Doc2Bot: Accessing Heterogeneous Documents via Conversational Bots

Haomin Fu[1,2,*] **Yeqin Zhang**[1,2,*] **Haiyang Yu**[2], **Jian Sun**[2], **Fei Huang**[2], **Luo Si**[2]
Yongbin Li[2†] and **Cam-Tu Nguyen**[1†]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, China
[2]Alibaba Group
{haominfu, zhangyeqin}@smail.nju.edu.cn
{yifei.yhy, jian.sun, f.huang, luo.si}@alibaba-inc.com
shuide.lyb@alibaba-inc.com, ncamtu@nju.edu.cn

Abstract

This paper introduces Doc2Bot, a novel dataset for building machines that help users seek information via conversations. This is of particular interest for companies and organizations that own a large number of manuals or instruction books. Despite its potential, the nature of our task poses several challenges: (1) documents contain various structures that hinder the ability of machines to comprehend, and (2) user information needs are often underspecified. Compared to prior datasets that either focus on a single structural type or overlook the role of questioning to uncover user needs, the Doc2Bot dataset is developed to target such challenges systematically. Our dataset contains over 100,000 turns based on Chinese documents from five domains, larger than any prior document-grounded dialog dataset for information seeking. We propose three tasks in Doc2Bot: (1) dialog state tracking to track user intentions, (2) dialog policy learning to plan system actions and contents, and (3) response generation which generates responses based on the outputs of the dialog policy. Baseline methods based on the latest deep learning models are presented, indicating that our proposed tasks are challenging and worthy of further research.

1 Introduction

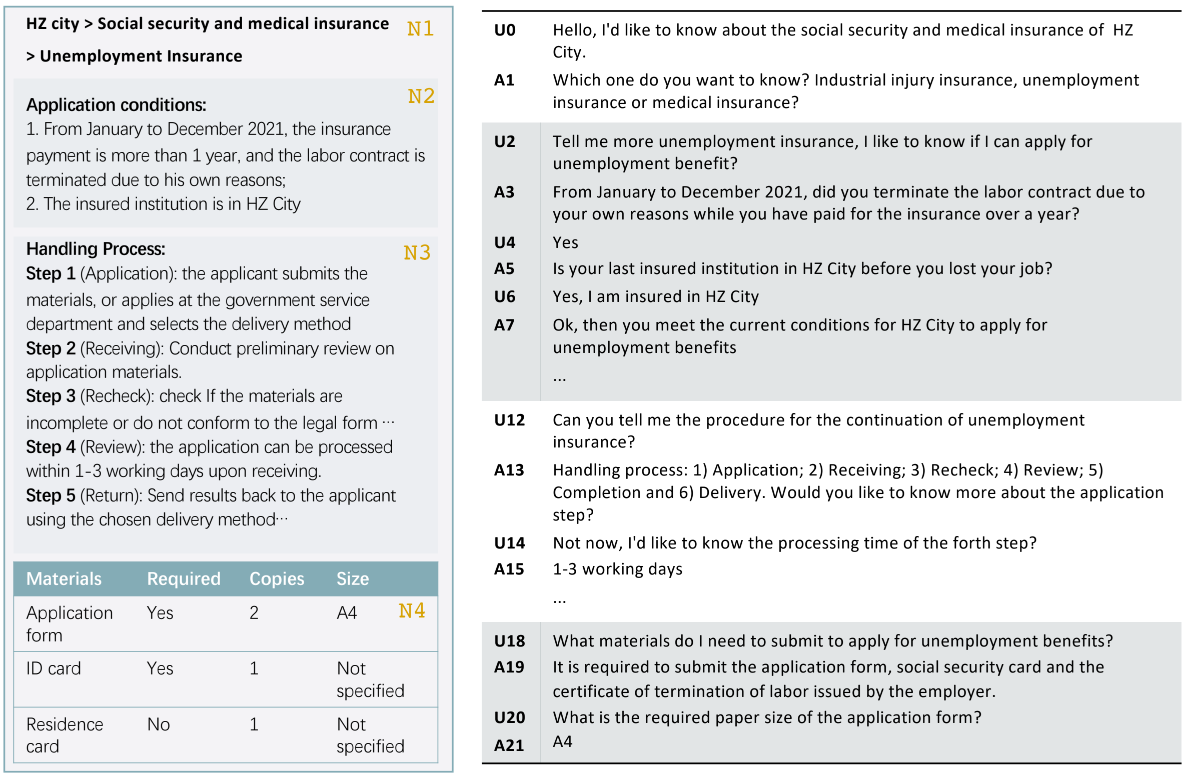
The last decade has witnessed a dramatic change in how humans interact with information retrieval systems. Although traditional search engines still play an important role in our daily life, the wide adoption of smart devices with small screens requires systems to answer user requests more concisely. Early attempts focus on answering independent questions (Rajpurkar et al., 2016), whereas recent studies pay attention to handling interconnected questions via conversations around a single passage (Pasupat and Liang, 2015; Chen et al., 2020) or documents (Feng et al., 2020, 2021). Yet, the nature of heterogeneous documents and our conversational setting pose challenges that require further attention. We, therefore, develop Doc2Bot[1] with these considerations in mind.

The first concerns the nature of heterogeneous documents, which often contain different types of structures such as tables and sequences. To answer questions regarding such structural types, systems need to acquire various skills. Figure 1 shows a conversation between a user and an agent, where the agent has access to a collection of documents. In this conversation, every utterance except the first one depends on the conversation history and the grounded documents. The document contains diverse structures including conditions (`N2`), sequences (`N3`), and tables (`N4`). Each type of structure poses different challenges for the agent. For example, the agent needs to understand the rule text in `N2` to ask questions such as `A3` and `A5`. The agent also needs to be able to summarize the procedure in `N3` to produce the answer (`A13`) or jump to a specific step of the procedure if being asked (`A15`). In addition, the agent needs to be capable of summarizing the tables (`A19`) or answering questions with row and column constraints (`A21`). Note that there have been previous attempts (Pasupat and Liang, 2015; Saeidi et al., 2018; Chen et al., 2020; Zhang et al., 2022b) to address different structures in isolation. However, little research has been done on handling multiple types of structures in the same document context.

The second concerns the nature of our conversational setting, which is to help users seek information from documents. Since user information needs are often under-specified, it is desirable for systems to ask clarifying questions. This resembles the task of relevance feedback and query refinement in traditional information retrieval. However, in Doc2Bot, system feedback is in the form of natural

[*]Equal contribution.
[†]Corresponding authors.

[1]https://github.com/Doc2Bot/Doc2Bot



HZ city > Social security and medical insurance > Unemployment Insurance N1

Application conditions: N2
1. From January to December 2021, the insurance payment is more than 1 year, and the labor contract is terminated due to his own reasons;
2. The insured institution is in HZ City

Handling Process: N3
Step 1 (Application): the applicant submits the materials, or applies at the government service department and selects the delivery method
Step 2 (Receiving): Conduct preliminary review on application materials.
Step 3 (Recheck): check If the materials are incomplete or do not conform to the legal form ···
Step 4 (Review): the application can be processed within 1-3 working days upon receiving.
Step 5 (Return): Send results back to the applicant using the chosen delivery method···

Materials	Required	Copies	Size
Application form	Yes	2	A4 N4
ID card	Yes	1	Not specified
Residence card	No	1	Not specified

U0	Hello, I'd like to know about the social security and medical insurance of HZ City.
A1	Which one do you want to know? Industrial injury insurance, unemployment insurance or medical insurance?
U2	Tell me more unemployment insurance, I like to know if I can apply for unemployment benefit?
A3	From January to December 2021, did you terminate the labor contract due to your own reasons while you have paid for the insurance over a year?
U4	Yes
A5	Is your last insured institution in HZ City before you lost your job?
U6	Yes, I am insured in HZ City
A7	Ok, then you meet the current conditions for HZ City to apply for unemployment benefits
	...
U12	Can you tell me the procedure for the continuation of unemployment insurance?
A13	Handling process: 1) Application; 2) Receiving; 3) Recheck; 4) Review; 5) Completion and 6) Delivery. Would you like to know more about the application step?
U14	Not now, I'd like to know the processing time of the forth step?
A15	1-3 working days
	...
U18	What materials do I need to submit to apply for unemployment benefits?
A19	It is required to submit the application form, social security card and the certificate of termination of labor issued by the employer.
U20	What is the required paper size of the application form?
A21	A4

Figure 1: An example dialog (right) grounded on a document (left) with heterogeneous structures. From the top, the dialog contains 4 segments S1−4 grounded on 4 corresponding document segments N1−4. Here U and A stand for user and agent, respectively.

questions, and thus more user-friendly. For example, in Figure 1, A1 is a kind of multiple-choice question that the agent asks to narrow down the search for the answer. In contrast, A3 and A5 are to verify user situations to answer questions regarding condition/solution structure. Although learning to construct questions from a single passage has been studied in Machine Reading Comprehension (Saeidi et al., 2018; Guo et al., 2021), such finer-grained questions are required only when the passage containing the answer has been found. For document-grounded dialog systems (DGDS), the agent needs to have the skills to narrow down the search (A1) as well as to ask finer questions such as A3 and A5.

Towards such goals, there are several challenges that we need to address. First, documents come in different formats, and thus the process of constructing our dataset is more difficult than those from single passages with homogeneous structures. The difference in formats also hinders the ability of machines to learn common patterns. Second, like human-human conversations, it is desirable to have samples of human-system conversations that are natural, and coherent while being diverse for the machine learning purpose. We target such challenges systematically and make the following contributions:

- We present a unified representation for heterogeneous structures, which not only facilitates our data collection process but also helps systems to learn patterns across documents.
- We propose an agenda-based dialog collection protocol that controls the diversity and coherence of dialogues by design. The protocol also encourages crowd-collaborators to introduce ambiguities to conversations.
- We introduce a new dataset Doc2Bot which is larger in scale compared to recent datasets for DGDS (Feng et al., 2020, 2021) while introducing new challenges such as a new language (Chinese), richer relations (e.g, sections, conditions, tables, sequences) and new tasks (e.g. dialog policy learning).
- We evaluate our proposed tasks with the latest machine learning methods. The experiments show that our tasks are still challenging, which suggests room for further research.

2 Related Works

Our work is most closely related to the document-grounded dialog systems (DGDS) in the literature. Based on the conversation objective, we can roughly categorize the related tasks into chitchat, comprehension, or information seeking.

Document-grounded chitchat datasets such as WoW (Dinan et al., 2019), Holl-E (Moghe et al., 2018), CMU-DoG (Zhou et al., 2018) aim to enhance early chitchat systems by using information from grounded textual passages for answer generation. The goal is similar to an open chitchat system as the dialog agent tries to keep users engaged in long, informative, and interactive conversations. This is different from our setting because users of our system often have clear goals (information needs), and the dialog agent needs to provide users with accurate information as soon as possible.

For document-grounded "comprehension" such as CoQA (Reddy et al., 2019), Abg-CoQA (Guo et al., 2021) and ShARC (Saeidi et al., 2018), the agent is given a textual paragraph and needs to answer users' questions about the paragraph. This setting is similar to Machine Reading Comprehension (MRC). However, the difference is that questions in MRC may not form a coherent dialog. Noticeably, several question strategies have been targeted in Abg-CoQA and ShARC. For example, in Abg-CoQA, systems can ask clarifying questions to resolve different types of ambiguities. In ShARC, the authors created conversations where the system can learn to ask "yes/no" questions to understand users' information and provide appropriate answers. The questioning strategy in ShARC is designed based on text rules that define the relationship between "conditions" and "solutions" exhibited in the given paragraph. Although we also address question strategies, our tasks are more challenging because we focus on multiple documents.

The third type of DGDS (Penha et al., 2019; Feng et al., 2020, 2021) is closest to our setting where the agent needs to provide answers to information seekers in the shortest possible time. Mantis (Penha et al., 2019) was collected from online forums, and the grounded documents are not given in advance. As a result, Mantis does not come with a detailed annotation which is needed to study the capability of the agents to understand documents. In contrast, given a set of documents, Doc2dial (Feng et al., 2020) and Multidoc2dial (Feng et al., 2021) were collected in 2 stages: 1) dialog flows are first generated by labeling and linking paragraphs, 2) crowdsourcers then write conversations based on the suggested flows. Note that Multidoc2dial was built by rearranging dialogues from doc2dial so that one conversation can contain information from multiple documents. Although we follow similar steps for constructing the dataset, our dialog flow generation is essentially different, which addresses the coherence of the generated dialogues, and the multi-document grounding issue by design. In addition, our dataset exceeds Doc2dial and Multidoc2dial in scale, while also highlighting new challenges such as under-specified user requests.

3 Dataset Collection

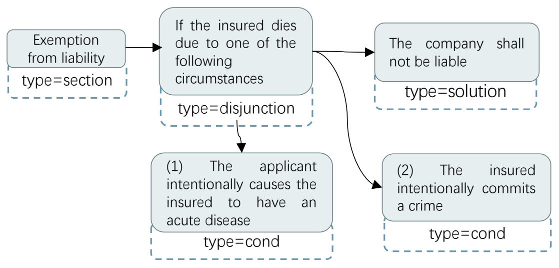
This section details the process of collecting Doc2Bot, including 4 stages: 1) **document collection** which selects targeted domains and documents; 2) **document graph construction** which unifies heterogeneous structures from multiple domains to build document graphs; 3) **dialog flow generation** that simulates the agenda of a user seeking information from a document graph; and 4) **dialog collection** where crowd-collaborators write dialogs based on the generated dialog flows.

3.1 Document Collection

For document collection, we examine several potential domains and select 5 representative ones including public services, technology, insurance, health care services, and wikiHow. For each domain, documents are selected based on two criteria: 1) the documents should be rich in structural types; 2) each document should have links to other documents so that we can test the ability of machines to reason over multiple documents. We design a simple ranking score based on these criteria and select the top-ranked documents for each domain.

3.2 Document Graph Construction

Documents from different domains or sources have vastly different formats (HTML, PDF, etc). Towards building scalable dialog systems across domains, it is important to have a unified format for encoding heterogeneous semantic structures in documents. Bear in mind that our target is to preserve those structures in the document context. This is unlike knowledge graphs and event graphs (Fu et al., 2020; Ma et al., 2021; Hogan et al., 2021) in which only entities or events are extracted while other context information is discarded.



Exemption from liability
type=section
If the insured dies due to one of the following circumstances
type=disjunction
The company shall not be liable
type=solution
(1) The applicant intentionally causes the insured to have an acute disease
type=cond
(2) The insured intentionally commits a crime
type=cond

Figure 2: The structure of a disjunction of conditions and the associated solution in the insurance domain.

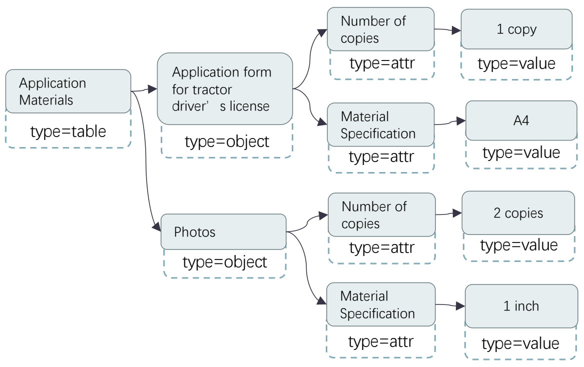


Figure 3: The structure of a table and its objects in the domain of public services.

Document Graph is defined as a directed graph where a node corresponds to a span of text in the document. Inspired by property graphs (Hogan et al., 2021), we associate each node with a node type and a set of additional property-value pairs. Each domain has a root node that connects to domain documents via title hierarchy.

A number of node types are defined to cover common discourse relations exhibited in multiple domains (Das et al., 2018; Stede et al., 2019). These include **section** type to denote section titles in documents. The types of **disjunction**, **conjunction**, **condition**, **solution**, **negation** are used to describe the condition-solution relation as depicted in Figure 2. The types of **table**, **object**, **attribute**, **value** are to encode the relations in tables as shown in Figure 3. The types of **sequence**, **sequence-step** are introduced to indicate the relations of texts in describing procedures such as `N3` in Figure 1. Last but not least, the **see-more** type is used to encode hyperlinks, and the **ordinary** type is assigned to the nodes belonging to none of the above.

The property-value pairs associated with nodes are used for additional information. For example, each node can be identified with `docid` and `nodeid`. Likewise, **see-more** nodes have properties such as *linked nodeid*. Additionally, we introduce *is-super-leaf* to indicate whether a node should be targeted in the dialog flow generation.

3.3 Dialog Flow Generation

Studies of human behaviors in goal-oriented dialog systems have long recognized the fact that users have hidden agendas (Schatzmann and Young, 2009) which direct the interactions between users and chatbots. This is also the idea behind the construction of well-known datasets such as MultiWoz (Budzianowski et al., 2018). Although the connection between DGDS in information-seeking scenarios and goal-oriented dialog systems has been suggested (Feng et al., 2020, 2021), DGDS have no explicit schemes, thus hindering the agenda-based approach to dialog collection. As an alternative, we exploit the graph structure of the document graph to build up agendas for simulating dialog flows between a user and an agent. Here, a dialog flow is defined as a sequence of goals, each goal corresponds to a node in our document graph. We mark nodes, that can be used as goals, with *is_super_leaf* being true using a semi-automatic method.

Our agenda-based procedure for generating a dialog flow is demonstrated in Algorithm 1. Here, the procedure takes as inputs the document graph G, the transition probabilities ξ, the maximum number of goals $nGoal$, and the initial selected document d. The objective is to generate diverse dialog flows based on which crowd contributors can write conversations. For each goal, a prompt can be generated to suggest questions that can be asked about the subtree rooted at the goal node (line 6). For example, given a table in Figure 3 as a goal, we can generate the corresponding prompt by: (1) randomly selecting some "objects" and "attributes" as constraints, e.g. *paper size* and *application form*; (2) using templates to convert the constraints to a guideline such as *"write a number of question-answer turns so that the system final answer is A4 - the **paper size** of the **application form**"*.

We use an agenda stack to contain a list of potential goals that a user might switch to (from the last goal). The candidates nearer to the top of the agenda stack are closer to the last goal in the document graph. The action of a user switching from one goal to another is simulated by three factors, the follow-up rate ξ_{fl}, the in-jump rate ξ_{inj} and the out-jump rate ξ_{outj}. When the action is *follow-up*, users tend to ask about the related information of

Algorithm 1 Agenda-based dialog flow generation

```
Require: $G$; $\xi = [\xi_{fl}, \xi_{inj}, \xi_{outj}]$, $nGoals$, $d$
Ensure: a dialog flow $flow$, that is a list of $goal$,
    each corresponds to a node in $G$
 1: $goal \leftarrow sample\_leaves(G, d.root)$
 2: $path \leftarrow get\_path(G, d.root, goal)$
 3: Push nodes in $path$ to the agenda stack $A$
 4: while $len(flow) < nGoals$ do
 5:     $goal \leftarrow$ pop a leaf from A        ▷ last goal
 6:     $prompt \leftarrow gen\_prompt(G, goal)$
 7:     $flow \leftarrow$ append $([goal, prompt])$
 8:     Sample $act$ based on $\xi$
 9:     if $act$ is $follow\_up$ then
10:         $st \leftarrow$ pop the top from $A$
11:     if $act$ is $in\_jump$ then
12:         $st \leftarrow$ sample a random node in$A$
13:         pop A till seeing $st$
14:     if $act$ is $out\_jump$ then
15:         $d \leftarrow sample\_connected\_doc(G, d)$
16:         $st \leftarrow d.root$
17:     $goal \leftarrow sample\_leaves(G, st.root)$
18:     $path \leftarrow get\_path(G, st.root, goal)$
19:     Push nodes in $path$ to the agenda stack $A$
```

the recent goal (line 10). If the action is *in-jump*, users ask about some goals further away from the last goal but still close to some goals in the past. The out-jump action, on the other hand, allows us to simulate the situation where users may ask about related documents. The out-jump rate might be increased if the current goal is linked to an outside document via a **see-more** node. Note that the procedure that samples leaf nodes (line 1, 17) should exclude the visited nodes.

Our agenda-based flow generation is adaptable to include new types of structures. This is because whenever we need to target a new structural type, we just need to adjust the document graph definition, and design a new prompt generation while keeping Algorithm 1 unchanged.

3.4 Dialog Collection

We ask crowd contributors to write conversations based on our generated dialog flows. We follow the protocol that one writer plays both agent and user roles and completes the whole dialog like (Feng et al., 2020, 2021). To further improve the coherence of the generated dialogs, we ask writers to examine each dialog flow and skip some goals if the goal is not consistent with the rest of the flow.

Once a dialog flow has been double-checked by a writer, he/she is requested to write dialog utterances based on the goals and their associated prompts (see Section 3.3). Each goal, its prompt, and its context are then presented one by one to a crowd contributor. By context, we mean the path from the graph root node to the goal node and its neighbors. The writer then interchangeably takes the role of a user or an agent with different interfaces (see Appendix A). When it is the user's turn, we encourage the writer to pose an under-specified question, which might make the system confused between the goal node and others in the context. When it is the system's turn, the writer is either asked to provide an answer based on the goal node or ask questions to clarify. Once the system has fulfilled the goal task, the writer should terminate the goal to move to the next one in the flow. Besides utterances, for each turn, the writer needs to provide annotations such as user/system acts, and grounding texts/nodes.

To ensure the quality of the dataset, crowd collaborators were selected and trained for two weeks. After the training period, we sampled several dialogs and provided feedback to writers in a weekly manner. Our task was completed in 3 months and we paid 0.836 RMB per dialog turn.

4 Data Analysis

Document Data Table 1 lists the number of documents, along with the number of structures by types in Doc2Bot. As we can see, documents in the domain of public services are very rich in structural types, whereas wikiHow contains a lot of sequences showing how-to procedures. Although the size of the document collection is still moderate, it is more than 3 times larger than the document collection size in doc2dial and multidoc2dial (Feng et al., 2020, 2021), the previous datasets for DGDS.

Dialog Data Doc2Bot contains 6,619 dialogues with 101,994 turns (see Table 2). The mean length of user and system utterances are 18.3 and 49.99 words, respectively. Each user utterance is annotated with a dialog state, consisting of a user action (Figure 4) and some grounding texts (1.39 texts on average). Similarly, each system utterance is annotated with a system action (Figure 5) and an average number of 1.81 grounding nodes (in the document graph). As we can see from Figure 5, system questions correspond to about 20% of the total number of system utterances. This implies

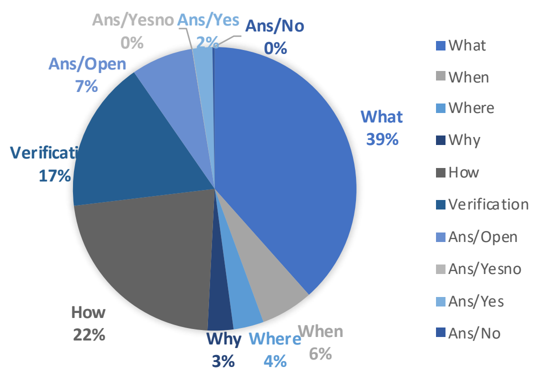


Figure 4: The distribution of user dialog actions.

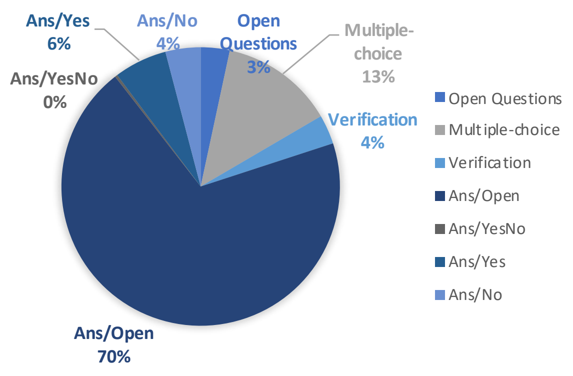


Figure 5: The distribution of system dialog actions.

Domain	#Docs	#Tab	#Seq	#Con	#Sec
Public	390	2747	1909	644	3993
Health	403	0	2	22	2468
Insurance	100	0	0	2423	6107
Wikihow	385	0	658	0	1900
Tech.	301	47	771	250	1526
All	**1579**	**2794**	**3340**	**3339**	**15994**

Table 1: Statistics on the number of documents and the structural types by domain. Here, "Public" is short for "Public Services". Tab, Seq, Cond, and Sec stand for tables, sequences, conditions/solutions, and sections.

Conversation and Goal Statistics			
#dialogs	6619	#turns	101994
#dial (>1 doc)	2207	#doc/dial	1.42
#gr/usr-turn	1.39	#gr/sys-turn	1.81
usr-turn len	18.3	sys-turn len	49.99
#ordinary	20747	#tables	607
#sequences	5707	#conditions	896

Table 2: Conversation statistics: the # of dialogues, turns, the average # of grounding (#gr) texts per turn, grounding documents per dialog. Goal statistics: the # of dialog segments about tables, conditions, etc.

that our agenda-based protocol has successfully encouraged crowd-collaborators to include a considerable number of ambiguities.

Table 2 shows the number of goals for different structural types. Since each goal corresponds to a dialog segment, it can be inferred that although the majority of requests are about plain texts (ordinary), Doc2Bot does contain a large number of scenarios grounded on other types of structures.

For evaluation, we define several tasks (see Section 5) and split the dialog dataset into a training set, consisting of 70% of dialogues, as well as a validation set of 10% and a test set of 20%.

5 Tasks and Baselines

Inspired by goal-oriented dialog systems, this paper considers three tasks: (1) dialog state tracking; (2) dialog policy learning; (3) response generation.

5.1 Dialog State Tracking

Dialog State Tracking (DST) aims at tracking user intentions and key information (Dai et al., 2021; Sun et al., 2021; Zhang et al., 2022a). Specifically, given a dialog history H and the latest user turn, we need to perform (1) user action prediction where the actions are given in Figure 4, and (2) grounding text matching where the candidate texts are extracted from nodes of the document graph without duplication.

Baseline Approach The problem of user act prediction can be formalized as a multi-class classification, where we use RoBERTa (Liu et al., 2019), BERT (Devlin et al., 2019) and ELECTRA (Clark et al., 2020) as our baselines. The problem of grounding text matching can also be formalized as a classification problem, where we classify a text as relevant vs irrelevant given the dialog history. Since the number of texts is too large for evaluation, we follow the retrieval then classification approach, where a retrieval model is first used to retrieve candidate texts, and the above models are used to classify if a candidate is relevant or not.

All in all, BM25 (Robertson et al., 2009) and Dense Passage Retrieval (Karpukhin et al., 2020) are used to select a number of candidate texts. And for each of the above classification baselines, we use two independent models for the user action and grounding text prediction.

	Act		**GN**	Joint
	ma-F1	F1	F1	Acc
DPR+RT	**77.45**	89.38	**57.26**	**34.55**
BM25+RT	-	-	38.62	15.39
DPR+BT	76.95	**89.52**	57.15	32.22
BM25+BT	-	-	42.32	19.90
DPR+ET	76.06	87.88	57.02	30.29
BM25+ET	-	-	39.24	16.73

Table 3: Results on user act prediction (Act) and grounding text prediction (GN) using BM25 and DPR for retrieval. Here, RT is for RoBERTa, BT is for BERT, and ET is for ELECTRA.

Evaluation Metrics For user action prediction, we report micro-F1 (F1) and macro-F1 (ma-F1) which are the mean F1 scores averaged over turns and actions, respectively. For grounding text prediction, micro-F1 is applied to measure the performance of our baselines. In addition, joint accuracy is used to measure the percentage of turns, for which the user action and all the grounding texts are correctly identified.

Experimental Results Table 3 shows that while we can achieve promising results with user action prediction, the task of grounding text matching is still very challenging. For the matching task, 200 candidates are retrieved by BM25 or DPR, and then reranked by the classification models (RoBERTa, BERT, or ELECTRA). It is observable that models based on DPR (e.g., DPR+RT) outperform BM25-based models (e.g. BM25+RT), indicating DPR is a better retrieval method compared to BM25. The best F1 score and joint accuracy, however, are only 57.26% and 34.55%, respectively. The low value of F1 on grounding text prediction suggests room for improving both the retrieval and the classification models. The lower value of joint accuracy shows that it is even more challenging to correctly identify all the grounding texts.

5.2 Dialog Policy Learning

This task aims at planning the system act and the contents to generate the next response (He et al., 2022). The input of this task includes (1) the dialog history H, (2) the document graph G, and (3) the dialog state DS; whereas the expected output consists of the system action (Figure 5) and the grounding nodes set.

Unlike grounding text matching in DST where we find texts related to user requests, grounding node prediction requires the agent to locate nodes that should be used for system response. For example, when asking about the table in Figure 3, users may provide the table name and the attribute names, which are found in the DST task. The dialog policy then infers the "value" nodes that contain the answer given the dialog states. By assuming that the dialog states are available, we hypothesize that the agent can fully understand user utterances, thus having a perfect DST module. In practice, this is still a difficult task as seen in the previous section.

To further simplify the task of grounding node prediction, we formalize it as a classification problem where the agent just needs to predict whether a candidate node should be used or not. Here, a candidate set is selected for each turn by combining the gold system grounding nodes, and 30 distractors, which are chosen randomly from the set constructed by: 1) selecting nodes with texts that match the most recent dialog states; 2) selecting the neighbors (parents, siblings) of the nodes found in (1) as well as those of the gold grounding nodes.

Baseline Approach For the system act prediction, we also use RoBERTa, BERT and ELECTRA as the classification models where the input is the dialog history, and the output is one of the system act (Figure 5). For grounding node prediction, we first form the input by concatenating (1) history: the two latest utterances in the dialog history; (2) the dialog state; (3) the structure information obtained by sequentializing the path leading to the candidate node in the document graph (Wan et al., 2021); and (4) the candidate node. Before each segment (1-4) of the input, we add a special prompt to indicate its semantics. We then use these classification models to predict the relevance of the candidate node.

For the ablation study, we consider two variants for each of these baselines for the grounding node prediction. The first one treats a document as a sequence of texts and replaces the structure information with the context window of the node in the original document, we refer to this as (- structure). The second variant excludes the dialog state information, which is referred to as (- dialog state).

Experimental Results The same metrics, which are used to evaluate dialog state tracking, are used here for evaluating dialog policy models. The ex-

Model	Act		GN	Joint
	ma-F1	F1	F1	Acc
RoBERTa	**46.66**	**82.87**	**81.67**	**43.13**
-structure	-	-	77.18	35.51
-states	-	-	73.71	32.49
BERT	45.00	82.58	81.40	41.75
-structure	-	-	75.61	32.08
-states	-	-	73.14	31.21
ELECTRA	42.77	81.91	81.31	40.31
-structure	-	-	75.66	32.18
-states	-	-	73.17	30.53

Table 4: Results on system action prediction (Act) and grounding node prediction (GN).

perimental results are shown in Table 4, from which several observations can be drawn. Firstly, the best baseline can only achieve macro-F1 of 46.66% on system action prediction, showing that this task is more difficult than user action prediction. The main reason for the difficulty of this task is the imbalance in the action distribution (see Figure 5), an issue that requires further attention. Secondly, both document structures and dialog states are essential for grounding node prediction, since the performance drops significantly without either one of them. And finally, despite having the full information of dialog states and a simplified formalization with only 30 most potential distractors, the best performance we can obtain is only 43.13% joint accuracy.

5.3 Response Generation

This task aims at generating a natural language response based on the given system act and grounding nodes set. The response can be a clarifying question or an answer. The input includes (1) dialog history H, (2) system act a_s, and (3) grounding nodes set $\mathbb{N}_g$. The target output is a system response r consistent with the chosen action, the planned contents, and the history.

Baseline Approach We use three encoder-decoder generative models, Pegasus (Zhang et al., 2020), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020), as baselines. Here, the encoder takes the concatenated sequence of all information as inputs, and the decoder generates the response.

To study the impact of the system action prediction task, we consider the variants of three baselines where the system actions are not included as input.

Model	BLEU-1	BLEU-2	BLEU-4
Pegasus	59.85	53.48	44.56
Pegasus$_{\texttt{w/o act}}$	58.12	52.09	43.59
BART	60.35	54.00	45.03
BART$_{\texttt{w/o act}}$	58.75	52.62	44.06
T5	**60.97**	**54.41**	**45.20**
T5$_{\texttt{w/o act}}$	59.30	52.98	44.19

Table 5: Response generation results for Pegasus, BART and T5. `w/o act` means the system actions are not included as input.

Experimental Results We use BLEU[2] (Papineni et al., 2002), a commonly used metric to evaluate the performance of the response generation. The experimental results are given in Table 5, where two main observations can be found. Firstly, T5 is slightly better than other baselines for response generation. One possible explanation is that T5 has more parameters (see Table 6) than BART, and the pre-training task of Pegasus is more suitable for the summarization task. It is worth mentioning that the results here are obtained by using gold values of system actions and grounding nodes. In practice, we need to take into account the errors accumulated by the DST model and the dialog policy model. Given that the best joint accuracy of DST and dialog policy are only 34.55% and 43.13%, one can see that the performance of response generation is still far from this upper bound. Secondly, while the grounding nodes are undeniably important for response generation, the introduction of system actions has not always been helpful in the previous dataset (Feng et al., 2020). However, in Doc2Bot, we find that having information on system actions can help improve the performance of response generation for all baselines. This partially validates our design choices of system actions in Doc2Bot.

6 Conclusion

This paper presented Doc2Bot, a novel dataset for DGDS for information seeking. Unlike prior datasets, Doc2Bot contains examples that simultaneously test the ability of machines to comprehend heterogeneous documents and clarify user information needs. We proposed three main tasks associated with Doc2Bot: (1) dialog state tracking, which tracks user intentions during the conversation, (2)

[2]https://github.com/nltk/nltk

dialog policy learning, which plans the next system action and contents, and (3) response generation, which is to generate system responses based on the outputs of the dialog policy. We then presented baselines for our tasks using several contemporary models. Our experimental results showed that: 1) Both dialog states information and document structure information are important for the task of dialog policy learning; 2) Planning system actions helps improve response generation; and 3) The tasks of dialog state tracking and dialog policy learning, which are essential for response generation, are still very challenging with joint accuracy of only 34.55% and 43.13% respectively. We hope that our dataset and such observations will be helpful for future research in this direction.

Limitations

Towards a practical document-grounded dialog system, some problems have not been addressed in this work. (1) The task of automatic construction of document graphs deserves further attention. Although there exist automatic solutions for parsing discourse relations, the results are still far from desirable for Chinese. As a result, manual post-processing was required for the construction of document graphs, which can be daunting, particularly for domains with many documents. (2) Doc2Bot has yet to include unanswerable cases. Although such samples can be created with a data recomposition step, due to time limitations, we have to leave this consideration for the future versions of Doc2Bot.

Ethics Statement

Several ethical issues need our attention. Firstly, more research should be done to make sure the robustness and effectiveness of document-grounded dialog systems. Without careful consideration, such systems will inconvenience both users and the organizations that own the dialog systems. Secondly, although we can control the diversity of our dataset by adjusting the sampling ratios, the conversations might still contain some level of biases, for which more careful examination should be done. Third, our dataset should be used only for research purposes. For Health domain, the responses must not be taken for diagnosis. Finally, although our collected dialogs contain no privacy sensitive data, a part of the documents has usage constraints, and thus we can only publish part of our dataset. The full dataset can be shared upon usage agreements, and for research purposes only.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. We also like to thank Dr. Song Feng and Dr. Bowen Yu for their helpful suggestions. This work was supported by Alibaba Innovative Research project "Document Grounded Dialogue System".

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations*.

Yinpei Dai, Hangyu Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, and Xiaodan Zhu. 2021. Preview, attend and review: Schema-aware curriculum learning for multi-domain dialogue state tracking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 879–885.

Debopam Das, Tatjana Scheffler, Peter Bourgonje, and Manfred Stede. 2018. Constructing a lexicon of english discourse connectives. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 360–365.

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of Wikipedia: Knowledge-powered conversational agents. In *Proceedings of the International Conference on Learning Representations*.

Song Feng, Siva Sankalp Patel, Hui Wan, and Sachindra Joshi. 2021. MultiDoc2Dial: Modeling dialogues grounded in multiple documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 6162–6176.

Song Feng, Hui Wan, Chulaka Gunasekara, Siva Patel, Sachindra Joshi, and Luis Lastras. 2020. doc2dial: A goal-oriented document-grounded dialogue dataset. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 8118–8128.

Bin Fu, Yunqi Qiu, Chengguang Tang, Yang Li, Haiyang Yu, and Jian Sun. 2020. A survey on complex question answering over knowledge base: Recent advances and challenges. *arXiv preprint arXiv:2007.13069*.

Meiqi Guo, Mingda Zhang, Siva Reddy, and Malihe Alikhani. 2021. Abg-coqa: Clarifying ambiguity in conversational question answering. In *Proceedings of the 3rd Conference on Automated Knowledge Base Construction*.

Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, et al. 2022. Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10749–10757.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. *Synthesis Lectures on Data, Semantics, and Knowledge*, 12(2):1–257.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*.

Longxuan Ma, Mingda Li, Wei-Nan Zhang, Jiapeng Li, and Ting Liu. 2021. Unstructured text enhanced open-domain dialogue system: A systematic survey. *ACM Transactions on Information Systems*, 40(1):1–44.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M. Khapra. 2018. Towards exploiting background knowledge for building conversation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2322–2332.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1470–1480.

Gustavo Penha, Alexandru Balan, and Claudia Hauff. 2019. Introducing mantis: a novel multi-domain information seeking dialogues dataset. *arXiv preprint arXiv:1912.04639*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:140:1–140:67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Marzieh Saeidi, Max Bartolo, Patrick S. H. Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2087–2097.

Jost Schatzmann and Steve Young. 2009. The hidden agenda user simulation model. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):733–747.

Manfred Stede, Tatjana Scheffler, and Amália Mendes. 2019. Connective-lex: A web-based multilingual lexical resource for connectives. *Discours. Revue de linguistique, psycholinguistique et informatique. A journal of linguistics, psycholinguistics and computational linguistics*.

Yajing Sun, Yong Shan, Chengguang Tang, Yue Hu, Yinpei Dai, Jing Yu, Jian Sun, Fei Huang, and Luo Si. 2021. Unsupervised learning of deterministic dialogue structure with edge-enhanced graph autoencoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13869–13877.

Hui Wan, Song Feng, Chulaka Gunasekara, Siva Sankalp Patel, Sachindra Joshi, and Luis Lastras. 2021. Does structure matter? encoding documents for machine reading comprehension. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4626–4634.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339.

Sai Zhang, Yuwei Hu, Yuchuan Wu, Jiaman Wu, Yongbin Li, Jian Sun, Caixia Yuan, and Xiaojie Wang. 2022a. A slot is not built in one utterance: Spoken language dialogs with sub-slots. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 309–321.

Zhenyu Zhang, Bowen Yu, Haiyang Yu, Tingwen Liu, Cheng Fu, Jingyang Li, Chengguang Tang, Jian Sun, and Yongbin Li. 2022b. Layout-aware information extraction for document-grounded dialogue: Dataset, method and demonstration. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 7252–7260.

Kangyan Zhou, Shrimai Prabhumoye, and Alan W Black. 2018. A dataset for document grounded conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 708–713.

A Dialog Collection Process

Each dialog flow is presented to crowd-collaborators with prompts that suggest questions related to selected goals in the dialog flow. We design different prompts associated with different types of nodes:

- Asking about Tables :We assume that all tables contain a four-level structure *table-object-attribute-value*. We consider each table as a list of key objects and some attributes with them. Inspired by (Pasupat and Liang, 2015; Chen et al., 2020), we design three types of prompt for tables: 1. asking for the general information of a table, such as *“what material do I need to offer?”* 2. asking for the general information of an object in the table, such as *“can you tell me more about the first material?”* 3. asking for a value of the object attribute, such as *“How many copies of the first material do I need to bring?”*.

- Asking about Sequence: Similar to tables, we design three patterns for asking about the sequence: 1. asking for the general information of a sequence, such as *“what should I do for the application?”* 2. asking for the general information of a step in the sequence, such as *“can you tell me more about step one?”* 3. asking for specific information for one step, such as *“how long does step one take?”*

- Asking about Condition/Solution: Depending on the final system answer, we design three QA patterns for asking about conditions. We will randomly select a pattern from YES/NO/-CONDITIONAL/SOLUTION as a prompt. For the first two patterns, the user need to ask a verifying question like *“can I apply for this fund?”*, and the final answer for system must be YES or NO after checking some conditions. For CONDITIONAL, the user needs to ask a question to know the conditions he needs to meet, such as *“I want to apply this fund, what do I need?”*. For SOLUTION, the user will explicitly say some conditions, and the final system answer should be the solution for that. For example, the user may ask *“I am a 35 year old worker, which insurance can I apply for?”*.

- Asking about Ordinary: Ordinary nodes correspond to unstructured texts where we would like to include samples similar to the task of machine reading comprehension. As a result, we ask crowd collaborators to make up questions based on the node text, so that a span in the node text can be used to answer the question.

The writer is asked to think about a consistent information-seeking situation based on the flow and has the option to skip some goals in the dialog flow. The writer then interchangeably takes the role of a user or an agent with different interfaces. When it is the user's turn (see Figure 6), we encourage the writer to pose an under-specified question, which might make the system confused between the goal node and others in the context. When it is the system turn (see Figure 7), the writer is either asked to provide an answer based on the goal node or ask questions to clarify. For example, a writer can exploit the section nodes to write questions so that the next user's answer can help the agent to better target the goal node in the document graph. Likewise, a writer can select condition nodes (see Figure 2) to write questions so that the user answer can help to answer the user question about the condition/solution structure. Once the system has fulfilled the goal task, the writer should terminate the goal to move to the next one in the flow. Besides utterances, for each (user or system turn) the writer needs to provide annotations such as (user/system) acts, node texts that entail user utterances, and grounding nodes for system questions/answers.

B Experiments

The implementation is in PyTorch and the pre-trained models we used are from HuggingFace Transformers[3]. The information of them is shown in the Table 6. For dialog act prediction, we use an MLP to map the pooling output of the pre-trained models to the action space. For all experiments, we evaluate the model at the end of each epoch and select the best-performed checkpoint. But for response generation baselines we only evaluate the last 5 epochs, since the inference stage is time-consuming. AdamW is used to optimize the parameters, with 1e-08 epsilon and 0.01 weight decay. All experiments are performed on one Tesla V100 with 32GB memory. The average runtime of training and inference for each baseline and best validation performance are shown in Table 7.

[3]https://huggingface.co

Hyperparameters for DST baselines For user act prediction, we fine-tune the baselines for 5 epochs with a batch size of 8. We use a learning rate of 5e-6, linear scheduling without warmup. For the DPR, we implement it by ourselves, using a Chinese version of pre-trained RoBERTa as the backbone network. We train the model for 50 epochs with a batch size of 12. The learning rate is 3e-6, linear scheduling with warmup steps of 500. We use one BM25 negative passage per query in addition to in-batch negatives. And the FAISS[4] is used to speed up the vector search. For the grounding text prediction baselines, we follow the SC[5] example from HuggingFace (Wolf et al., 2020). We use the top-20 retrieved samples, except for the gold ones, as negative samples for training. We use a learning rate of 2e-5, linear scheduling with warmup steps of 500, for 5 epochs.

B.1 Hyperparameters for DLP baselines

For system act prediction, we use the same hyperparameters as the user act prediction in Section B. For baselines and variants in the candidate nodes classification task, we train them for 5 epochs with a batch size of 20. The learning rate is 1e-5 for ELECTRA and 2e-5 for the others, with linear scheduling of 500 warmup steps.

B.2 Hyperparameters for RG baselines

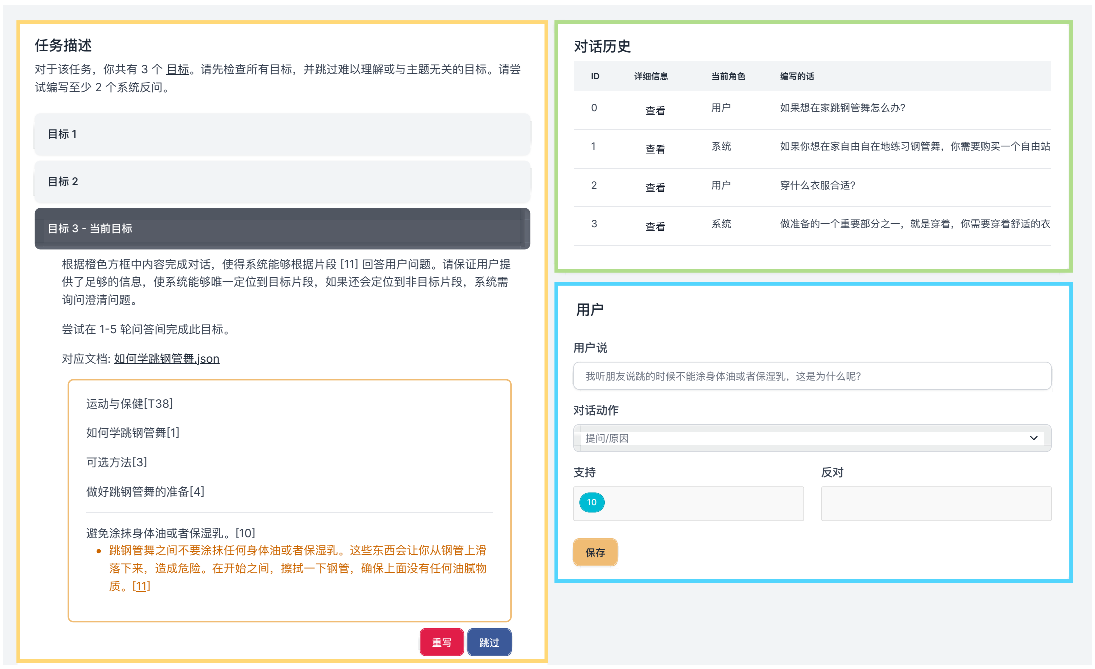
We finetune the pre-trained ELECTRA and BART for 20 epochs with a batch size of 12. And the learning rate is 2e-5, also linear scheduling of 500 warmup steps. For the T5 model, we train it for 20 epochs with a batch size of 10. The learning rate is 3e-4 and we use the same linear scheduling as the others. For inference, we set the beam search size as 4, the max generate length as 512, and the length penalty as 1.0. The variants of them use the same hyperparameters since the only difference is the input sequence.

C Challenges

There are different types of challenges exhibited in Doc2Bot, which can be divided into three categories: 1) understating documents; 2) understanding dialog context; and 3) abstract reponse generation.

[4]https://github.com/facebookresearch/faiss

[5]https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertForSequenceClassification


任务描述

对于该任务，你共有 3 个 目标。请先检查所有目标，并跳过难以理解或与主题无关的目标。请尝试编写至少 2 个系统反问。

目标 1

目标 2

目标 3 - 当前目标

根据橙色方框中内容完成对话，使得系统能够根据片段 [11] 回答用户问题。请保证用户提供了足够的信息，使系统能够唯一定位到目标片段，如果还会定位到非目标片段，系统需询问澄清问题。

尝试在 1-5 轮问答间完成此目标。

对应文档: 如何学跳钢管舞.json

运动与保健[T38]

如何学跳钢管舞[1]

可选方法[3]

做好跳钢管舞的准备[4]

避免涂抹身体油或者保湿乳。[10]
- 跳钢管舞之间不要涂抹任何身体油或者保湿乳。这些东西会让你从钢管上滑落下来，造成危险。在开始之间，擦拭一下钢管，确保上面没有任何油腻物质。[11]

对话历史

ID	详细信息	当前角色	编写的话
0	查看	用户	如果想在家跳钢管舞怎么办?
1	查看	系统	如果你想在家自由自在地练习钢管舞，你需要购买一个自由站
2	查看	用户	穿什么衣服合适?
3	查看	系统	做准备的一个重要部分之一，就是穿着，你需要穿着舒适的衣

用户

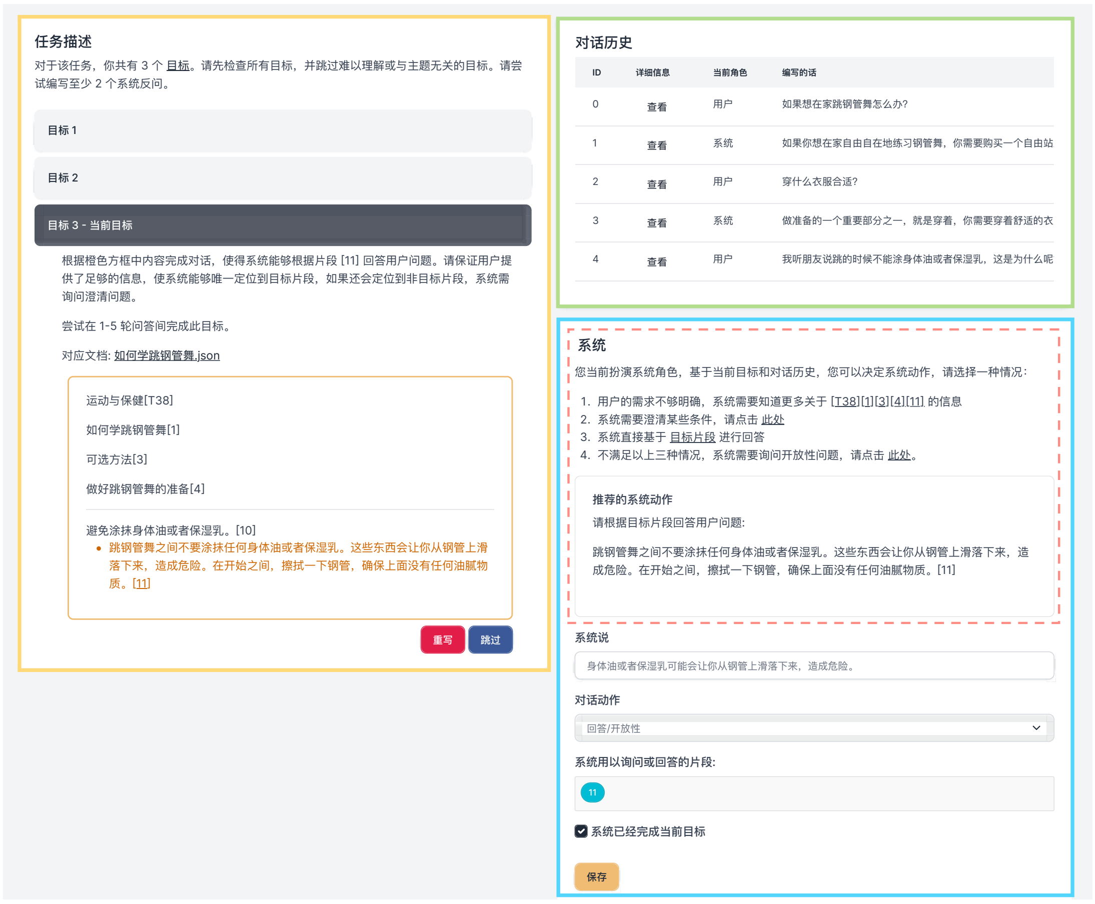
用户说: 我听朋友说跳的时候不能涂身体油或者保湿乳，这是为什么呢?

对话动作: 提问/原因

支持: 10

反对:

Figure 6: Annotation interface to write a user sentence. The dialog flow we generated is shown in the yellow box, and the dialog history is shown in the green box. The crowdsourcers are asked to write a sentence, select dialog act and grounding nodes in the blue box, to lead the dialog to the orange target text of the current goal.


任务描述

对于该任务，你共有 3 个 目标。请先检查所有目标，并跳过难以理解或与主题无关的目标。请尝试编写至少 2 个系统反问。

目标 1

目标 2

目标 3 - 当前目标

根据橙色方框中内容完成对话，使得系统能够根据片段 [11] 回答用户问题。请保证用户提供了足够的信息，使系统能够唯一定位到目标片段，如果还会定位到非目标片段，系统需询问澄清问题。

尝试在 1-5 轮问答间完成此目标。

对应文档: 如何学跳钢管舞.json

运动与保健[T38]

如何学跳钢管舞[1]

可选方法[3]

做好跳钢管舞的准备[4]

避免涂抹身体油或者保湿乳。[10]
- 跳钢管舞之间不要涂抹任何身体油或者保湿乳。这些东西会让你从钢管上滑落下来，造成危险。在开始之间，擦拭一下钢管，确保上面没有任何油腻物质。[11]

对话历史

ID	详细信息	当前角色	编写的话
0	查看	用户	如果想在家跳钢管舞怎么办?
1	查看	系统	如果你想在家自由自在地练习钢管舞，你需要购买一个自由站
2	查看	用户	穿什么衣服合适?
3	查看	系统	做准备的一个重要部分之一，就是穿着，你需要穿着舒适的衣
4	查看	用户	我听朋友说跳的时候不能涂身体油或者保湿乳，这是为什么呢

系统

您当前扮演系统角色，基于当前目标和对话历史，您可以决定系统动作，请选择一种情况:

1. 用户的需求不够明确，系统需要知道更多关于 [T38][1][3][4][11] 的信息
2. 系统需要澄清某些条件，请点击 此处
3. 系统直接基于 目标片段 进行回答
4. 不满足以上三种情况，系统需要询问开放性问题，请点击 此处。

推荐的系统动作

请根据目标片段回答用户问题:

跳钢管舞之间不要涂抹任何身体油或者保湿乳。这些东西会让你从钢管上滑落下来，造成危险。在开始之间，擦拭一下钢管，确保上面没有任何油腻物质。[11]

系统说: 身体油或者保湿乳可能会让你从钢管上滑落下来，造成危险。

对话动作: 回答/开放性

系统用以询问或回答的片段: 11

☑ 系统已经完成当前目标

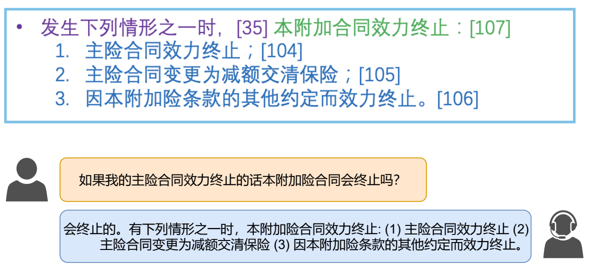
Figure 7: Annotation interface to write a system sentence. Compared to the interface of user turn, we will have more suggestions to help crowdsourcers make up an utterance, which is shown in the red box.

Model	**Variant**	**Parameters**	**HuggingFace Pre-trained Model**	**Corpus**
BERT	base	102M	`hfl/chinese-bert-wwm-ext`	-
RoBERTa	base	102M	`hfl/chinese-roberta-wwm-ext`	-
ELECTRA	base	102M	`hfl/chinese-electra-base-discriminator`	-
Pegasus	base	214M	`uer/pegasus-base-chinese-cluecorpussmall`	14G
BART	base	116M	`fnlp/bart-base-chinese`	200G
T5	base	215M	`uer/t5-base-chinese-cluecorpussmall`	14G

Table 6: The detail of the pre-trained model for all baselines. Exact corpus size of the RoBERTa model is not reported by the owner.

Task	**Model**	**Best Epoch**	**Runtime**		**Score**	
			Training	**Inference**	**Validation**	**Test**
DST	DPR	47 / 50	21.3h	0.1h	**75.06**	73.81
	BERT$_{act}$	3 / 5	1.6h	< 0.1h	**89.91**	89.52
	BERT$_{bm25}$	5 / 5	17.3h	4.3h	**89.69**	89.39
	BERT$_{dpr}$	3 / 5	16.8h	3.3h	**84.52**	83.67
	RoBERTa$_{act}$	5 / 5	1.7h	< 0.1h	**89.73**	89.38
	RoBERTa$_{bm25}$	4 / 5	19.4h	4.2h	**90.32**	89.95
	RoBERTa$_{dpr}$	2 / 5	18.4h	2.0h	**84.78**	83.99
	ELECTRA$_{act}$	5 / 5	1.6h	< 0.1h	87.34	**87.88**
	ELECTRA$_{bm25}$	4 / 5	17.4h	3.9h	**90.50**	90.09
	ELECTRA$_{dpr}$	3 / 5	16.6h	3.5h	**85.29**	84.31
DPL	BERT$_{act}$	2 / 5	1.7h	< 0.1h	81.61	**82.58**
	BERT$_{full}$	5 / 5	25.0h	0.5h	**89.16**	81.40
	- structure	5 / 5	26.6h	0.6h	**82.75**	75.61
	- dialog state	5 / 5	16.5h	0.3h	**84.48**	73.14
	RoBERTa$_{act}$	5 / 5	1.5h	< 0.1h	82.38	**82.87**
	RoBERTa$_{full}$	5 / 5	24.3h	0.5h	**89.67**	81.65
	- structure	5 / 5	30.8h	0.6h	**84.44**	77.18
	- dialog state	5 / 5	18.7h	0.3h	**84.86**	73.71
	ELECTRA$_{act}$	5 / 5	1.7h	< 0.1h	**81.96**	81.91
	ELECTRA$_{full}$	5 / 5	22.5h	0.5h	**88.28**	81.31
	- structure	5 / 5	26.6h	0.6h	**82.24**	75.66
	- dialog state	5 / 5	15.7h	0.3h	**83.63**	73.17
RG	Pegasus	18 / 20	13.0h	1.4h	52.39	**52.63**
	Pegasus$_{\texttt{w/o act}}$	18 / 20	12.6h	1.4h	**51.41**	51.27
	BART	19 / 20	10.7h	0.8h	53.04	**53.13**
	BART$_{\texttt{w/o act}}$	16 / 20	13.1h	0.8h	51.72	**51.81**
	T5	19 / 20	12.0h	1.2h	53.18	**53.53**
	T5$_{\texttt{w/o act}}$	20 / 20	13.8h	1.2h	51.77	**52.16**

Table 7: The average runtime and validation performance for all baseline models. For the validation score, we use a sample-level F1 score for the dialog state tracking and dialog policy learning baselines. For response generation, we use the average value of BLEU-{1, 2, 4}. We also give the sample-level score of the test set, as a comparison of the validation set results.



- 发生下列情形之一时，[35] 本附加合同效力终止：[107]
  1. 主险合同效力终止；[104]
  2. 主险合同变更为减额交清保险；[105]
  3. 因本附加险条款的其他约定而效力终止。[106]

如果我的主险合同效力终止的话本附加险合同会终止吗?

会终止的。有下列情形之一时，本附加险合同效力终止: (1) 主险合同效力终止 (2) 主险合同变更为减额交清保险 (3) 因本附加险条款的其他约定而效力终止。

Figure 8: Example of a condition-solution case, where the type of the purple text is condition/disjunction, of the green text is solution and of the blue text is condition.

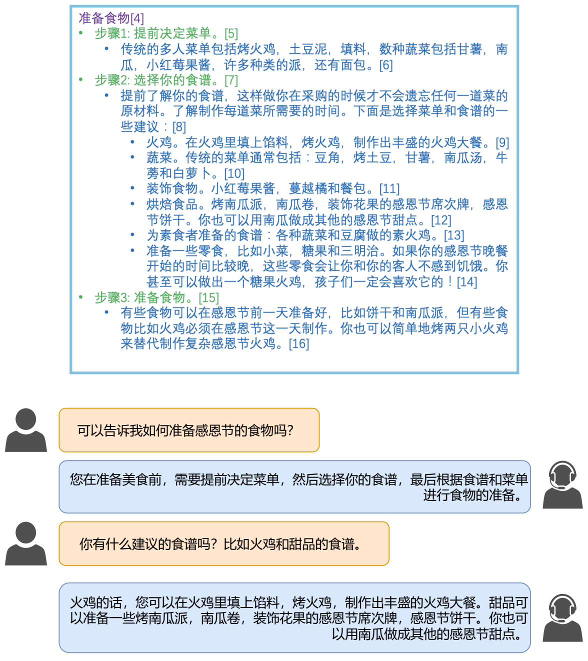
C.1 Understanding Documents

Understanding Condition-Solution the agent needs to be able to recognize fundamental parts of consolution-solution structures and exploit appropriate information to generate system responses.

In Figure 8, the user asks *"If the effectiveness of my main insurance contract is terminated, will this additional contract be terminated?"* (如果我的主险合同效力终止的话本附加险合同会终止吗？). Here, the user intention is to know whether he/she meets the conditions of termination of the additional contract. According to the given document, there are three conditions for the termination of the effectiveness of the additional contract. These conditions form a disjunction condition, and the user meets the first condition. Therefore, it is necessary to inform the user that the conditions for termination of this additional contract are met. Here, the agent replies *"Yes, it will be terminated"* (会终止的), with an additional introduction of all the conditions for the termination of the additional contract.

Understanding Sequence-Step The agent needs to recognize the sequences and its steps, and know to answer followup questions about specific steps.

In Figure 9, the user asks *"Can you tell me how to prepare Thanksgiving food?"* (可以告诉我如何准备感恩节的食物吗？). The type of ground node is sequence, whose content is *"prepare food"*(准备食物). The agent needs to summarize the steps (the green text) to respond. The user follow-up query is *"Do you have any suggested recipes? Like Turkey and desserts."* (你有什么建议的食谱吗？比如火鸡和甜品的食谱), targeting on the second step: *"Choose your recipe"* (选择你的食谱). The second step node has two child nodes, node [10] and node [12]. Here, node[10] is *"Turkey. Fill the turkey with stuffing, roast the turkey, and make a rich Turkey meal."* (火鸡。在火鸡里填上馅料，烤火鸡，制作出丰盛的火鸡大餐。). And node [12] is *"Baked goods. Baked pumpkin pie, pumpkin roll, Thanksgiving dinner plate decorated with flowers, Thanksgiving cookies. You can also make other Thanksgiving desserts out of pumpkins."* (烘焙食品。烤南瓜派，南瓜卷，装饰花朵的感恩节席次牌，感恩节饼干。你也可以用南瓜做成其他感恩节甜点。). The agent needs to combine the information from the two nodes to answer.



准备食物[4]
- 步骤1: 提前决定菜单。[5]
  - 传统的多人菜单包括烤火鸡，土豆泥，填料，数种蔬菜包括甘薯，南瓜，小红莓果酱，许多种类的派，还有面包。[6]
- 步骤2: 选择你的食谱。[7]
  - 提前了解你的食谱，这样做你在采购的时候才不会遗忘任何一道菜的原材料。了解制作每道菜所需要的时间。下面是选择菜单和食谱的一些建议：[8]
    - 火鸡。在火鸡里填上馅料，烤火鸡，制作出丰盛的火鸡大餐。[9]
    - 蔬菜。传统的菜单通常包括：豆角，烤土豆，甘薯，南瓜汤，牛蒡和白萝卜。[10]
    - 装饰食物。小红莓果酱，蔓越橘和餐包。[11]
    - 烘焙食品。烤南瓜派，南瓜卷，装饰花果的感恩节席次牌，感恩节饼干。你也可以用南瓜做成其他的感恩节甜点。[12]
    - 为素食者准备的食谱：各种蔬菜和豆腐做的素火鸡。[13]
    - 准备一些零食，比如小菜，糖果和三明治。如果你的感恩节晚餐开始的时间比较晚，这些零食会让你和你的客人不感到饥饿。你甚至可以做出一个糖果火鸡，孩子们一定会喜欢它的！[14]
- 步骤3: 准备食物。[15]
  - 有些食物可以在感恩节前一天准备好，比如饼干和南瓜派，但有些食物比如火鸡必须在感恩节这一天制作。你也可以简单地烤两只小火鸡来替代制作复杂感恩节火鸡。[16]

可以告诉我如何准备感恩节的食物吗?

您在准备美食前，需要提前决定菜单，然后选择你的食谱，最后根据食谱和菜单进行食物的准备。

你有什么建议的食谱吗？比如火鸡和甜品的食谱。

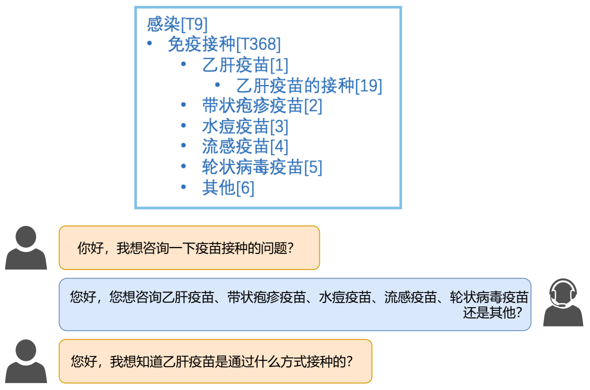
火鸡的话，您可以在火鸡里填上馅料，烤火鸡，制作出丰盛的火鸡大餐。甜品可以准备一些烤南瓜派，南瓜卷，装饰花果的感恩节席次牌，感恩节饼干。你也可以用南瓜做成其他的感恩节甜点。

Figure 9: Example of a sequence-step case, where the type of the purple text is sequence, the green text is step and the blue text is ordinary.

C.2 Understanding Dialog Context

User Under-Specified Requests The user request could be under-specified, and so the agent needs to decide to either ask clarifying questions or provide answer.

In Figure 10, the user first asks *"Hello, I'd like to ask about vaccination."* (你好，我想咨询一下疫苗接种的问题。). Since the question is too general, the system asks the user to choose among several choices of vaccines: *"Hello, would you like to consult hepatitis B vaccine, herpes zoster vaccine, chickenpox vaccine, influenza vaccine, rotavirus vaccine or others?"* (您好，您想咨询乙肝疫苗、带状疱疹疫苗、水痘疫苗、流感疫苗、轮状病毒疫苗还是其他？). The user selects one and replies *"Hello, I want to know how the hepatitis B vaccine is vaccinated."* (您好，我想知道乙肝疫苗是通过什么方式接种的。).


感染[T9]
- 免疫接种[T368]
    - 乙肝疫苗[1]
        - 乙肝疫苗的接种[19]
    - 带状疱疹疫苗[2]
    - 水痘疫苗[3]
    - 流感疫苗[4]
    - 轮状病毒疫苗[5]
    - 其他[6]

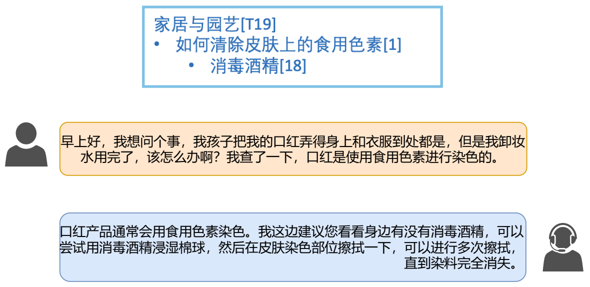
你好，我想咨询一下疫苗接种的问题?

您好，您想咨询乙肝疫苗、带状疱疹疫苗、水痘疫苗、流感疫苗、轮状病毒疫苗还是其他?

您好，我想知道乙肝疫苗是通过什么方式接种的?

Figure 10: Example of an under-specified request.

Textual Entailment The agent needs to be able to recognize if a node text entails the user utterance, thus requiring deeper semantic understanding than text matching. In Figure 11, the user asks *"Good morning. I want to ask you something. My child has made my lipstick all over my body and clothes, but I have run out of makeup remover. What should I do? I checked that lipstick is dyed with food coloring."* (早上好，我想问个事，我孩子把我的口红弄得身上和衣服到处都是，但是我卸妆水用完了，该怎么办啊？我查了一下，口红是使用食用色素进行染色的。). From "lipstick is everywhere" and "I have run out of makeup remover", we can know that the user intent is to remove the traces of lipstick. In combination with "I checked that lipstick is dyed with food coloring", we know that the user request entails *"How to clear the food pigment on the skin"* (如何清除皮肤上的食用色素).


家居与园艺[T19]
- 如何清除皮肤上的食用色素[1]
    - 消毒酒精[18]

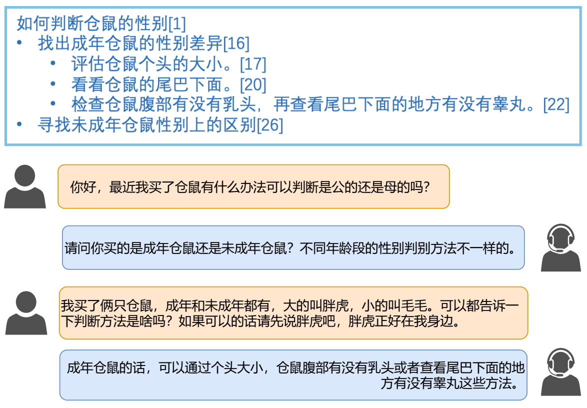
早上好，我想问个事，我孩子把我的口红弄得身上和衣服到处都是，但是我卸妆水用完了，该怎么办啊？我查了一下，口红是使用食用色素进行染色的。

口红产品通常会用食用色素染色。我这边建议您看看身边有没有消毒酒精，可以尝试用消毒酒精浸湿棉球，然后在皮肤染色部位擦拭一下，可以进行多次擦拭，直到染料完全消失。

Figure 11: Example of a textual entailment case

Coreference The phenomenon of coreference could occur in a number of user queries. The agent needs to identify these cases and resolve such coreferences for better text understanding.

In Figure 12, user asks *"Hello, I bought a hamster recently. Is there any way to judge whether it is a male or a female?"* (你好，最近我买了仓鼠有什么办法可以判断是公的还是母的？). Since the current user questions are under specified, the agent guides the user to clarify whether the hamster is an adult. Then the user replies *"I bought two hamsters, an adult and a baby. The big one is Panghu and the small one is Maomao. Can you tell me what the judgment method is? If you can, please tell me Panghu first. Panghu is right beside me."* (我买了俩只仓鼠，成年和未成年都有，大的叫胖虎，小的叫毛毛。可以都告诉一下判断方法是啥吗？如果可以的话请先说胖虎吧，胖虎正好在我身边。). In this sentence, Panghu and adult refer to the same hamster. And he/she wants to know the answer for the adult hamster first, so the system needs resolve coreference to provide correct answer.


如何判断仓鼠的性别[1]
- 找出成年仓鼠的性别差异[16]
    - 评估仓鼠个头的大小。[17]
    - 看看仓鼠的尾巴下面。[20]
    - 检查仓鼠腹部有没有乳头，再查看尾巴下面的地方有没有睾丸。[22]
- 寻找未成年仓鼠性别上的区别[26]

你好，最近我买了仓鼠有什么办法可以判断是公的还是母的吗?

请问你买的是成年仓鼠还是未成年仓鼠？不同年龄段的性别判别方法不一样的。

我买了俩只仓鼠，成年和未成年都有，大的叫胖虎，小的叫毛毛。可以都告诉一下判断方法是啥吗？如果可以的话请先说胖虎吧，胖虎正好在我身边。

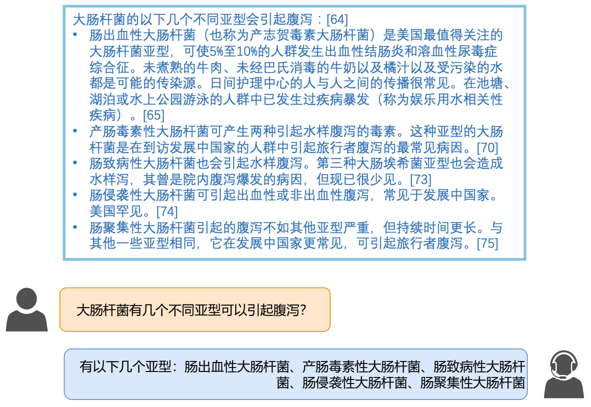
成年仓鼠的话，可以通过个头大小，仓鼠腹部有没有乳头或者查看尾巴下面的地方有没有睾丸这些方法。

Figure 12: Example of a situation where coreference analysis is needed for understanding user request.

C.3 Abstractive Response Generation

Summarization If the number of grounding nodes for a turn is large, the system needs to be able to summarize the information for more concise responses.

In Figure 13, the user wants to know how many different subtypes of E. coli can cause diarrhea (大肠杆菌有几个不同亚型可以引起腹泻？). Since there are 5 grounding nodes, including [65], [70], [73], [74] and [75], the agent needs to summarize information, and give more concise response: *"There are the following subtypes: EHEC, enterotoxigenic Escherichia coli, enteropathogenic Escherichia coli, enteroinvasive Escherichia coli, and enteroaggregative Escherichia coli."* (有以下几个亚型：肠出血性大肠杆菌、产肠毒素性大肠杆菌、肠致病性大肠杆 菌、肠侵袭性大肠杆菌、肠聚集性大肠杆菌。).



大肠杆菌的以下几个不同亚型会引起腹泻：[64]
- 肠出血性大肠杆菌（也称为产志贺毒素大肠杆菌）是美国最值得关注的大肠杆菌亚型，可使5%至10%的人群发生出血性结肠炎和溶血性尿毒症综合征。未煮熟的牛肉、未经巴氏消毒的牛奶以及橘汁以及受污染的水都是可能的传染源。日间护理中心的人与人之间的传播很常见。在池塘、湖泊或水上公园游泳的人群中已发生过疾病暴发（称为娱乐用水相关性疾病）。[65]
- 产肠毒素性大肠杆菌可产生两种引起水样腹泻的毒素。这种亚型的大肠杆菌是在到访发展中国家的人群中引起旅行者腹泻的最常见病因。[70]
- 肠致病性大肠杆菌也会引起水样腹泻。第三种大肠埃希菌亚型也会造成水样泻，其曾是院内腹泻爆发的病因，但现已很少见。[73]
- 肠侵袭性大肠杆菌可引起出血性或非出血性腹泻，常见于发展中国家。美国罕见。[74]
- 肠聚集性大肠杆菌引起的腹泻不如其他亚型严重，但持续时间更长。与其他一些亚型相同，它在发展中国家更常见，可引起旅行者腹泻。[75]

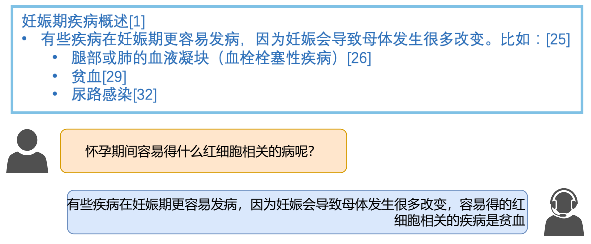
大肠杆菌有几个不同亚型可以引起腹泻?

有以下几个亚型：肠出血性大肠杆菌、产肠毒素性大肠杆菌、肠致病性大肠杆菌、肠侵袭性大肠杆菌、肠聚集性大肠杆菌

Figure 13: Example of a situation where the agent needs to answer by summarizing the information.

Common-Sense Knowledge For some user requests, common-sense knowledge is needed for agents to reason the next system response.

In Figure 14, the user asks *"What red blood cell related diseases are easy to get during pregnancy?"* (怀孕期间容易得什么红细胞相关的病呢？). In the document, we can identify three diseases that are easy to get during pregnancy： 1. Blood clots in the legs or lungs (thromboembolic disease)（腿部或肺的血液凝块（血栓栓塞性疾病)) 2. Anemia（贫血） 3. Urinary tract infection（尿路感染）. The fact that "Anemia is a red blood cell related disease" is not in the document, but it is a common knowlwedge. So the response is *"Some diseases are more likely to occur during pregnancy since it will lead to many changes in the mother. The easily acquired red blood cell related disease is anemia."* (有些疾病在妊娠期更容易发病，因为妊娠会导致母体发生很多改变，容易得的红细胞相关的疾病是贫血。).



妊娠期疾病概述[1]
- 有些疾病在妊娠期更容易发病，因为妊娠会导致母体发生很多改变。比如：[25]
    - 腿部或肺的血液凝块（血栓栓塞性疾病）[26]
    - 贫血[29]
    - 尿路感染[32]

怀孕期间容易得什么红细胞相关的病呢?

有些疾病在妊娠期更容易发病，因为妊娠会导致母体发生很多改变，容易得的红细胞相关的疾病是贫血

Figure 14: Example of a situation where common-sense knowledge is needed.