When and What to Ask Through World States and Text Instructions

IGLU NLP Challenge Solution

Zhengxiang Shi* **Jerome Ramos*** **To Eun Kim** **Xi Wang**
Hossein A. Rahmani **Aldo Lipani**
{zhengxiang.shi.19, jerome.ramos.20, to.kim.17, xi-wang
hossein.rahmani.22, aldo.lipani}@ucl.ac.uk
Web Intelligence Group
University College London, London, United Kingdom

1 Introduction

In collaborative tasks, effective communication is crucial for achieving joint goals (Fu et al., 2022). One such task is collaborative building (Narayan-Chen et al., 2019) where builders must communicate with each other to construct desired structures in a simulated environment such as MINECRAFT. We aim to develop an intelligent builder agent to build structures based on user input through dialogue. However, in collaborative building, builders may encounter situations that are difficult to interpret based on the available information and instructions, leading to ambiguity. In the NeurIPS 2022 Competition NLP Task (Kiseleva et al., 2022), we address two key research questions, with the goal of filling this gap: when should the agent ask for clarification, and what clarification questions should it ask? We move towards this target with two sub-tasks, a classification task and a ranking task. For the classification task, the goal is to determine whether the agent should ask for clarification based on the current world state and dialogue history. For the ranking task, the goal is to rank the relevant clarification questions from a pool of candidates. In this report, we briefly introduce our methods for the classification and ranking task. For the classification task, our model achieves an F1 score of 0.757, which placed the 3rd on the leaderboard. For the ranking task, our model achieves about 0.38 for Mean Reciprocal Rank by extending the traditional ranking model. Lastly, we discuss various neural approaches for the ranking task and future direction.

2 Classification Task

In this section, we introduce the proposed builder model, built upon Shi et al. (2022), as shown in Figure 1, and then present the experimental results.

*Equal Contribution

2.1 Method

The model comprises four major components: the *utterance encoder*, the *world state encoder*, the *fusion module*, and the *slot decoder*. The utterance encoder (§2.1) and world state encoder (§2.1) learn to represent the dialogue context and the world state. These encoded representations are then fed into the fusion module (§2.1) that learns contextualized embeddings for the grid world and textual tokens through the single and cross-modality modules. Finally, the learned world and text representations are mapped into scalar values for the binary classification (§2.1).

Dialogue Context Encoder. We add "architect" and "builder" annotations before each architect utterance A_t and each builder utterance B_t respectively. Then, the dialogue utterances are represented as

$$D_t = \text{“architect”}A_t \oplus \text{“builder”}B_t$$

at the turn t, where $\oplus$ is the operation of sequence concatenation.The entire dialogue context is defined as:

$$H = D_1 \oplus D_2 \oplus \cdots \oplus D_t \quad (1)$$

Given the dialogue context H, we truncate the tokens from the end of the dialogue context or pad them to a fixed length as inputs and then use the dialogue context encoder to encode utterance history into $U \in \mathbb{R}^{s \times d_w}$, where d_w is the dimension of the word embedding and s is the maximum number of tokens for a dialogue context. BERT (Devlin et al., 2019) is chosen as the dialogue context encoder.

Grid World State Encoder. The world state is represented by a voxel-based grid. We first represent each grid state as a 7-dimensional one-hot vector that stands for empty or a block having one of six colours, yielding a 7×11×9×11 world state representation.

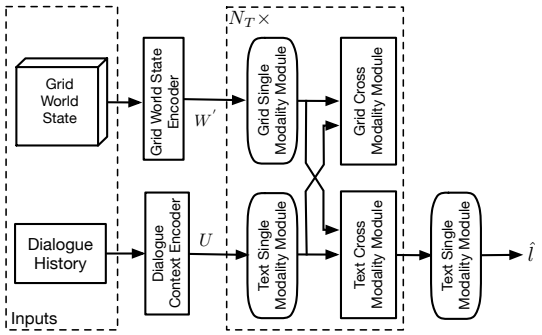

Figure 1: Model Architecture for the classification task.

The structure of the world state encoder is similar to Jayannavar et al. (2020)'s, that is, consisting of k 3D-convolutional layers (f_1) with kernel size 3, stride 1 and padding 1, followed by a ReLU activation function. Between every successive pair of these layers there is a 1×1×1 3D-convolutional layer (f_2) with stride 1 and no padding followed by ReLU:

$$W_i = \text{ReLU}(f_2^i(\text{ReLU}(f_1^i(W_{i-1})))), \quad (2)$$

$$W_k = \text{ReLU}(f_1^i(W_{k-1})), \quad (3)$$

where $i = 1, 2, \ldots, k-1$. $W_k \in \mathbb{R}^{d_c \times 11 \times 9 \times 11}$ is the learned world grid-based representation where d_c is the dimension of each grid representation. Then we reshape W_k into $W' \in \mathbb{R}^{d_c \times 1089}$.

Fusion Module. The fusion module comprises four major components: one *single modality modules* and two *cross-modality modules*. The former modules are based on self-attention layers and the latter on cross-attention layers. These take as input the world state representation and dialogue history representation. Between every successive pair of grid single-modality modules or text single-modality modules, there is a cross-modality module. We take N_T layers for cross-modality modules. We first revisit the definition and notations about the attention mechanism (Bahdanau et al., 2015) and then introduce how they are integrated into our single-modality modules and cross-modality modules.

Attention Mechanism. Given a query vector x and a sequence of context vectors $\{y_j\}_{j=1}^K$, the attention mechanism first computes the matching score s_j between the query vector x and each context vector y_j. Then, the attention weights are calculated by normalizing the matching score: $a_j = \frac{exp(s_j)}{\sum_{j=1}^K exp(s_j)}$. The output of an attention layer is the attention-weighted sum of the context vectors: $Attention(x, y_j) = \sum_j a_j \cdot y_j$. Particularly, the attention mechanism is called self-attention when the query vector itself is in the context vectors $\{y_j\}$. We use the multi-head attention following (Devlin et al., 2019).

Single-Modality Module. Each layer in a single-modality module contains a self-attention sub-layer and a feed-forward sub-layer, where the feed-forward sub-layer is further composed of a linear transformation layer, a dropout layer and a normalization layer. We take N_T and N_T+1 layers for the grid single-modality modules and the text single-modality modules respectively, interspersed with cross-modality module as shown in Figure 1. Since new blocks can only be feasibly placed if one of their faces touches the ground or another block in the Minecraft world, we add masks to all infeasible grids in the grid single-modality modules. For a set of text vectors $\{u_i^n\}_{i=1}^s$ and a set of grid vectors $\{w_j^m\}_{j=1}^{1089}$ as inputs of n-th text single-modality layer and m-th grid single-modality layer, where $n \in \{1, \ldots, N_T+1\}$ and $m \in \{1, \ldots, N_G+1\}$, we first feed them into two self attention sub-layers:

$$u_i^n = \text{SelfAttn}_u^n(u_i^n, \{u_i^n\}), \quad (4)$$

$$w_j^m = \text{SelfAttn}_w^m(w_j^m, \{w_j^m\}, mask) \quad (5)$$

Lastly, the outputs of self attention modules, u_i^n and w_j^m, are followed by feed-forward sub-layers to obtain $\hat{u_i}^n$ and $\hat{w_j}^m$.

Cross-Modality Module. Each layer in the cross-modality module consists of one cross-attention sub-layer and one feed-forward sub-layer, where the feed-forward sub-layers follow the same setting as the single-modality module. Given the outputs of n-th text single-modality layer, $\{\hat{u_i}^n\}_{i=1}^s$, and the m-th grid single-modality layer, $\{\hat{w_j}^m\}_{j=1}^{1089}$, as the query and context vectors, we pass them through cross-attention sub-layers, respectively:

$$\hat{u_i}^{n+1} = \text{CrossAttn}_u^n(\hat{u_i}^n, \{\hat{w_j}^m\}), \quad (6)$$

$$\hat{w_j}^{m+1} = \text{CrossAttn}_w^m(\hat{w_j}^m, \{\hat{u_i}^n\}), \quad (7)$$

The cross-attention sub-layer is used to exchange the information and align the entities between the two modalities in order to learn joint cross-modality representations. Then the output of the cross-attention sub-layer is processed by one feed-forward sub-layer to obtain $\{u_i^{n+1}\}_{i=1}^s$ and $\{w_j^{m+1}\}_{j=1}^{1089}$, which will be passed to the following singe-modality modules.

Finally, we obtain a set of word vectors, $\{\hat{u_i}^{N_T+1}\}_{i=1}^{s}$, and a set of grid vectors, $\{\hat{w_j}^{N_T}\}_{j=1}^{1089}$. We take word representations as U^{N_T}.

Slot Decoder. The Slot Decoder contains one linear projection layers of trainable parameters W. We compute the average of $U^{N_T} \in \mathbb{R}^{s \times d_w}$ alongside the s-dimension to obtain $u \in \mathbb{R}^{d_w}$. Then we obtain a scalar value for the binary classification via: $\hat{l} = \text{Sigmoid}(W \cdot u)$. We train the model by minimizing the cross-entropy loss.

2.2 Results

Table 1 presents the performance of all comparison approaches for the classification task. Our proposed method achieves a 75.7% F_1 score and secures the 3rd position among all the compared approaches. It is important to note that the performance difference between the teams is relatively small.

3 Ranking Task

In this section, we introduce our proposed method for the ranking task. We have made several efforts on improving the ranking task, including expanding the classical ranking methods, query expansion, and neural approaches. Details are as follows.

3.1 Grid Search of BM25

First, to find the best parameter of the BM25 ranker, we conducted a grid search of the two parameter `k1` and `b`. From the grid search, we have found that `k1=1.8` and `b=0.98` can perform the best in the validation set.

3.2 Use of Classic Ranking methods with heuristic

Next, we have explored multiple classic ranking techniques from Information Retrieval, including the TF-IDF, PL2, DPH weighting models as well as reproducing the results of BM25.

We implemented additional preprocessing by changing all characters to lowercase and removing punctuation. In addition, we added a **heuristic** for all clarification questions that scored a BM25 score of zero. We ranked all questions with a score of zero by the number of words, in ascending order. The reasoning is that shorter questions are more general and are able to answer a wider variety of questions, thus making it more likely to be the correct answer. Furthermore, long questions are more likely to contain more unique words, which means that it is highly unlikely to be the correct answer if none of the words matched the query. We found that these two steps improved the performance of the ranking task, and the results were as follows:

```
MRR5:  0.36846441947565517
MRR10: 0.38006777242732276
MRR20: 0.388398839553438
```

3.3 Query Expansion

In order to improve the ranking performance, we experimented with a more recent approach of ranking techniques. Query expansion techniques have shown their advantages in improving the recall performance of a ranking model. In our experiments, we explored the use of the Bo1, KL divergence-based and RM3 query expansion techniques in a re-ranking setup with a TF-IDF weighting model. In addition, we also extend the query expansion technique with a fine-tuned T5 model. A T5 model is trained on instruction-clarification question pairs and we aim to automatically generate possible clarification questions to be added to a given instruction (i.e., query) for query expansion and then leverage various classic weighting models (TF-IDF, BM25, PL2, DPH) for clarification question (document) retrieval.

3.4 Reranking with Semantic Textual Similarity (Semantic Search)

We used a Sentence Transformer's RoBERTa model as a sentence encoder to get a Semantic Textual Similarity (STS) between instructions and clarifying questions. We encoded instructions and clarifying questions separately but with the same encoder. Then, the STS could be computed with the cosine similarity between the embedding of instruction and question. After we first rank the questions with the BM25, we reranked the top 20 of the result with the score of STS.

3.5 Reranking with BERT model

We attempted to do reranking using BERT and RoBERTa models. We encoded the instruction input and clarification question and trained a classifier to calculate the probability that the instruction input matches the clarification question. We then reranked the clarification questions collected by BM25 by the highest probability. Given the high skew towards negative samples, we decided to have the dataset contain 33% positive labels and 66% negative labels. However, since the size of

Team Name	Rank	F1 Score (Nearest Bin)	F1 Score	Successful Entries
try1try	1	0.750	0.766	39
felipe_B	2	0.750	0.761	54
Ours	3	0.750	0.757	39
testa	4	0.750	0.756	29
745H1N	5	0.750	0.756	14
ITL-ed	6	0.750	0.754	66
craftsmanfly	7	0.750	0.751	74
ITL-ed	8	0.750	0.754	66

Table 1: Test Results from the IGLU NLP Task Leaderboard. Successful Entries stand for the number of trials. Our proposed method is highlighted in blue.

the dataset is small, we found it difficult to train a reranker model that significantly improved ranking performance.

3.6 Use of Bi-encoder and Cross-encoder

We attempted various combinations of ranker and reranker. First, we tried using a bi-encoder only for the ranking task. The bi-encoder model is *sentence-transformers/multi-qa-MiniLM-L6-cos-v1*[1], which was trained for semantic search.

The ranking results were like the following:

```
MRR5:  0.2026591760299625
MRR10: 0.21386614945603696
MRR20: 0.2222604358213725
```

Next, we added a cross-encoder for the reranker. Two models we have tried are *cross-encoder/ms-marco-MiniLM-L-12-v2* and *cross-encoder/ms-marco-MiniLM-L-6-v2*[2], which are both trained on the MS Marco Passage Ranking task. The ranking results were improved compared to the previous attempt.

```
MRR5:  0.26166666666666677
MRR10: 0.27329498840734795
MRR20: 0.28037113927919943
```

Since BM25 is still a great retriever, we used BM25 as an initial ranker and use a cross-encoder as reranker. The score was improved as follows:

```
MRR5:  0.2841760299625468
MRR10: 0.2980809702158013
MRR20: 0.30612438951917825
```

4 Discussion and Future Direction

Classification Task. To further enhance the model performance, it may be beneficial to explore larger language models (Kaplan et al., 2020; Hoffmann et al., 2022), prompting-oriented methods (Schick and Schütze, 2021), or further pre-training (Gururangan et al., 2020; Shi and Lipani, 2023) in future research.

Ranking Task. Although we have experimented with recent approaches, the BM25 with added heuristic performed the best. We will have to research more about the reason behind it in the future. In the literature, advanced dense retrieval approaches have also been experimented with, including MonoT5 and ColBERT. We plan to continue our research with these models on the IGLU competition dataset.

Generation of Instructions and Clarifying Questions. As an additional task, we can try to generate pairs of instruction and clarifying questions. The generation of those can help augment the training data for the classification and ranking models.

Not only that, this direction aligns with the recent research in user simulation (Kim and Lipani, 2022) and Theory of Mind (Sap et al., 2022), where they view conversation as a cooperative and collaborative mind game based on the study of *pragmatics*. According to this view, it is important for a conversational system to be able to model users (listeners) (Fried et al., 2018), i.e., endowing a conversational system with an ability to reason counterpart's mental states and realities of the surroundings. With this aspect, clarifying question generators can act as a user (Builder) simulator in the interactive conversational environment, enabling the Architect to generate instructions that are less ambiguous to the Builder. Moreover, it is worth exploring if we can find a meaningful relationship between the satisfaction level of the Builder and the need for clarifying questions. Estimating the Builder's satisfaction level, following the work of Kim and Lipani (2022), may help classify the need for clarifying questions.

[1]https://huggingface.co/sentence-transformers
[2]https://huggingface.co/cross-encoder

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018. Unified pragmatic models for generating and following instructions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1951–1963, New Orleans, Louisiana. Association for Computational Linguistics.

Xiao Fu, Emine Yilmaz, and Aldo Lipani. 2022. Evaluating the cranfield paradigm for conversational search systems. In *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '22, page 275–280, New York, NY, USA. Association for Computing Machinery.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Prashant Jayannavar, Anjali Narayan-Chen, and Julia Hockenmaier. 2020. Learning to execute instructions in a Minecraft dialogue. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2589–2602, Online. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

To Eun Kim and Aldo Lipani. 2022. A multi-task based neural model to simulate users in goal oriented dialogue systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pages 2115–2119, New York, NY, USA. Association for Computing Machinery.

Julia Kiseleva, Alexey Skrynnik, Artem Zholus, Shrestha Mohanty, Negar Arabzadeh, Marc-Alexandre Côté, Mohammad Aliannejadi, Milagro Teruel, Ziming Li, Mikhail Burtsev, et al. 2022. Iglu 2022: Interactive grounded language understanding in a collaborative environment at neurips 2022. *arXiv preprint arXiv:2205.13771*.

Anjali Narayan-Chen, Prashant Jayannavar, and Julia Hockenmaier. 2019. Collaborative dialogue in Minecraft. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5405–5415, Florence, Italy. Association for Computational Linguistics.

Maarten Sap, Ronan LeBras, Daniel Fried, and Yejin Choi. 2022. Neural theory-of-mind? on the limits of social intelligence in large lms.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Zhengxiang Shi, Yue Feng, and Aldo Lipani. 2022. Learning to execute actions or ask clarification questions. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2060–2070, Seattle, United States. Association for Computational Linguistics.

Zhengxiang Shi and Aldo Lipani. 2023. Don't stop pretraining? make prompt-based fine-tuning powerful learner. *arXiv preprint arXiv:2305.01711*.