for conversational search. In *Proceedings*

of the 2017 conference on conference human information interaction and retrieval, pages 117–126.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *proceedings of the 24th ACM international on conference on information and knowledge management*, pages 553–562.

Zhenduo Wang and Qingyao Ai. 2021. Controlling the risk of conversational search via reinforcement learning. In *Proceedings of the Web Conference 2021*, pages 1968–1977.

Liu Yang, Minghui Qiu, Chen Qu, Jiafeng Guo, Yongfeng Zhang, W Bruce Croft, Jun Huang, and Haiqing Chen. 2018. Response ranking with deep matching networks and external knowledge in information-seeking conversation systems. In *The 41st international acm sigir conference on research & development in information retrieval*, pages 245–254.