# AmbigChat: Interactive Hierarchical Clarification for Ambiguous Open-Domain Question Answering


Jiaju Ma*
Stanford University
Stanford, CA, USA

Lei Shi
Google DeepMind
Mountain View, CA, USA

Kenneth Robertsen
Google DeepMind
San Francisco, CA, USA

Peggy Chi
Google DeepMind
Mountain View, CA, USA


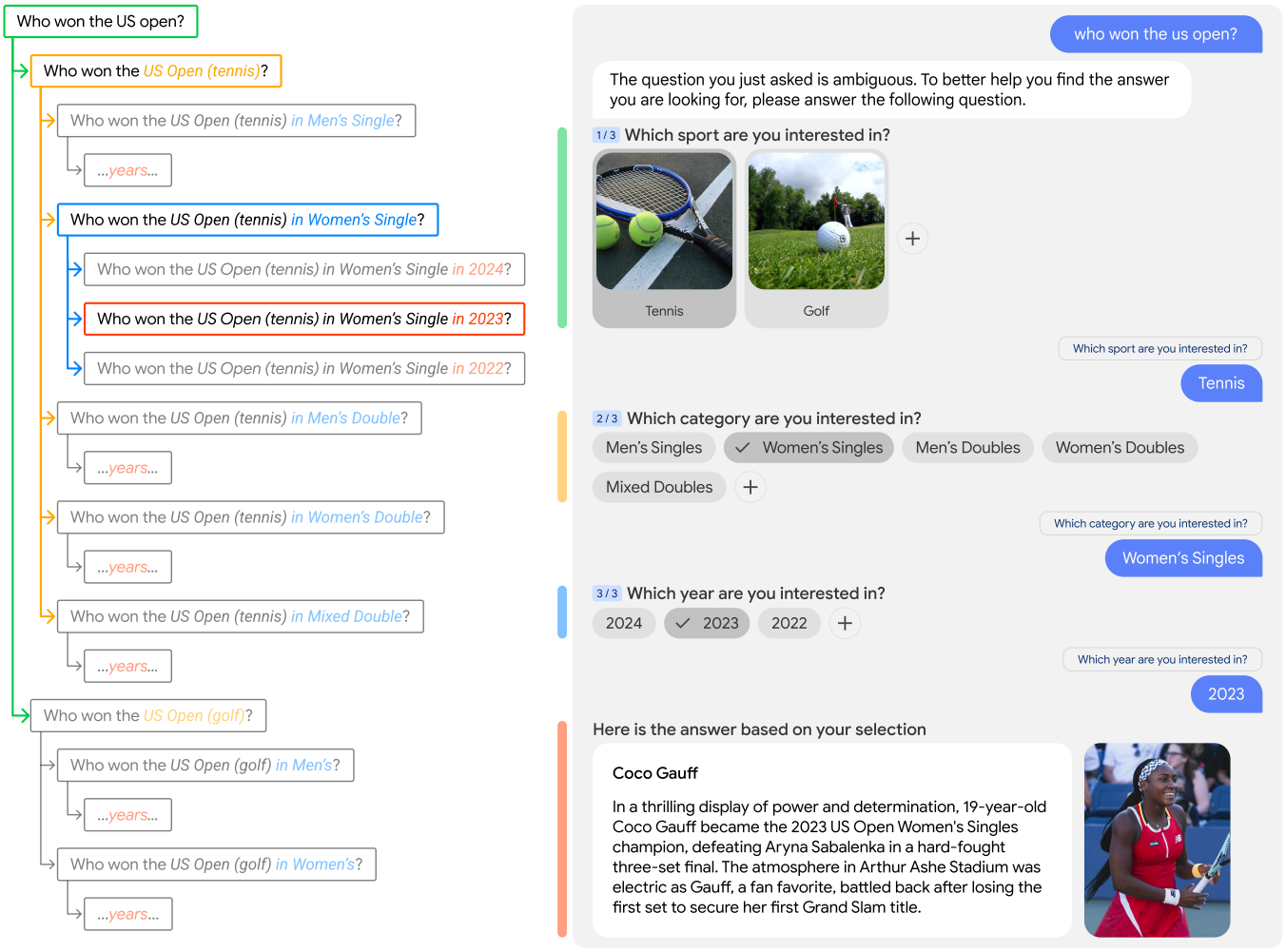


**Figure 1: *AmbigChat* is an automatic approach that hierarchically disambiguates an open-domain question (*left*) and surfaces interactive *disambiguation widgets* in a multi-turn conversational interface with an LLM. It supports users in both accurately finding answers and structurally exploring knowledge (*right*). *Image Credit: Wikimedia Commons, Tennis / Golf / Coco Gauff.***


## Abstract

When conversing with large language models, it is common for users to ask an ambiguous open-domain question that could lead to multiple answers, especially when exploring new topics. For example, "*Who won the US Open?*" can result in different athletes according to the referenced events and years. We propose AmbigChat, an automatic approach that hierarchically disambiguates a factual question and guides users to navigate answers via UI widgets in a multi-turn conversational interface. Using the ambiguity


*This work was done while the author was a Student Researcher at Google DeepMind.



*UIST '25, Busan, Republic of Korea*



ACM ISBN 979-8-4007-2037-6/2025/09
https://doi.org/10.1145/3746059.3747686

taxonomy we generated from an analysis of 5,000 queries, AmbigChat identifies ambiguous facets of a question and constructs a *disambiguation tree*, where each level corresponds to a facet. Users can traverse the tree to explore answers via interactive disambiguation widgets and expand the conversation by referencing tree nodes through drag and drop. We iterated our interaction design with six design professionals and tested the effectiveness of the disambiguation tree generation algorithm on a variety of factual queries. Our evaluation with 16 participants shows that AmbigChat not only helps the participants find answers more easily and efficiently, but also facilitates structured explorations of the topic space.

## CCS Concepts

• **Human-centered computing** → **Interactive systems and tools**; *Interaction design.*

## Keywords

Language ambiguity, factual question answering, conversational interfaces, large language model

**ACM Reference Format:**

Jiaju Ma, Lei Shi, Kenneth Robertsen, and Peggy Chi. 2025. AmbigChat: Interactive Hierarchical Clarification for Ambiguous Open-Domain Question Answering. In *The 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25), September 28-October 1, 2025, Busan, Republic of Korea.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3746059.3747686

## 1 Introduction

To acquire knowledge of an unfamiliar domain, people commonly ask questions through conversations. Recent advances in large language models (LLMs) have greatly enhanced the capabilities of systems for open-domain question answering (ODQA) [17], allowing users to efficiently retrieve factual information across diverse topics. Yet, because of the inherent ambiguity of natural language and the challenges of writing appropriate prompts [39], it is common for a user prompt of a factual question to lead to multiple possible answers [17, 41]. For example, a short question like "*Who won the US Open?*" could refer to different athletes given the sport types (e.g., tennis or golf), event categories (e.g., women's singles or mixed doubles), and year (e.g., 2024).

To resolve such ambiguities, existing conversational question answering (QA) systems ask a series of clarification questions in the text format until an answer is reached [17]. To disambiguate the example question "*Who won the US Open?*", a system might ask "*Which sport type are you interested in?*" and "*Which event category are you looking for?*" However, this linear and text-only approach optimized solely for arriving at a single answer fails to support the multimodal and exploratory nature of human conversations, as people often engage multiple modalities (pointing at objects, making facial expressions, etc.) to clarify their intentions and ask for related answers and follow-up questions to learn more about a topic [3, 11]. As recent works investigated integrating visual aids and UI widgets for clarification in other domains [7, 18, 22], it remains an open question how to employ visual and interactive modalities to support open-domain question disambiguation for both target answer finding and exploratory knowledge discovery.

In this work, we propose AmbigChat, an automatic approach that disambiguates a user query in a hierarchical manner and generates UI widgets for interactive guidance in a conversational interface (see Figure 1). As a query can be ambiguous in many different aspects (referred to as facets [26]), we first develop a *taxonomy of ambiguous query facets* (see Table 1) by analyzing nearly 5,000 queries from AmbigNQ, an ambiguous ODQA dataset [17]. We use this taxonomy to construct a *disambiguation tree* that organizes various interpretations of an input user query and their answers in a hierarchical manner. Given a query, we detect its ambiguous facets by guiding an LLM (Gemini 1.5 Pro) with our taxonomy. We then convert these facets into disambiguated rewrites and answers in a breadth-first manner; each level of the tree corresponds to an ambiguous facet, and each leaf node represents an answer. We use web search results to improve the LLM's factual accuracy throughout this process via retrieval-augmented generation (RAG) [14].

To guide users to interactively explore information stored in the disambiguation trees, we propose a set of interactive techniques in the form of *disambiguation widgets* that combine text prompting and GUI interaction, which we co-designed through a design study with six user experience (UX) professionals. These include *question widgets* for asking clarifying questions and *answer widgets* for displaying answers with rich visual information (see Figure 1). To encourage follow-up questions, every answer widget and option in the question widgets is draggable as *context chips* for direct reference. For example, the user can drag multiple answer widgets and ask, "*Compare their performance:* Coco Gauff *or* Aryna Sabalenka " (see Figure 3b). AmbigChat automatically expands the user prompt with the current conversation context and the referenced nodes' positions in the disambiguation tree for the LLM to generate new responses (see Figure 3c).

We verify the correctness and completeness of our ambiguous query facet taxonomy with human raters, and test our disambiguation tree construction pipeline's ability to answer ambiguous questions on the AmbigNQ dataset. We demonstrate the capability of AmbigChat through example conversations on a wide range of topics. Finally, we evaluate AmbigChat with 16 participants to examine its interaction usability, and the results show that AmbigChat is effective at facilitating both target answer acquisition and structured knowledge exploration.

In summary, we make the following contributions:

- A taxonomy of ambiguous query facets summarized from nearly 5,000 queries in the AmbigNQ dataset [17].
- An automated pipeline that converts an ambiguous query into a disambiguation tree based on the taxonomy.
- A set of interaction techniques to respond to ambiguous prompts and support user navigation and clarification in a conversational UI, informed by a design study with UX professionals.
- A set of evaluations of our approach, including a verification study on the quality of our taxonomy, a technical evaluation of the disambiguation tree construction pipeline, and a user study on the usability of our interaction techniques.

## 2 Related Work

AmbigChat draws insight from prior work on ODQA, conversational interface design, and interactive UI widgets.

### 2.1 Clarification Techniques for ODQA

Open-domain question answering, or ODQA, supports a diverse set of user needs, from fact acquisition to open-ended knowledge discovery from textual sources [17, 32]. Yet, these questions are often underspecified, resulting in many factually valid answers [32, 40]. To help users arrive at their desired answers, prior work has proposed a family of approaches that ask a series of clarification questions for disambiguation [3, 26, 41]. We build upon this general approach for ambiguous queries to support both accurate answer acquisition and exploratory knowledge expansion [3, 11, 24, 38].

Earlier question answering (QA) systems ask clarifying questions without providing suggestive options [3, 26]. They require users to have sufficient domain knowledge to answer these questions, which can be challenging when exploring unfamiliar topics. To mitigate this problem, later systems generate a few suggestive options alongside the clarification questions [13, 41]. For example, the technique proposed by Lee et al. generates one clarification question with options each turn. However, this single-turn, greedy approach may result in a long disambiguation process without any progress indicator. Zhao et al. improve this approach by generating one clarifying question for each underspecified aspect (i.e., *facet*) of a user query to minimize redundant disambiguation turns [41]. While they provide suggestive options as UI chips for users to select, their system does not indicate the progress of clarification.

In our work, we generate facets of a question by supplying the LLM with web search results and our *ambiguous facet taxonomy*. Inspired by prior work [12, 37], we organize the facets with clarification questions and answers into a hierarchical *disambiguation tree*, where each level corresponds to a facet. As a result, we can indicate the number of clarification questions and enable users to backtrack and explore different answers in the topic space.

### 2.2 Multimodal Conversational UI

Recent works have integrated modalities other than text into multi-turn conversations to facilitate information seeking and enhance user experiences. Commercial LLM-based chatbots such as Gemini [34], Perplexity [1], and ChatGPT [20] are capable of responding to conversation topics with relevant website links and images. Project Bespoke further offers multi-modal GUI experiences with interactive widgets based on the conversational context using Gemini [19]. Claude's Artifacts and Gemini's Canvas features allow users to iteratively work with the LLM chatbot on artifacts, such as code and web apps, in the form of a panel next to the main conversation [9, 23].

In research literature, Macaw is an open-source framework that supports multimodal interactions in chat interfaces [38]. Users can input text or voice and the system responds with retrieved documents and web page links with previews. Other prior work has explored integrating visualizations into conversations [8, 42]. GameBot is a sports-focused chatbot that displays game statistical data when answering user questions [42]. For example, it can display a team's upcoming schedule or a shot chart of a certain play in a basketball game. Hearst et al. have studied the effect of displaying charts and graphs alongside textual answers to user queries to show trends and comparisons [8]. They find that approximately 60% of the participants prefer to see charts and appreciate the additional context provided by the visualizations.

Inspired by these methods, we enhance the text-based chat experiences with images and visualizations in the form of UI widgets. We display reference images with textual answers and visualize disambiguation options with icons based on the type of ambiguous facet to help users more easily and efficiently make their selections.

### 2.3 Generative UI Widgets

Prior work has explored integrating UI widgets to facilitate user workflows. Vaithilingam and Guo convert command-line inputs to GUIs via user demonstrations [36]. They use a rule-based approach to generate lightweight interfaces to make editing command parameters easier. SneakPique supports autocompletion of data visualization generation queries [27]. As the user types out a query, the system generates suggestive options, which are visualized as UI widgets to let the user quickly preview the result of selecting an option. DynaVis synthesizes UI widgets using LLMs from user's editing command in natural language [35]. Users can interact with the generated widgets to perform the desired data visualization edits. Similarly, BISCUIT generates UI widgets to facilitate learning machine learning code in Jupyter notebooks [4]. Based on natural language commands, widgets in BISCUIT surface important model parameters for users to tune or generate an adjustable graph to visualize the training process. In AmbigChat, we generate UI widgets based on our disambiguation trees to facilitate question answering.

DataTone is closely related to our work [7]. Given a dataset such as Olympics medal statistics, DataTone creates data visualizations based on the user's natural language query, such as "show me medal for hockey and skating by country." It detects ambiguities in words such as "medal", "hockey", and "skating" and offers disambiguated options like "gold medal", "ice hockey", and "figure skating" through dropdown menus. The visualizations are updated accordingly based on user selections. Mitra et al. later recreated DataTone with their NL4DV framework as an example application [18].

Although sharing the same topic of disambiguation, our method differs significantly from DataTone. We focus on the task of facilitating ambiguous question answering in the open domain. DataTone's disambiguation mechanism is confined between the natural language query and data entries and operations in the designated dataset, while we design a hierarchical disambiguation approach grounded by retrieved documents from the open web. Moreover, we allow users to iteratively edit the disambiguation trees via option expansion in the question widgets.

## 3 Understanding Question Ambiguity

A factual question can be ambiguous in many aspects, such as sports types, event categories, and year (see Figure 1). We refer to these aspects as *facets*. To comprehensively extract ambiguous facets and generate corresponding clarifying questions, we investigated how facets can be underspecified. We analyzed data from the AmbigNQ dataset [17] to develop a *taxonomy of ambiguous query facets* (see Table 1), which we verified and refined with 10 annotators.

**Table 1: *Taxonomy of Ambiguous Query Facets*. We developed this taxonomy by summarizing facets from queries in the *train* set of AmbigNQ [17] and refined and validated the coverage and correctness of this taxonomy with annotators.**

| | Type | Definition | Example |
|---|---|---|---|
| Question Nouns | **Entity Reference** | Multiple entities, such as people, places, and events can share the same name | When did the *US Open* take place?<br>DQ1: When did the *US Open Tennis Championship* take place?<br>DQ2: When did the *US Open Golf Championship* take place? |
| | **Part of Entity Reference** | For a specified entity, there can be different parts or variations of that entity | Who won the *Paris Olympics*?<br>DQ1: Who won the *Paris Olympics Women's 100m*?<br>DQ2: Who won the *Paris Olympics Men's Marathon*? |
| | **Relationships Between Entities** | The same set of entities can have different relationships with each other | Who plays *Lara Croft* in the new Tomb Raider?<br>DQ1: Who plays Lara Croft in the new Tomb Raider *as a 14-year-old*?<br>DQ2: Who plays Lara Croft in the new Tomb Raider *as an adult*? |
| | **Underspecified Common Nouns** | Common nouns define a class of objects and can be underspecified | When did Ariana Grande's *new album* come out?<br>DQ1: When did Ariana Grande's *new live album* come out?<br>DQ2: When did Ariana Grande's *new remix album* come out? |
| Question Verbs | **Degree of an Action** | The degree of how much of an action is performed can be ambiguous, such as a missing adverb | When did the gold standard *end* in the US?<br>DQ1: When did the gold standard *begin to end* in the US?<br>DQ2: When did the gold standard *end temporarily* in the US?<br>DQ3: When did the gold standard *completely end* in the US? |
| | **Means of an Action** | The manner in which the action is performed can be ambiguous | When was the great pacific garbage patch *found*?<br>DQ1: When was the great pacific garbage patch *found to exist by hypothesis*?<br>DQ2: When was the great pacific garbage patch *found to exist in person*? |
| Adverbs | **Output Type** | The question adverbs can be unclear about what types of answers should be given | *When* did the runner up stop becoming vice president?<br>DQ1: *In what year* did the runner up stop becoming vice president?<br>DQ2: *What event* caused the runner up to stop becoming vice president? |
| Question Scope | **Temporal Dependency** | The timeframe in which the question situates can result in different answers | How long is the term for Texas Governor?<br>DQ1: How long was the term for the Texas Governor *between 1876 and 1972*?<br>DQ2: How long was the term for the Texas Governor *between 1972 and now*? |
| | **Geographical Dependency** | The geographical region in which the question situates changes answers | When was the Tomb Raider (2018) movie released?<br>DQ1: When was the movie Tomb Raider (2018) released *in Berlin*?<br>DQ2: When was the movie Tomb Raider (2018) released *in the UK*? |
| | **Information Source Dependency** | The answer may change based on which information source is referenced | Where does India rank in the world economy?<br>DQ1: Where does India rank in the world economy *according to IMF*?<br>DQ2: Where does India rank in the world economy *according to World Bank*? |

## 3.1 Facet Summarization

Sekulić et al. created the ClariQ-Fkw dataset [26] by extracting facets from each pair of a query and its corresponding clarifying questions in the ClariQ dataset [2]. In AmbigNQ [17], each data entry consists of an original question and pairs of disambiguated questions and answers. We adopted the ClariQ-Fkw approach to summarize facets for AmbigNQ. For each entry in the *train* set (4,749 total entries with multiple answers), we performed few-shot prompting with Gemini 1.5 Pro to extract a list of facets for that entry. For example, we obtained `[Sport Type, Event Category, Year]` from the entry `{"question": "Who won the US Open?", "qa-pairs": [("Who won the US Open Tennis Women's Singles in 2023?", "Coco Gauff"), ("Who won the US Open Tennis Men's Singles in 2017?", "Rafael Nadal")]}`. At the same time, we used SBERT [25] to group the data entries into clusters based on the embedding similarities between the original questions of each data entry. We organized the 4,749 entries into 922 clusters. Next, we used the LLM to summarize the extracted facets of queries in the same cluster into higher-level facet categories. For example, we obtained *Temporal Information* as one of the facet categories from a cluster with questions related to sporting event winners. Finally, the authors went through the summarized facet categories of each cluster to refine them into a preliminary version of the taxonomy.

## 3.2 Taxonomy Verification With Human Raters

To refine and verify the correctness and completeness of this preliminary taxonomy, we conducted a remote verification study with 10 annotators in our organization. We randomly sampled 100 data entries from the *dev* set of AmbigNQ and prompted Gemini 1.5 Pro to identify the ambiguities in each of the disambiguated QA pairs by supplying it with our preliminary taxonomy. We manually corrected errors in the identification results. Next, we distributed the results evenly and randomly to the annotators such that each result was reviewed by two of them. For each entry, we asked the annotator to label if all the ambiguities between the original and disambiguated question pair had been identified and labeled with the correct facet type. If not, we asked them to list any missed or incorrectly labeled ambiguities or propose a new category.

We found that 81% of the question pairs in our 100 entries were labeled as correct by two randomly assigned annotators. We manually examined the remaining 19 questions that the raters disagreed on. The primary disagreements involved how to classify identified ambiguities (e.g., *Entity Reference* vs. *Part of Entity Reference*) and whether certain ambiguities had been missed or mislabeled. To address this, we updated certain definitions of our taxonomy and provided clearer examples. Two raters proposed new taxonomy types. One proposed coreference resolution ambiguity (i.e., how to identify which entities referential expressions like pronouns refer

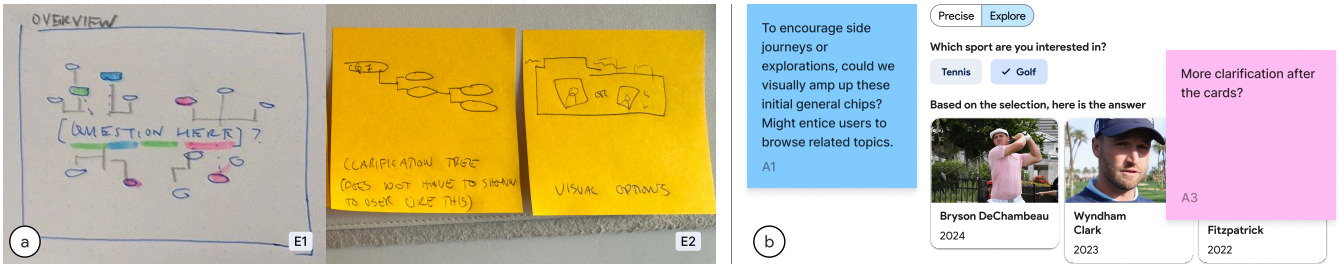


**Figure 2: During the design study interviews, the designers sketched out potential designs for the disambiguation widgets. (a) depicts sketches by E1 and E2 suggesting hierarchical disambiguation. (b) shows selected feedback we gathered on our initial prototype in the design review session. Note our previous design that uses a toggle button to separate answer finding from exploration. *Image Credit: Wikimedia Commons, Bryson DeChambeau / Wyndham Clark.***

to). The other mentioned that a lack of background knowledge might cause ambiguity when technical terms are used in a query: "For someone who does not know US history, the 'Battle of Blackburn's Ford' may not be interpreted as a historical event." For the first suggestion, we did not incorporate it because it rarely occurs as part of the first question of a conversation. For the latter, while we acknowledge the issue, it is outside the scope of the taxonomy and is handled by the interactions with AmbigChat.

### 3.3 Ambiguous Query Facet Taxonomy

We present our full taxonomy of ambiguous query facets in Table 1 with detailed definitions. We organize the taxonomy by the common syntactic structure of a factual question: "`<Question Adverb> <Verb> <Noun>?`" (e.g., "Who won the US Open?"). We observe that ambiguities most commonly occur in nouns and the question scope, as evident by the presence of many specific categories and the SituatedQA dataset [40]. We use this taxonomy to inform the design of AmbigChat (see Section 4) and as part of our disambiguation tree generation process (see Section 6).

## 4 Design Study

While prior work has investigated automatic techniques to resolve query ambiguity, few are designed for interactive explorations in conversational interfaces. Therefore, we conducted a design study to explore how to proactively involve users in the disambiguation process. We recruited six UX professionals (including visual and interaction designers, UX engineers, and UX researchers) from our organization via an internal study invitation. The participants, referred to as E1 to E6, have 8-19 years of experience in their respective roles ($\bar{x}$ = 11.17, SD = 5.12) and have spent the recent 2-6 years ($\bar{x}$ = 2.83, SD = 2.14) on designing conversational UI.

### 4.1 Method

We first conducted a one-hour interview with each UX professional. We began with an overview of the ambiguity problem and introduced common approaches designed for resolving query ambiguity. We asked each participant to comment on the advantages and disadvantages of each approach and collected their experiences of disambiguating factual queries. Next, we introduced our ambiguous query facet taxonomy and brainstormed the UX design of a disambiguation journey with each participant. Based on the initial round of interviews, we created a variety of prototypes. We invited the same group of UX professionals for a one-hour focus session for critiques using FigJam, a collaborative brainstorming tool. This session helped us narrow down and iterate on our designs.

### 4.2 Findings

**Support an iterative journey**. All participants acknowledged that they experienced ambiguities when interacting with an LLM and suggested that the facet taxonomy could be used for disambiguation. E2 took an iterative refinement approach to add "patches" into their original query, since they "cannot foresee and put in all the specificity at once." E6 also commented that, although they tried to be as specific as possible with their questions, it became challenging when they had limited knowledge about the topic in question. Both E2 and E6 said integrating the taxonomy could guide a more structured disambiguation process.

**Traverse a hierarchical path**. Utilizing the ambiguous facet taxonomy, E2 suggested that the system should disambiguate hierarchically with one facet at a time until reaching a precise answer, as if the user were "traveling down a path" (see Figure 2a). E6 commented that such a hierarchy should be "tucked into" the UI to avoid mental overload, while E4 added that a system should clearly reveal this notion of "branching" to the user.

**Visualize options and progress**. All participants suggested the incorporation of visual and interactive components in the form of UI widgets into the disambiguation process. Furthermore, discussions during the design reviews informed us to use the ambiguous facets taxonomy (see Table 1) to guide when to surface these widgets. Three designers (E1-E3) sketched out preliminary designs during the brainstorming sessions (see Figure 2). E2, E4, and E6 suggested that the UI should present potential answers to the clarification questions as options and allow users to add alternatives. E4 added that the disambiguation progress should be indicated to the user.

**Encourage explorations**. Three participants stated that the clarification process should support information discovery. E2 shared that this could be helpful when the user "has some knowledge but wants to learn more and expand on it." Further, E5 suggested that the system should "invite the user to interact with the information discovered" and make it easy to reference existing context to expand and gain further information. Finally, during the design

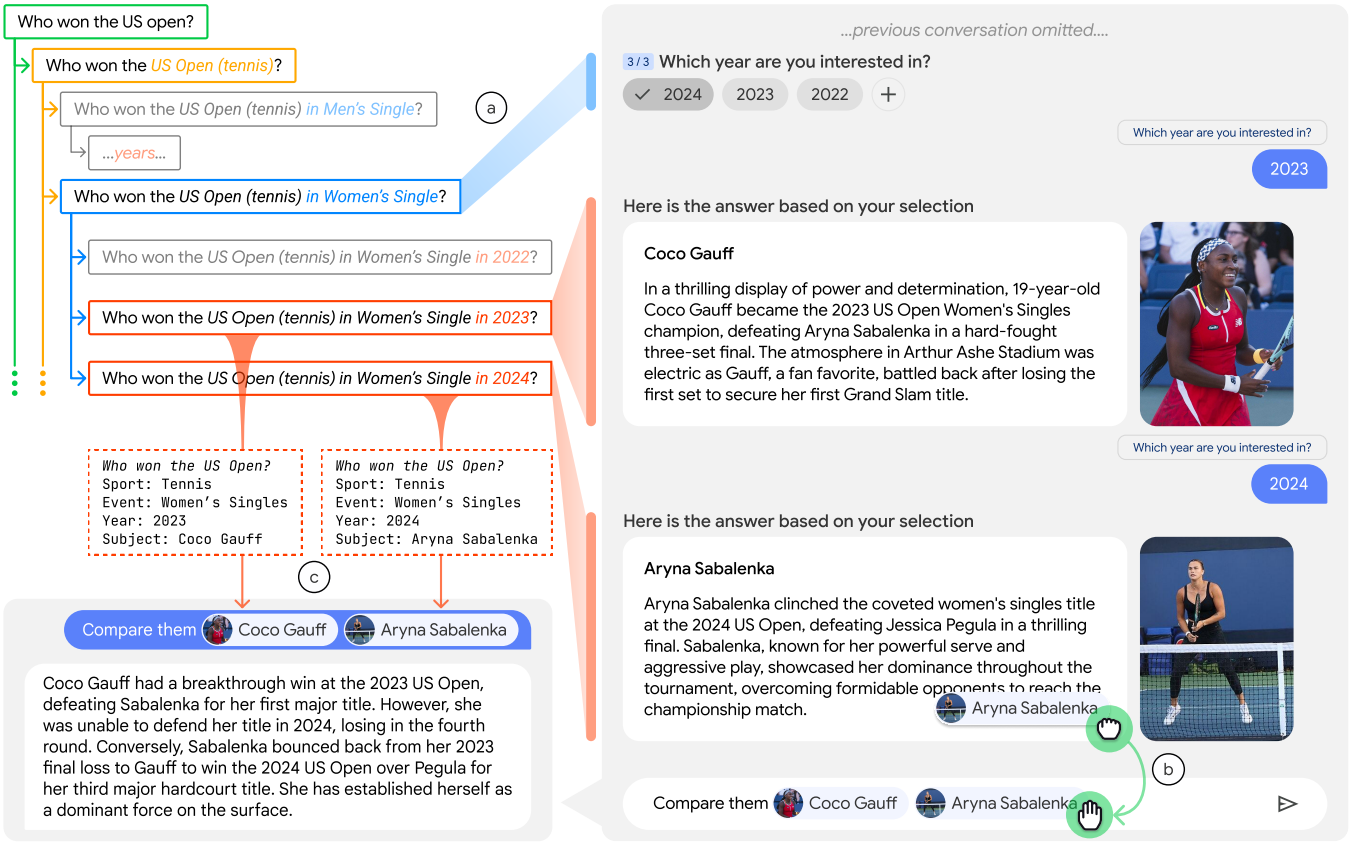


**Figure 3: Based on the disambiguation tree that AmbigChat constructs given an ambiguous user query, AmbigChat dynamically generates interactive *disambiguation widgets* for the corresponding tree node (a). When composing a text prompt, the users can drag and drop widgets to create *context chips* (b), which AmbigChat uses to fetch referenced tree nodes to build context-rich prompts for grounded model responses (c). *Image Credit: Wikimedia Commons, Coco Gauff / Aryna Sabalenka* .**

reviews, both E4 and E6 mentioned that it would be helpful to see the answer in context, such as in relationship with other related answers. Based on this feedback, we removed the toggle button design that supports a "precise" mode where only one answer is displayed (see top of Figure 2b).

## 4.3 Design Goals

Based on the findings from our design study and prior work, we formulated a set of design goals (DG) to guide the design of interactive query disambiguation techniques in a conversational UI.

**DG1: Support iterative refinement.** An interface should not overwhelm the user by asking many clarifications at once. Instead, it should iteratively present clarification questions to users in a concise format. The iterative process should be informed and guided by the ambiguous query facet taxonomy.

**DG2: Leverage suggestive and interactive components.** When asking for clarifications, the interface should provide suggestive answer options to aid disambiguation. Visual aids could help distinguish between options, such as icons and images. The UI should also support corrections and expansions of the disambiguation options.

**DG3: Encourage structured exploration and follow-up.** The system should guide users to explore alternatives and gain an in-depth understanding of the topic space of interest. Besides arriving at their intended answers, the interface should support users to review related answers and ask follow-up questions.

**DG4: Be transparent and mindful of biases.** An interface should be cautious when making assumptions or direct interpretations of the user's intent. It should be fully transparent about its decisions by providing clear explanations.

# 5 AmbigChat

AmbigChat is an automatic technique that hierarchically disambiguates open-domain questions and provides interactive guidance in a conversational UI for answer acquisition and exploration. Here we demonstrate AmbigChat's main functionalities through a user scenario. Note that users do not need to follow the steps described below as AmbigChat supports free-form conversations.

**Initial query.** Assume a user, Alice, enjoys playing sports and recently picked up tennis. During the summertime, the US Open Tennis Championship takes place. Alice wants to learn about the

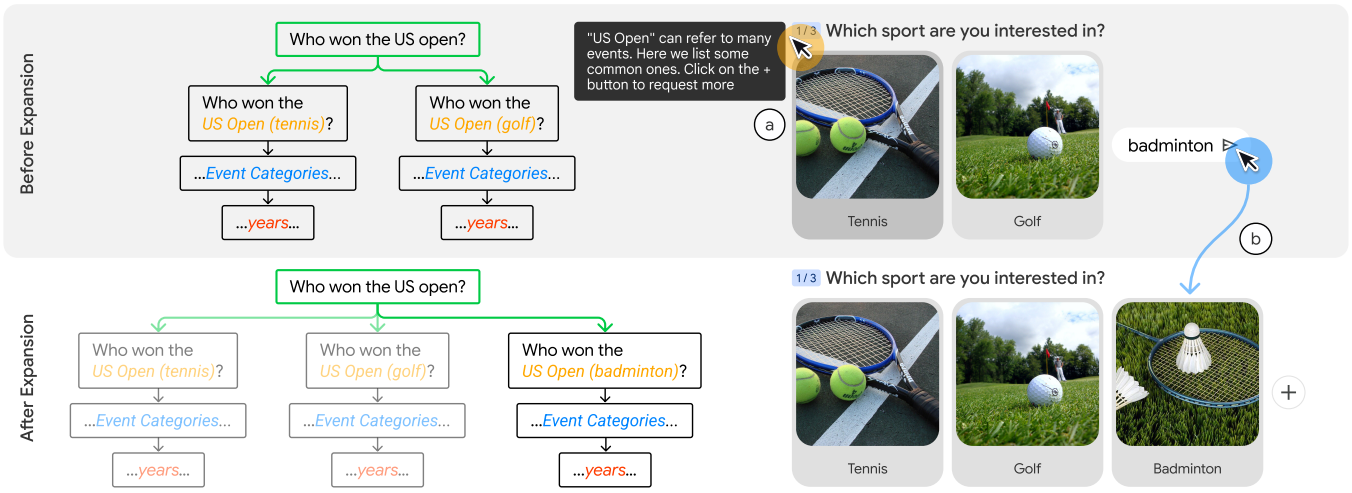


**Figure 4: In AmbigChat's UI, the *disambiguation progress indicator* is a small badge (a). Users can hover over for a tooltip that explains the clarification question. They can also explore new information via option expansion (b), where AmbigChat reruns its pipeline to expand the disambiguation tree. *Image Credit: Wikimedia Commons, Tennis / Golf / Badminton.***

winning athletes, so she inputs a general text query "*who won the us open?*" in AmbigChat's UI (see Figure 1 right). As someone new to tennis, she is aware of famous players like Roger Federer and Serena Williams, but she does not know the specifics of events such as the US Open. Upon receiving the query, AmbigChat detects three ambiguous facets from this user query: 1) the sporting event referenced by "US Open", 2) the specific event categories within "US Open," and 3) the year in which the event takes place (**DG4**). It hierarchically constructs a disambiguation tree with three corresponding levels (see Figure 1 left).

**Clarification via question widgets with tooltips.** In an iterative disambiguation process of this query (**DG1**), AmbigChat begins with the first clarification question "Which sport are you interested in?" and presents suggestive options with representative images in the form of a "question widget" (**DG2**). Alice notices the 1/3 badge, learning that this is one of the three clarification questions. She hovers over that badge and sees a tooltip explaining that "US Open" is a name shared between multiple sporting events (see Figure 4a). She also learns that tennis and golf are the two most common interpretations of the US Open. Alice observes a *plus button* after the two options that allows her to explore other US Open sporting events. Continuing with the information seeking journey, Alice clicks on "Tennis" and sees the second clarification question, "Which event are you interested in?" Having single players in mind, Alice is surprised to see "Mixed Doubles," an event type she was not previously aware of. She hovers over the option and sees a tooltip stating "Mixed doubles is played between two teams of two players, with one male and one female on each side."

**Answer display with answer widgets.** Alice continues with her event of interest "Women's Singles." AmbigChat surfaces the last clarification question asking about the year. Alice chooses 2023 and sees the "answer widget" showing "Coco Gauff" with an image of her playing and a paragraph of descriptions. As the question widget for year clarification remains in the chat, Alice clicks on other years for exploration, seeing more results such as the 2024 winner Aryna Sabalenka and 2022 winner Iga Świątek (see Figure 3).

**Follow-up conversations with context chips.** As Alice is curious about how these players compare with each other, she drags and drops the answer widgets of Coco Gauff and Aryna Sabalenka into the prompt input field (**DG3**) and asks "*Compare them* Coco Gauff Aryna Sabalenka " (see Figure 3b). AmbigChat parses the selected chips to include context parsed from the chips' corresponding tree nodes, including their disambiguated facet values and these players' names (see Figure 3c). It returns a text paragraph comparing their performances in the context of the US Open Women's Singles events, grounded by web search results.

**Disambiguation tree expansion for exploration.** Finally, Alice wonders if there is a US Open event for other racket sports, such as badminton. She scrolls back to the first clarification question and clicks on the plus button with a keyword "Badminton." AmbigChat performs a web search and expands a new subtree with information about the US Open Badminton Championships (see Figure 4). Since this expansion action already serves as the answer to the first clarification question, AmbigChat directly surfaces the second-level question on event categories to allow Alice to continue her exploration of the US Open Badminton Championships (**DG3**).

## 6 Method

We first describe the algorithmic pipeline of AmbigChat that constructs a *disambiguation tree* from a given query. We then describe the implementation of AmbigChat's disambiguation widgets.

### 6.1 Disambiguation Tree Construction Pipeline

We develop a pipeline to build a disambiguation tree with an LLM in three stages: 1) facet generation, 2) tree building, and 3) answer

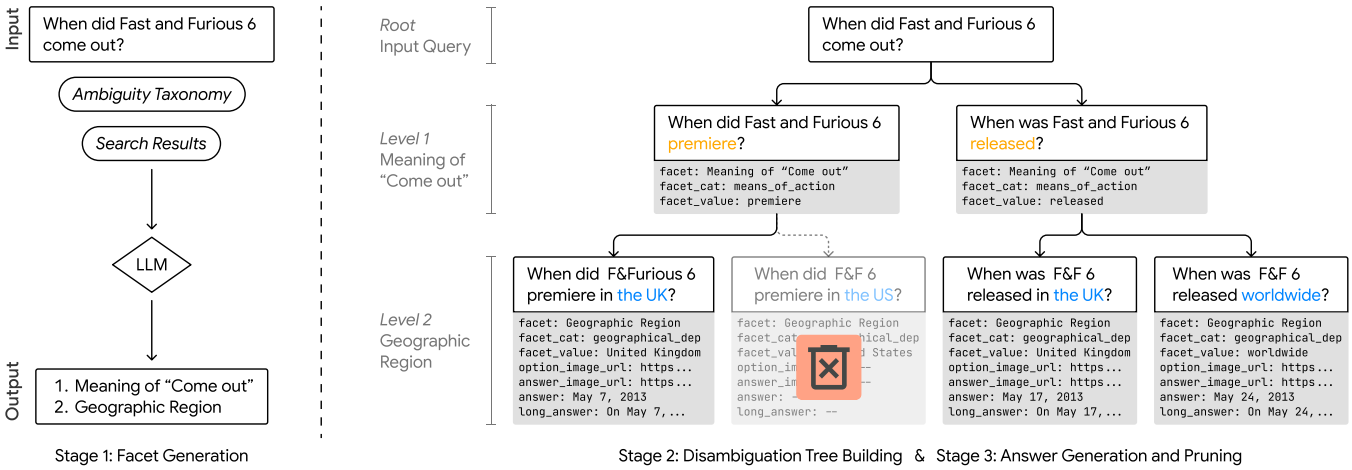


**Figure 5: AmbigChat's disambiguation tree generation pipeline consists of three stages. Stage 1 generates an ordered list of ambiguous facets from the input query. Stage 2 constructs the disambiguation tree whose levels correspond to the list of facets. After answers are generated for leaf nodes, we prune the tree to remove all unanswerable nodes in Stage 3. Both Stage 2 and 3 use web search results for grounding.**

generation and pruning (see Figure 5). We implement the pipeline in Python and communicate with Gemini 1.5 Pro via the Gemini API [10]. For illustration purposes, we use the query "*When did Fast and Furious 6 come out?*" with two ambiguous facets.

*6.1.1 Stage 1: Facet Generation.* Given an input query $q$, we prompt an LLM to generate a list of ambiguous facets $f_i$ labeled with its ambiguity type $t_i$ as $\mathcal{L}_{\text{fct}} = [(f_1, t_1), (f_2, t_2), ..., (f_n, t_n)]$, where $n$ is the number of facets. If no ambiguous facets can be detected from $q$, we produce an empty list $\mathcal{L}_{\text{fct}} = []$. Following the RAG approach [14], we supply the LLM with top 10 web pages retrieved by querying Google with $q$ and our ambiguity taxonomy (see Table 1). By performing an online search, we alleviate hallucination with grounded data for factually correct results. In addition, we ask the model to order the facets based on the information coverage in the search results [41] and guidance from our taxonomy. More specifically, we guide the model to follow the top-to-bottom sequence of the facet types listed in Table 1. We prioritize the ambiguities in the question stem (nouns, verbs, and the question adverb), with the question scopes, such as missing temporal and geographical information at the end. Moreover, ambiguous facets in the categories of *Part of Entity Reference* and *Relationships between Entities* must follow after *Entity Reference*, as the first two depend on the last one. For the query "*When did Fast and Furious 6 come out?*", our pipeline at this stage outputs `[(Meaning of "Come out", means_of_action), (Geographic Region, geographical_dep)]` (Stage 1 in Figure 5).

*6.1.2 Stage 2: Hierarchical Tree Building.* Based on the ordered facet list $\mathcal{L}_{\text{fct}}$, we construct the disambiguation tree in a breadth-first manner. The tree takes the user input query $q$ as the root node. Each level below the root level corresponds to an ambiguous facet $f_i \in \mathcal{L}_{\text{fct}}$ in the same order (see Figure 5). To populate the nodes at each tree level, we prompt the LLM to generate values for the given facet $f_i$ conditioned on the query stored in the parent node and obtain $\mathcal{V}_{i,p} = \{v_{i,p,1}, v_{i,p,2}, ..., v_{i,p,m}\}$, where $p$ is the parent node and $m$ is the number of facet values augmented by top Google Search results. For example, the first facet $f_1$ is `Meaning of "Come out"` and its parent node is the root node where $p = 1$. The generated facet values are $v_{1,1,1}$ = `premiere` and $v_{1,1,2}$ = `release` (see "Level 1" of the tree in Figure 5). Moreover, for each facet value $v_{i,p,m}$, we instruct the LLM to generate a rewrite of the input query $q$ as $q'_{i,p,m}$. For example, AmbigChat rewrites the root query $q$ ="*When did Fast and Furious 6 come out*" to $q'_{1,1,1}$ ="*When did Fast and Furious 6 premiere?*" for the facet value $v_{1,1,1}$ = `premiere`.

After obtaining this set of information, AmbigChat adds a new child node with the facet value $v_{i,p,m}$ and the query rewrite $q'_{i,p,m}$ to the parent node $p$. The facet $f_i$ and its type label $t_i$ are also stored in the node (see Figure 5). In order to surface question widgets with tooltips and visual aids, we add to the node a short description of the facet value $v_{i,p,m}$, an explanation of why the facet $f_i$ is ambiguous, and an image URL fetched via Google Search with $q'_{i,p,m}$. We repeat this process for every facet $f_i$ with respect to every node in the $i-1$ level of the tree as the parent node. Note that this hierarchical decoding is critical, as the facet values of deeper levels depend on those of earlier facet levels (e.g., the geographic regions depend on whether it is asking for the movie's premiere date or release date).

*6.1.3 Stage 3: Answer Generation and Pruning.* After we populate all levels with tree nodes, we instruct the LLM to generate an answer for each fully disambiguated queries stored in the leaf nodes of the tree. We again ground this generation process with Google Search results for accuracy. We generate both a short answer and a long-form answer with a few paragraphs, and additionally fetch an image with the short answer as the search query. It is possible that a certain facet value $v_{i,p,m}$ is incorrectly generated. For example, $v_{2,1,2}$ = `United States` is an incorrect value for the parent query $q'_{1,1,1}$ = "*When did Fast and Furious 6 premiere?*", as the movie only premiered in the UK. For this leaf node, the LLM would return `No answer` (see Figure 5 right). For each of these unanswerable leaf

nodes, we iteratively prune the tree until we remove this node and all of its parent nodes that do not have other children.

## 6.2 Interactive Disambiguation Widgets

We built our front-end conversational interface on Project Bespoke [19] with the Flutter framework [6]. To respond to the user input during runtime, we use the Remote Flutter Widgets (RFW) to generate two types of dynamic widgets for questions and answers. RFW renders widgets based on declarative UI descriptions generated by our back-end server.

*6.2.1 Question Widget.* In our UI, the question widget serves as the main driver of disambiguation (see Figure 1). We surface the question widget whenever the user reaches a non-leaf node in the disambiguation tree. A question widget consists of a clarification header followed by a list of option chips (see Figure 3). The clarification header is a templated question with either `"Which <facet> are you interested in?"` or `"By <facet> do you mean..."` depending on the facet type. The header is preceded by a progress indicator badge, and its value is `current_node_depth / tree_depth` (see Figure 4a). The option chips are the facet values of the current node's children. We utilize the ambiguous facet taxonomy to decide whether to use chips with or without visual aid. If the facet in the constructed disambiguation tree is of type "Entity Reference", "Part of Entity Reference", "Relationships Between Entities", or "Geographical Dependency", we present images or graphical icons alongside text. Figure 3a shows an example of a question widget with plain text options, while Figure 4 displays a widget with visual options. Hovering over either the progress badge or an option chip displays a tooltip of explanation (see Figure 4a).

When the user selects one of the option chips, AmbigChat displays the selection in the chat as a typed message quoting the question widget (see Figure 1). We then traverse to the corresponding child node and surface either a question widget again or an answer widget if a leaf node is reached. Trailing the list of option chips is a "+" button. The user can click on it and enter a custom facet value not already in the list (see Figure 4b). In the back end, we run the stage 2 and 3 of the disambiguation tree pipeline to build a new subtree attached to the current node.

*6.2.2 Answer Widget.* The answer widget signifies that the user has reached the end of a disambiguation path at a leaf node (see Figure 1). It consists of a pre-defined title "Here is the answer based on your selection," an answer card, and an accompanying image that displays `answer_image_url` (see Figure 5). The title of the answer card is the short answer stored in the leaf node, and the body text of the answer is the value of `long_answer`.

*6.2.3 Context Chips.* We designed the context chips to facilitate contextual reference for follow-up conversations, inspired by the direct manipulation principles employed by DirectGPT [16]. Users can drag and drop any option chip in a question or answer widget. Our UI turns a dragged element to a chip with a circle avatar that serves as a minimal representation of the source widget (see Figure 3b). When the UI sends a user's input prompt with context chips to the back end, we locate the referenced nodes in the tree and include disambiguation information as context when prompting the LLM to generate a response as shown in Figure 3c.

While we also supply the LLM with the entire conversation history as context, certain information might be diluted and missed by the model due to reasons such as appearing too early in a conversation or having a terse user prompt. For example, the user might want to know whether the first US Open Tennis Championship happened earlier than the Golf Championship. For a short prompt such as "*which one is earlier tennis or golf,*" even with the previous conversation context, the model is likely to return an answer about the origin of these two sports. With context chips, we turn the query into "*which one is earlier* tennis *or* golf " supplied with the context that the user is disambiguating the US Open event. The model would therefore provide a more accurate response, such as "*The U.S. Open Tennis tournament is older than the U.S. Open Golf tournament, as U.S. Open Tennis started in 1881 while the U.S. Open Golf in 1895.*" Broadly, users may use individual context chips for information grounding or "filtering," such as "*when was the first* golf ," to obtain context-aware answers.

# 7 Results

To demonstrate the effectiveness of AmbigChat, we applied our automatic approach to a wide variety of ambiguous queries and summarize our observations. We further test the efficacy of the disambiguation ability of AmbigChat in a technical evaluation.

## 7.1 Preliminary Findings

Figure 1, 6, and 7 show example conversations enabled by AmbigChat with queries from the AmbigNQ dataset [17]. These examples cover a wide range of topics from college ranking ("*Where does Arizona State University rank nationally 2024?*" in Figure 7), films ("*When did Fast and Furious 6 come out?*" in Figure 5 and 6), to sports ("*Who won the us open?*" in Figure 1, 3, and 4). These queries contain eight different types of ambiguous facets across all four major categories in our ambiguity taxonomy.

Figure 7 shows a conversation about the national ranking of a state university. AmbigChat detects only one ambiguous facet of the information source and provides a long list of six ranking source options. This shows how the disambiguation tree could grow in breadth. In contrast, in Figure 5 and 6, the disambiguation tree for the query "*When did Fast and Furious 6 come out?*" has a relatively small number of options at each level. Specifically, the node that disambiguates the verb to "premiere" only has one child node, as the movie only premiered in one location. These two examples demonstrate how the breadth and depth of a tree can vary widely depending on the phrasing and topic.

We present additional example conversations in the Appendix. Figure A.1 shows a conversation about *The Shining* ("*Where was the hotel in The Shining located?*"). The model's answer reveals both the "Overlook Hotel" and the "Stanley Hotel". Although listed as two different options, they are highly interrelated as the latter is a real hotel that serves as the inspiration of the former. This shows an inherent limitation of disambiguation, as it might be counterintuitive to differentiate highly related entities. On the other hand, the conversation about the question "*Who won the us open?*" can take advantage of the highly structured nature of sporting events. This makes it easy for users to expand the conversation by comparing winners across years and categories (Figure 3).

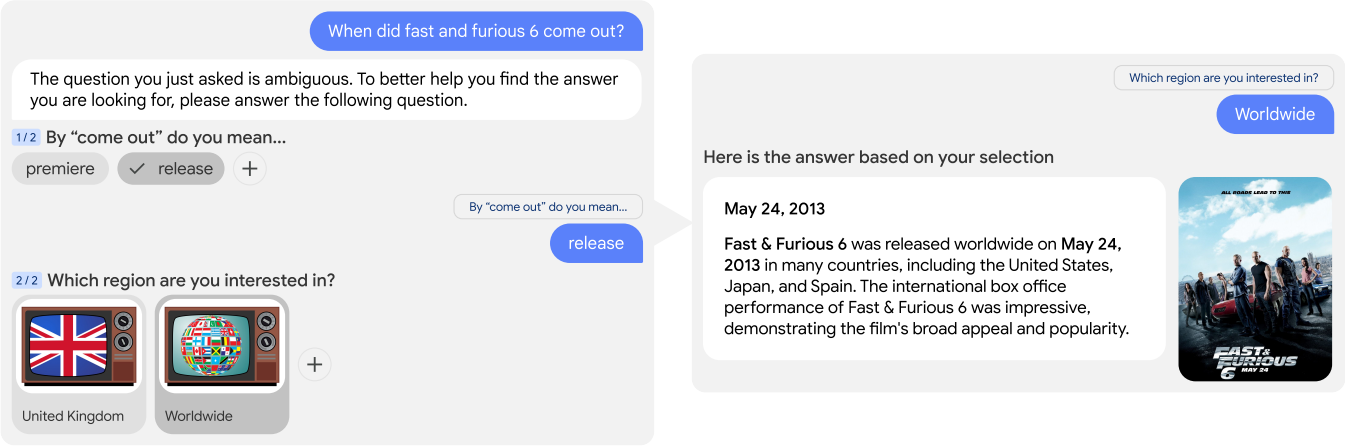


**Figure 6: The input query, "*When did fast and furious 6 come out?*", has two ambiguous facets: 1) meaning of "come out" and 2) the geographic region. The first facet has the ambiguous type "Means of an Action" and the second is "Geographical Dependency." See Figure 5 for the disambiguation tree of this example. *Image Credit: Wikimedia Commons, UK TV Icon / World TV Icon / Fast and Furious 6.***

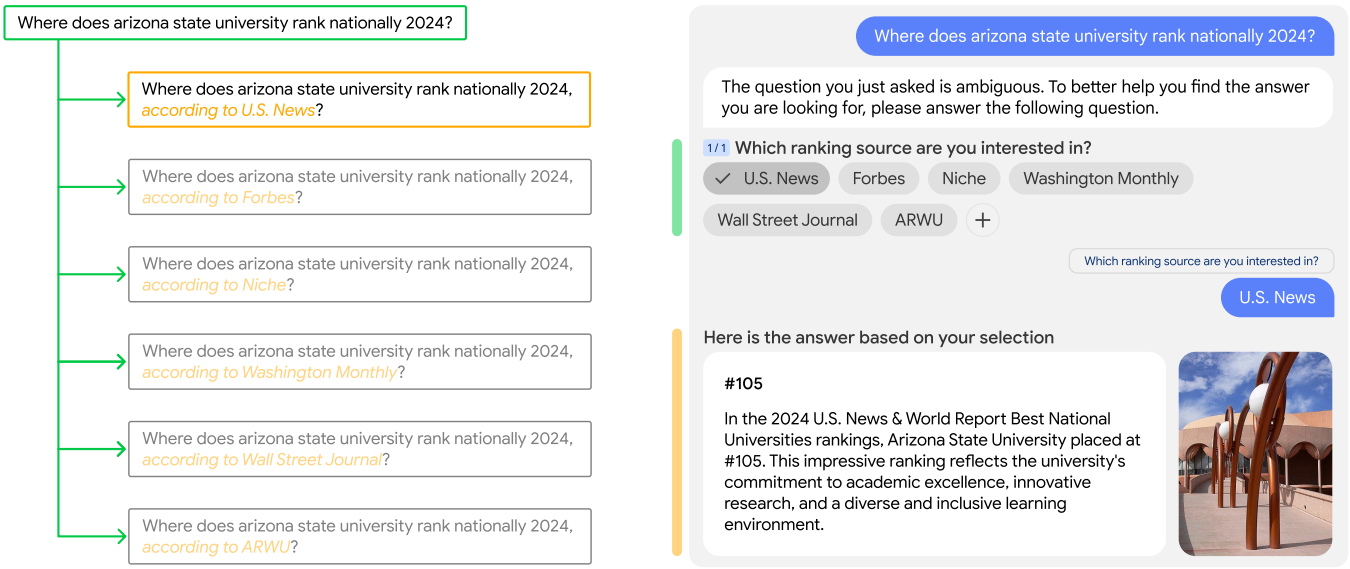


**Figure 7: "*Where does arizona state university rank nationally 2024?*" is a relatively well-specified query with just one ambiguous facet of type "Information Source Dependency." However, the disambiguation tree of this query, while small in depth, is wide with six different options for the ranking source. *Image Credit: Wikimedia Commons, ASU Gammage Auditorium.***

Figure A.2 starts with an underspecified question on art history ("*When was David created?*") with four facets. Based on our taxonomy, AmbigChat organized this complex space by the meaning of "created", life stage of David, specific artwork, and the meaning of "when." While this is one reasonable breakdown of the query, there are other ways to organize this information space, such as by artists, material (e.g., oil painting, marble, or bronze), and the nature of patronage (i.e., how the artwork was commissioned).

## 7.2 Technical Evaluation

To evaluate the effectiveness of our disambiguation tree building pipeline for generating disambiguated questions and their answers, we followed the experiment setup and adopted the metrics described in AmbigQA [17]. We compute the F1 scores and compare the performance of AmbigChat against other models. We use the *dev* set of AmbigNQ [17], as it contains both annotator-generated disambiguated queries and their corresponding answers. We did not use the *test* set as its answers are not publicly available.

**Table 2: We performed AmbigChat's automatic pipeline on ambiguous queries in the *dev* set of AmbigNQ [17] and computed F1 scores for comparison. The *multi* measure only includes queries with multiple question-answer pairs. AmbigChat achieves similar or better scores across all F1 variations compared to other methods. Note: * denotes that only one F1 score is reported.**

| Methods | $F1_{ans}(all)$ | $F1_{ans}(multi)$ | $F1_{BLEU}$ | $F1_{EDIT-F1}$ |
|---|---|---|---|---|
| SpanSeqGen 2020 [17] | 42.3 | 31.7 | 14.3 | 8.0 |
| AmbigPrompt 2023 [33] | 48.7 | 38.8 | - | - |
| Shi et al. 2024 [28] | 53.47* | | - | - |
| Baseline (Few-shot RAG) | 43.9 | 48.8 | 33.7 | 17.0 |
| **AmbigChat** | **60.2** | **55.8** | **45.3** | **31.7** |

Given an input query $q$, the experiment asks a model to generate a list of pairs of a disambiguated question $q'_i$ and its corresponding answer $a'_i$ in the form of $\mathcal{L}_{dqa} = [(q'_1, a'_1), (q'_2, a'_2), ..., (q'_n, a'_n)]$. For AmbigChat, we produced this list by taking all the queries and answers stored in the leaf nodes of the disambiguation tree. The F1 score computes the aggregated precision and recall between $\mathcal{L}_{dqa}$ and the ground truth $\mathcal{G}_{dqa} = [(\bar{q}'_1, \bar{a}'_1), (\bar{q}'_2, \bar{a}'_2), ..., (\bar{q}'_n, \bar{a}'_n)]$. $F1_{ans}$ is the F1 score on answers only. $F1_{BLEU}$ uses BLEU [21] to account for string similarity between $q'_i$ and $\bar{q}'_i$, while $F1_{EDIT-F1}$ uses unigram editing distance instead. We refer the reader to the AmbigQA [17] paper for more details.

We selected four existing methods to compare against AmbigChat (Table 2). SpanSeqGen is the original method proposed alongside AmbigNQ by Min et al. [17] and does not use an LLM. Both AmbigPrompt [33] and the method by Shi et al. [28] are recent works utilizing LLMs. To better place the scores of AmbigChat into perspective for a fair comparison, we implemented a baseline technique in a few-shot manner with RAG. Similar to AmbigChat, the baseline method uses Gemini 1.5 Pro. We supply it with a few examples of disambiguated question and answer pairs taken from the *train* set of AmbigNQ. The baseline method similarly queries Google Search to ground its generation.

We present the results in Table 2. Overall, we found that AmbigChat achieved better performances than the state-of-the-art techniques. Our baseline method was robust enough to outperform the non-LLM-based SpanSeqGen method [17]. It performed better than AmbigPrompt [33] when only considering ambiguous queries with more than one annotated question-answer pair. Only the method by Shi et al. [28] achieved a higher score. By outperforming the baseline that uses the same LLM model, AmbigChat is effective at generating disambiguated questions and answers.

## 8 User Evaluation

To understand how AmbigChat could help users to resolve ambiguous queries, we conducted a within-subjects study with 16 participants. We asked participants to disambiguate two factual queries in two conditions: AmbigChat and a text-only LLM-based baseline. We aim to answer the following research questions:

**RQ1** Would AmbigChat help users easily and efficiently disambiguate open-domain questions to find desired answers?

**RQ2** Would AmbigChat support and encourage structured explorations of topics in an ambiguous query?

**RQ3** In what ways would AmbigChat affect the experience of an information seeking journey?

### 8.1 Participants

We recruited 16 participants (9 females, 6 males, 1 chose not to disclose; age 25 to 54) through a mailing list of over 10,000 recipients in our organization. They have a diverse range of professions, including engineers, designers, analysts, and financial specialists. In terms of frequency of using conversational interfaces (e.g., Gemini [34], ChatGPT [20], and Perplexity [1]), 9 participants reported more than once a week in the past three months; 3 reported once a week; 3 reported once or twice a month; 1 had no prior experience. All participants were compensated for study completion.

### 8.2 Study Design

Informed by prior study designs [7, 35], our study consists of five sub-tasks (**ST1**—**ST5**) in two categories. The first type is targeted disambiguation sub-tasks with increasing difficulty in the style of "Jeopardy evaluation" [7] (**ST1**—**ST3**). The second is open exploration sub-tasks (**ST4**—**ST5**) designed to simulate real-life information seeking experiences. Table 3 lists the specific requirements of each sub-task. The set of five sub-tasks was performed over two ambiguous queries we composed from the AmbigNQ [17] dataset: **(AQ1)** "*Who won the US Open?*" and **(AQ2)** "*When was David created?*" (see Figure 1 and A.2). We chose these two queries for their relatively high number of ambiguity levels (three for **AQ1** and four for **AQ2**). In addition, their ambiguities spanned a wide range of types, including entity reference, instance reference, temporal, answer type, and means of action. **AQ1** represents a class of queries about popular timely events, while **AQ2** resembles queries that seek historical information in a specific domain.

A challenge of QA task design is "parroting" [7]; if we give a participant an ambiguous question ("*Who won the us open?*") with its disambiguated rewrite ("*Who won the us open tennis men's single in 2021*") and ask them to find an answer to that question, they could use the given disambiguated query to directly retrieve the answer. To address this, we adopted the "Jeopardy evaluation" methodology proposed by Gao et al. [7] and adopted by subsequent work [5, 15, 29–31]. In **ST1**—**ST3**, we asked the participants to identify how the initial query could be fully disambiguated to produce the target answer. For example, for **AQ1** "*Who won the US Open?*", we provided the target answer "*Novak Djokovic*" and asked participants to identify the path to disambiguate the query (see Table 3). In this case, one needed to identify that US Open refers to the tennis event and that Novak Djokovic won the Men's Singles category in years either 2011, 2015, 2018, or 2023. Note that there may exist multiple ways to disambiguate a query for the target answer, and we considered all paths correct. We discouraged but allowed participants to reverse-engineer the answer to complete the task, such as asking "*Who is Novak Djokovic.*"

We designed **ST1** to **ST3** with an increase of difficulty. The answers of **ST1** are one of the most common answers to their respective query (i.e., the most recent winner of US Open Tennis Championship's Men's Singles event; the years in which Michelangelo's *David* was created). In this way, participants could find that answer either by selecting all the first options in the AmbigChat

**Table 3: Our user evaluation consists of two types of sub-tasks: three *jeopardy-style* tasks asking users to identify how the input question should be disambiguated to produce target answers, and two *open exploration* tasks leveraging various functionalities of AmbigChat. These tasks are performed over two different questions AQ1 and AQ2.**

| Task Design | | Jeopardy-style Sub-tasks | | | Open Exploration Sub-tasks | |
|---|---|---|---|---|---|---|
| | | ST1 | ST2 | ST3 | ST4 | ST5 |
| Task 1 | AQ1 **Who won the US Open?** | Novak Djokovic | Allisen Corpuz | Yushi Tanaka | *Find three more answers that are different from the three answers identified in the previous sub-tasks* | *Ask at least one follow-up question that makes direct references to at least two of the answers you have identified* |
| Task 2 | AQ2 **When was *David* created?** | Between 1501 and 1504 | Baroque Period | Between 1610 and 1624 | | |

condition, or by entering the query into the chat in the baseline condition. The answers of **ST2** are less popular interpretations of the queries, but they are included in the constructed disambiguation trees. The answers in **ST3** are the least common interpretations and do not exist in the prebuilt trees. To find these answers, one possible way is to expand the "sport" level of the tree for **AQ1** with "badminton" and the "specific artworks" level under sculptures of young David for **AQ2** with "Bernini."

## 8.3 Procedure

Our study had two conditions: a baseline condition and AmbigChat. We implemented a text-only chat interface as the baseline that had accurate logging with timestamps. We used the same baseline method (few-shot with RAG) as described in Section 7.2 to ensure that the model was adequate at answering ambiguous questions. We did not incorporate the ambiguity taxonomy in the baseline.

Each study session was 50 minutes. In a five-minute warm-up prior to the tasks, the experimenter gave a tutorial of the condition with the sample query "*When did Fast & Furious 6 come out?*" (see Figure 6). We helped participants familiarize themselves with the tool through this query. Then, each participant completed the five sub-tasks in 15 minutes. After each task, we verbally asked the participant to answer seven 7-point Likert-scale questions based on their task completion experience. Finally, after the participant completed the two tasks in both conditions, we interviewed them for their subjective feedback. We conducted all study sessions remotely. We counterbalanced the ordering of both the tool conditions and the tasks. For each unique combination of a tool and a task (four total), we collected eight participant data points.

## 8.4 Quantitative Results

We recorded every conversation in both conditions with timestamps and interaction types, including text prompts and widget interactions, for quantitative measurements. We used the paired t-test to compute statistical significance. The measurements support a positive answer to **RQ1** on efficiency, including the significantly shorter task completion time and decreased number of textual interactions. The significantly higher Likert-scale ratings affirms both **RQ1** and **RQ2** on exploration (see Figure 8).

*8.4.1 Time to completion.* All participants completed all tasks in both conditions. We report the time to completion of the jeopardy-style sub-tasks (**ST1**—**ST3**). We discounted the model response time as it fluctuated depending on the model server status. Each participant spent an average of *4 minutes and 19 seconds* to complete **ST1** to **ST3** with AmbigChat ($\bar{x}$ = 258.55, SD = 79.53). They spent *6 minutes and 9 seconds* ($\bar{x}$ = 369.39, SD = 116.17) in the baseline condition. The time to completion of AmbigChat is significantly faster than that of the baseline ($p = .01, t = 2.85$). With AmbigChat, each participant took an average of 3 minutes and 36 seconds on Task 1 and 5 minutes and 1 second on Task 2, while they spent 6 minutes and 17 seconds on Task 1 and 6 minutes and 1 second on Task 2 on average with the baseline.

*8.4.2 Textual and Widget Interactions.* In the baseline, the participants used an average of 14.19 textual prompts (SD = 4.60) to complete a task ($\bar{x}_{\text{task}_1}$ = 15.13, $\text{SD}_{\text{task}_1}$ = 2.80, $\bar{x}_{\text{task}_2}$ = 13.25, $\text{SD}_{\text{task}_2}$ = 5.72). This number decreased significantly ($p < .001$, $t = 9.03$) to $\bar{x} = 5.0$ for AmbigChat (SD = 2.35). We observed that interactions with the disambiguation widgets, which the participants used an average of 23.38 times (SD = 3.66), replaced manual text prompts and resulted in the decrease. Specifically, each participant used 4.88 textual prompts (SD = 2.20) and 22.63 widget interactions (SD = 3.87) in Task 1, and 5.13 textual prompts (SD = 2.47) and 24.13 widget interactions (SD = 3.26) in Task 2.

In the AmbigChat condition, each participant explored an average of 9.44 answers (SD = 2.29), which is more than the required 6 answers to complete each task ($\bar{x}_{\text{task}_1}$ = 10.88, $\text{SD}_{\text{task}_1}$ = 1.96, $\bar{x}_{\text{task}_2}$ = 8.00, $\text{SD}_{\text{task}_2}$ = 1.58). Each participant placed an average of 2.94 context chips (SD = 1.48) in their prompts ($\bar{x}_{\text{task}_1}$ = 2.63, $\text{SD}_{\text{task}_1}$ = 0.70, $\bar{x}_{\text{task}_2}$ = 3.25, $\text{SD}_{\text{task}_2}$ = 1.92). Although we designed the tasks to require the participants to expand the disambiguation tree only once, we found that each participant used this feature 1.94 times (SD = 1.34) to explore different answers ($\bar{x}_{\text{task}_1}$ = 2.0, $\text{SD}_{\text{task}_1}$ = 1.41, $\bar{x}_{\text{task}_2}$ = 1.88, $\text{SD}_{\text{task}_2}$ = 1.27).

*8.4.3 Subjective Ratings.* In the post-task ratings, we asked the participants to give subjective ratings regarding the information seeking journey of both conditions. Figure 8 shows the Likert-scale questions and the aggregated results. We used the Wilcoxon signed-rank test to compute significant differences between the ratings. We found that AmbigChat received significantly higher ratings for all six statements. Specifically, for **Q1**, the participants rated AmbigChat with $\bar{x}_{\text{ours}} = 5.94$, which is significantly higher ($p < .001, z = 1.5$) than the baseline ($\bar{x}_{\text{base}} = 3.0$). Rating for **Q2** for our tool $\bar{x}_{\text{ours}} = 5.81$ is also significantly higher ($p < .001$, $z = 4.5$) than the baseline ($\bar{x}_{\text{base}} = 2.94$). It follows similarly for the remaining four ratings: **Q3**: $\bar{x}_{\text{ours}} = 6.06$, $\bar{x}_{\text{base}} = 3.19$, $p < .001$, $z = 0.0$; **Q4**: $\bar{x}_{\text{ours}} = 6.13$, $\bar{x}_{\text{base}} = 3.06$, $p < .001$, $z = 0.0$; **Q5**: $\bar{x}_{\text{ours}} = 6.19$, $\bar{x}_{\text{base}} = 3.06$, $p < .001$, $z = 0.0$; **Q6**: $\bar{x}_{\text{ours}} = 5.9$, $\bar{x}_{\text{base}} = 3.38$, $p < .001$, $z = 0.0$.

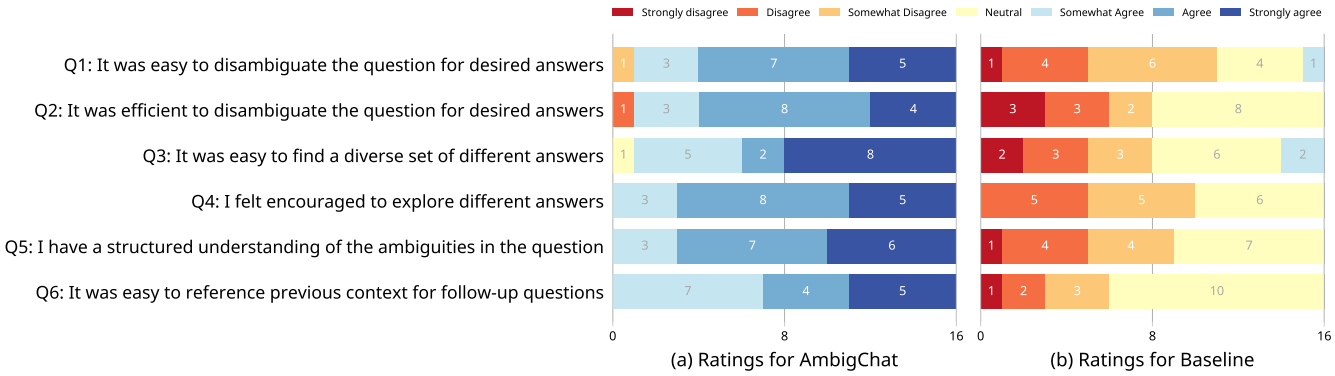


**Figure 8: The ratings of each question are significantly higher for AmbigChat than the baseline condition. Note that the baseline receives mostly neutral or close-to-neutral ratings, as many participants verbally acknowledged that they were quite used to the disambiguation behavior of the existing LLM chatbots.**

## 8.5 Qualitative Findings

We summarize our qualitative observations of participant behaviors and feedback from the surveys and semi-structured interviews to discuss insights for **RQ3**.

*8.5.1 Alleviated Mental Burden.* 14 participants mentioned that the question widgets reduced their mental effort for disambiguation. In the baseline condition, P13 commented that it often frustrated them because they did not know the available options in response to a clarification question. When recounting their experience with the US Open query, P10 stated, "I didn't know that there are five different event categories [in US Open Tennis] so I had no idea how to respond." P6 summarized the baseline disambiguation experience as "shooting a shot in the dark".

With AmbigChat, both P4 and P6 shared that, because of the presence of the suggested option chips, they did not have to come up with their own questions and answers from scratch. P2 noted, "the options in the UI widgets guided me to know which questions to ask." Furthermore, as evident in the subjective rating (see Figure 8), the participants liked how AmbigChat helped them arrive at precise answers more easily and efficiently: "It helps to have a visual of the hierarchical organizational structure of the questions to arrive at the precise answer" (P6).

*8.5.2 Exploratory Information Discovery.* 12 participants mentioned that interacting with AmbigChat enabled them to explore the information space spanned by the original ambiguous query, evident in the significantly higher ratings of **Q3** and **Q4** (see Figure 8). In the baseline condition, P9, P14, and P16 expressed that they felt discouraged from exploring more answers to the initial query. P16, given the often limited suggestive follow-ups, stated that they "didn't know what to explore." P14 said that they felt like they were "confined to stuff [they] already knew about."

With AmbigChat, many participants explored options and answers outside of their initial intentions: "the system helped me to explore more options than what I intended before" (P5). Regarding how AmbigChat had encouraged them, some participants mentioned the option chips: "By seeing the chips, you could also see some things that you didn't know before that are related to your original topic, and can explore further or go off on more tangents" (P8). P4 noted, "[The chips] motivated me to ask more questions and to find more information because I can just simply click." Furthermore, P9 thought that the UI widgets acted like "partial prompts" that provided them with hints and inspirations for exploration and follow-up. Regarding the context chip feature, P5 stated, "drag and drop makes it easier to ask follow-up questions, reducing the inertia for typing newly learned concepts." P11 summarized the experience with AmbigChat as "proactively inspiring certain needs."

*8.5.3 Structured Understanding of Ambiguity.* P8 and P10 regarded the baseline interactions as flat and linear. P1 added, "when it's just plain text with some questions, it's harder to navigate." This is echoed by the baseline's lower ratings in Q5 (see Figure 8).

With AmbigChat, 10 participants shared that they completed the tasks with a more organized understanding of where the ambiguity lies in the initial queries. Many found patterns and structures in their information seeking journeys: "I could recognize patterns of where I am being ambiguous" (P13) and "It helps with drilling down or expanding my knowledge in a structured way" (P4). P4, P7, P8, and P10 all mentioned that interacting with AmbigChat helped them build a mental map of the new information acquired. P7 and P10 stated that they imagined a branching structure: "it branches off into two different artworks. And the artwork has certain properties associated with it" (P7 on **AQ2**). When we asked participants to describe the structure, both P1 and P10 used the analogy of an encyclopedia: "It feels like having an encyclopedia somewhere with chapters and you can navigate the content" (P1). For P8, they regarded each disambiguation widget as "anchors": "I thought the anchors were an interesting concept, and could see how that may help refine a thread as they get really long."

*8.5.4 Individual Behavior Patterns.* In the baseline condition, we observed that 13 participants took a "catch-all" prompting approach by asking queries such as "*list of us open winners*" (P1) and "*list some famous baroque david paintings*" (P15). For **ST1** and **ST2**, this

strategy often produced a list of names that included the target answers. However, for **ST3** of Task 1, P1 and P12 reverse-engineered the query by asking questions like "*Who is Yushi Tanaka*" after failing at other prompting approaches, resulting in longer time to completion and increased textual prompt interactions. Overall, 10 participants mentioned that the baseline interaction design felt familiar. P8 stated, "[AmbigChat] will probably take some getting used to as opposed to just typing everything out in free form, which I have been more accustomed to." P14 added that the disambiguation process of AmbigChat could "feel like an overkill." This might explain the predominantly neutral subjective ratings (see Figure 8).

For AmbigChat, we observed that all participants were able to use the disambiguation widgets to find the target answers. Some participants relied on prior knowledge to arrive at an answer (which we also observed in the baseline), while some others visited more answers before finding out the required ones. For **ST2**, P14 clicked on all options of Young David before arriving at the correct one, resulting in increased widget interactions. For **ST3**, since the target answer was not embedded in the disambiguation tree, we observed that all but one participant (P14) used the tree expansion function to complete this sub-task. They often first asked the LLM questions to learn the potential expansion options before expanding and traversing the tree with widgets. The participants also used this feature to find more answers for **ST4**, with expanded options like "bowling" and "chess" (P15) for Task 1 and "old david" (P16) for Task 2. Most participants used the context chips for **ST5**. P12 expanded on the usage of context chips by combining a few of them to filter and constrain the answer space, such as asking " Young David Baroque Period *sculptures by other artists*" for **ST3**.

## 9 Discussion

Based on the user evaluation results, we discuss insights for future research in open-domain question answering.

**Concretize ambiguity with multi-modal conversational interactions.** Our study demonstrates the effectiveness of scaffolding the abstract nature of language ambiguity with multi-modal turn-based interactions. The use of a hierarchical disambiguation tree, surfaced to users through a series of interactive disambiguation widgets with visual aids, makes the information space navigable and reduces the cognitive load associated with clarification dialogues. These persistent widgets in an evolving conversation can act as mental "anchors" that ground users as they explore the ambiguity space, helping them keep track of their current contexts and allowing for easy backtracking and branching into new lines of inquiry. We suggest that this approach of externalizing and concretizing ambiguity in natural language can be applied to other tasks where users need to navigate through a large space of possibilities.

**Navigate the transparency-convenience trade-off.** Guided by **DG4**, we intentionally avoid providing an answer directly without clarifications in our interaction flow. Some participants valued this design choice for making the system's "thought process" visible. However, this is less desirable for users who prefer immediate answers, especially for simpler questions with less facets. To better balance these needs, we could implement a "direct answer" mode in AmbigChat by surfacing all question widgets with a pre-selected option chips at once. This modification would offer an immediate answer, while still providing the full interactive structure. This allows users to either accept the fast result or refine the query with full control. This highlights a broader design implication for future conversational systems, as they should adapt to both user preferences and the task complexity. Instead of solely optimizing for transparency or convenience, a system should adaptively balance this trade-off, allowing the interaction to be as structured or as direct based on user preferences and interaction contexts.

**Mitigate biases in interpretation and presentation.** Our study reveals two types of system biases. The first lies in how a system initially interprets an ambiguous query and the second in how it presents clarification options. The interpretation bias was evident when users interacted with the baseline system. For example, one of the participants received an answer for the US Open Men's Singles tennis championship when they were looking for Women's Singles. AmbigChat's current design of not providing an answer without clarification is a deliberate strategy to mitigate this type of bias. The second kind of bias, the presentation bias, relates to the framing effect. In our case, this means that the display of the option chips can influence a user's choices and understanding. By surfacing the option chips in a certain order, our system may inadvertently create a perceived hierarchy of importance, leading to limited exposure to less common alternatives. While features like the option expansion and explanation tooltips are designed to mitigate this, their effectiveness relies on the user's willingness to seek other options. Therefore, a key insight for future systems is to minimize the influence of system biases on users' information seeking process; they should be not only cautious about premature interpretations of user intent, but also transparent about how they present the clarification options.

## 10 Limitations and Future Work

With overall positive participant feedback, we discuss limitations of our approach and future research opportunities. Our study design evaluates the AmbigChat system as a whole. For a more isolated analysis of the effectiveness of AmbigChat's interaction design, additional studies could consider incorporating the ambiguity taxonomy into the baseline condition. We adopted the jeopardy method [7] to create controlled experiments to quantitatively evaluate the disambiguation mechanisms of AmbigChat (see Section 8.4). However, such design is limited in its ecological validity, as the participants were aware that the input questions were ambiguous. While we included open-ended exploration sub-tasks (**ST4**—**ST5**) to better simulate real-world information seeking like prior work [5, 7, 15, 29–31], future research could evaluate the system in a more naturalistic setting where ambiguity is discovered organically by users pursuing their own information needs.

We scoped our work to factual queries as an entry point to design interactions for disambiguation. As our study participants suggested, future work could investigate approaches that handle more open-ended queries that require multi-step reasoning, such as job hunting or travel planning. Similarly, there are a wide range of ambiguity spaces that require diverse and customized answers. To further enhance user agency in these more complex scenarios, future work could support user-initiated clarifications. Instead of

guiding users down a pre-computed path, a system could dynamically reconstruct the disambiguation tree based on free-form text inputs and a user's interaction history. This would create a more personalized and collaborative partnership, allowing systems to better handle topics that go beyond a rigid taxonomy and to align more closely with users' evolving intent.

## 11 Conclusion

We presented AmbigChat, an automatic approach to disambiguate ambiguous open-domain questions in human-AI conversations. By leveraging a taxonomy of ambiguous facets derived from a corpus of 5,000 queries, AmbigChat constructs hierarchical disambiguation trees and guides users with interactive UI widgets. Our design study informed our interaction design, leading to a system that not only improves the efficiency of finding desired answers but also facilitates structured exploration of the topic space. We demonstrated AmbigChat's effectiveness through a technical evaluation and a user study with 16 participants. Our findings demonstrate the potential of AmbigChat to improve the user experiences of human-AI interactions with ambiguous questions.

## Acknowledgments

We thank all the participants in our interviews and user studies for their valuable insights. In addition, this work was made possible thanks to the support of people including, but not limited to, the following (in alphabetical order of last name): Ashish Chaudhary, Kirsten Climer, Irfan Essa, Ashwin Ganti, Stephanie Guamán, Peter McDermott, Palash Nandy, Siggi Orn, David Robishaw, David Salesin, Justin Secor, Rachel Shim, Shilp Vaishnav, Mathangi Venkatesan, and Shudi Zhang.

## References

[1] Perplexity AI. 2025. *Perplexity.* https://www.perplexity.ai/

[2] Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. 2020. ConvAI3: Generating clarifying questions for open-domain dialogue systems (ClariQ). *arXiv preprint arXiv:2009.11352* (2020).

[3] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2019. Asking Clarifying Questions in Open-Domain Information-Seeking Conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 475–484. https://doi.org/10.1145/3331184.3331265

[4] Ruijia Cheng, Titus Barik, Alan Leung, Fred Hohman, and Jeffrey Nichols. 2024. BISCUIT: Scaffolding LLM-Generated Code with Ephemeral UIs in Computational Notebooks. arXiv:2404.07387 [cs.HC] https://arxiv.org/abs/2404.07387

[5] Yingchaojie Feng, Xingbo Wang, Kam Kwai Wong, Sijia Wang, Yuhong Lu, Minfeng Zhu, Baicheng Wang, and Wei Chen. 2024. PromptMagician: Interactive Prompt Engineering for Text-to-Image Creation. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (2024), 295–305. https://doi.org/10.1109/TVCG.2023.3327168

[6] Flutter. 2025. *Flutter - Build apps for any screen.* Retrieved August, 2025 from https://flutter.dev/

[7] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G. Karahalios. 2015. DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 489–500. https://doi.org/10.1145/2807442.2807478

[8] Marti Hearst and Melanie Tory. 2019. Would You Like A Chart With That? Incorporating Visualizations into Conversational Interfaces. In *2019 IEEE Visualization Conference (VIS)*. 1–5. https://doi.org/10.1109/VISUAL.2019.8933766

[9] Google Inc. 2023. *Introducing Gemini: our largest and most capable AI model.* Retrieved August, 2025 from https://blog.google/technology/ai/google-gemini-ai/

[10] Google Inc. 2025. *Gemini models | Google AI for Developers.* Retrieved August, 2025 from https://ai.google.dev/models/gemini

[11] Kimiya Keyvan and Jimmy Xiangji Huang. 2022. How to Approach Ambiguous Queries in Conversational Search: A Survey of Techniques, Approaches, Tools, and Challenges. *ACM Comput. Surv.* 55, 6, Article 129 (dec 2022), 40 pages. https://doi.org/10.1145/3534965

[12] Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. 2023. Tree of Clarifications: Answering Ambiguous Questions with Retrieval-Augmented Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 996–1009. https://doi.org/10.18653/v1/2023.emnlp-main.63

[13] Dongryeol Lee, Segwang Kim, Minwoo Lee, Hwanhee Lee, Joonsuk Park, Sang-Woo Lee, and Kyomin Jung. 2023. Asking Clarification Questions to Handle Ambiguity in Open-Domain QA. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 11526–11544. https://doi.org/10.18653/v1/2023.findings-emnlp.772

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 9459–9474. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

[15] V. K. Chaithanya Manam, Joseph Divyan Thomas, and Alexander J. Quinn. 2022. TaskLint: Automated Detection of Ambiguities in Task Instructions. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 10, 1 (Oct. 2022), 160–172. https://doi.org/10.1609/hcomp.v10i1.21996

[16] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. 2024. DirectGPT: A Direct Manipulation Interface to Interact with Large Language Models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 975, 16 pages. https://doi.org/10.1145/3613904.3642462

[17] Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering Ambiguous Open-domain Questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 5783–5797. https://doi.org/10.18653/v1/2020.emnlp-main.466

[18] Rishab Mitra, Arpit Narechania, Alex Endert, and John Stasko. 2022. Facilitating Conversational Interaction in Natural Language Interfaces for Visualization. In *2022 IEEE Visualization and Visual Analytics (VIS)*. 6–10. https://doi.org/10.1109/VIS54862.2022.00010

[19] Palash Nandy, Sigurdur Orn Adalgeirsson, Anoop K. Sinha, Tanya Kraljic, Mike Cleron, Lei Shi, Angad Singh, Ashish Chaudhary, Ashwin Ganti, Christopher A Melancon, Shudi Zhang, David Robishaw, Horia Ciurdar, Justin Secor, Kenneth Aleksander Robertsen, Kirsten Climer, Madison Le, Mathangi Venkatesan, Peggy Chi, Peixin Li, Peter F McDermott, Rachel Shim, Selcen Onsan, Shilp Vaishnav, and Stephanie Guamán. 2024. Bespoke: Using LLM agents to generate just-in-time interfaces by reasoning about user intent. In *Companion Proceedings of the 26th International Conference on Multimodal Interaction* (San Jose, Costa Rica) *(ICMI Companion '24)*. Association for Computing Machinery, New York, NY, USA, 78–81. https://doi.org/10.1145/3686215.3688372

[20] OpenAI. 2025. . https://openai.com/chatgpt/

[21] 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*. 311–318.

[22] Brendan Park, Madeline Janecek, Naser Ezzati-Jivan, Yifeng Li, and Ali Emami. 2024. Picturing Ambiguity: A Visual Twist on the Winograd Schema Challenge. arXiv:2405.16277 [cs.CL] https://arxiv.org/abs/2405.16277

[23] Anthropic PBC. 2025. What are artifacts and how do I use them? https://support.anthropic.com/en/articles/9487310-what-are-artifacts-and-how-do-i-use-them

[24] Hossein A. Rahmani, Xi Wang, Yue Feng, Qiang Zhang, Emine Yilmaz, and Aldo Lipani. 2023. A Survey on Asking Clarification Questions Datasets in Conversational Systems. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 2698–2716. https://doi.org/10.18653/v1/2023.acl-long.152

[25] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[26] Ivan Sekulić, Mohammad Aliannejadi, and Fabio Crestani. 2021. Towards Facet-Driven Generation of Clarifying Questions for Conversational Search. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval* (Virtual Event, Canada) *(ICTIR '21)*. Association for Computing Machinery, New York, NY, USA, 167–175. https://doi.org/10.1145/3471158.3472257

[27] Vidya Setlur, Enamul Hoque, Dae Hyun Kim, and Angel X. Chang. 2020. Sneak Pique: Exploring Autocompletion as a Data Discovery Scaffold for Supporting Visual Analysis. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 966–978. https://doi.org/10.1145/3379337.3415813

[28] Yunxiao Shi, Xing Zi, Zijing Shi, Haimin Zhang, Qiang Wu, and Min Xu. 2024. Enhancing Retrieval and Managing Retrieval: A Four-Module Synergy for Improved Quality and Efficiency in RAG Systems. In *ECAI 2024*. IOS Press. https://doi.org/10.48550/arxiv.2407.10670

[29] Arjun Srinivasan, Bongshin Lee, Nathalie Henry Riche, Steven M. Drucker, and Ken Hinckley. 2020. InChorus: Designing Consistent Multimodal Interactions for Data Visualization on Tablet Devices. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376782

[30] Arjun Srinivasan and John Stasko. 2018. Orko: Facilitating Multimodal Interaction for Visual Exploration and Analysis of Networks. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 511–521. https://doi.org/10.1109/TVCG.2017.2745219

[31] Arjun Srinivasan and John Stasko. 2020. How to Ask What to Say?: Strategies for Evaluating Natural Language Interfaces for Data Visualization. *IEEE Computer Graphics and Applications* 40, 4 (2020), 96–103. https://doi.org/10.1109/MCG.2020.2986902

[32] Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. ASQA: Factoid Questions Meet Long-Form Answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 8273–8288. https://doi.org/10.18653/v1/2022.emnlp-main.566

[33] Weiwei Sun, Hengyi Cai, Hongshen Chen, Pengjie Ren, Zhumin Chen, Maarten de Rijke, and Zhaochun Ren. 2023. Answering Ambiguous Questions via Iterative Prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 7669–7683. https://doi.org/10.18653/v1/2023.acl-long.424

[34] Gemini Team. 2023. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805 [cs.CL] https://arxiv.org/abs/2312.11805

[35] Priyan Vaithilingam, Elena L. Glassman, Jeevana Priya Inala, and Chenglong Wang. 2024. DynaVis: Dynamically Synthesized UI Widgets for Visualization Editing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 985, 17 pages. https://doi.org/10.1145/3613904.3642639

[36] Priyan Vaithilingam and Philip J. Guo. 2019. Bespoke: Interactively Synthesizing Custom GUIs from Command-Line Applications By Demonstration. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 563–576. https://doi.org/10.1145/3332165.3347944

[37] Yael Vinker, Andrey Voynov, Daniel Cohen-Or, and Ariel Shamir. 2023. Concept Decomposition for Visual Exploration and Inspiration. *ACM Trans. Graph.* 42, 6, Article 241 (dec 2023), 13 pages. https://doi.org/10.1145/3618315

[38] Hamed Zamani and Nick Craswell. 2020. Macaw: An Extensible Conversational Information Seeking Platform. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 2193–2196. https://doi.org/10.1145/3397271.3401415

[39] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 437, 21 pages. https://doi.org/10.1145/3544548.3581388

[40] Michael Zhang and Eunsol Choi. 2021. SituatedQA: Incorporating Extra-Linguistic Contexts into QA. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 7371–7387. https://doi.org/10.18653/v1/2021.emnlp-main.586

[41] Ziliang Zhao and Zhicheng Dou. 2024. Generating Multi-turn Clarification for Web Information Seeking. In *Proceedings of the ACM on Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 1539–1548. https://doi.org/10.1145/3589334.3645712

[42] Qiyu Zhi and Ronald Metoyer. 2020. GameBot: A Visualization-augmented Chatbot for Sports Game. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3334480.3382794

# A Appendix

Here we present two additional example conversations enabled by AmbigChat from questions with multiple ambiguous facets.

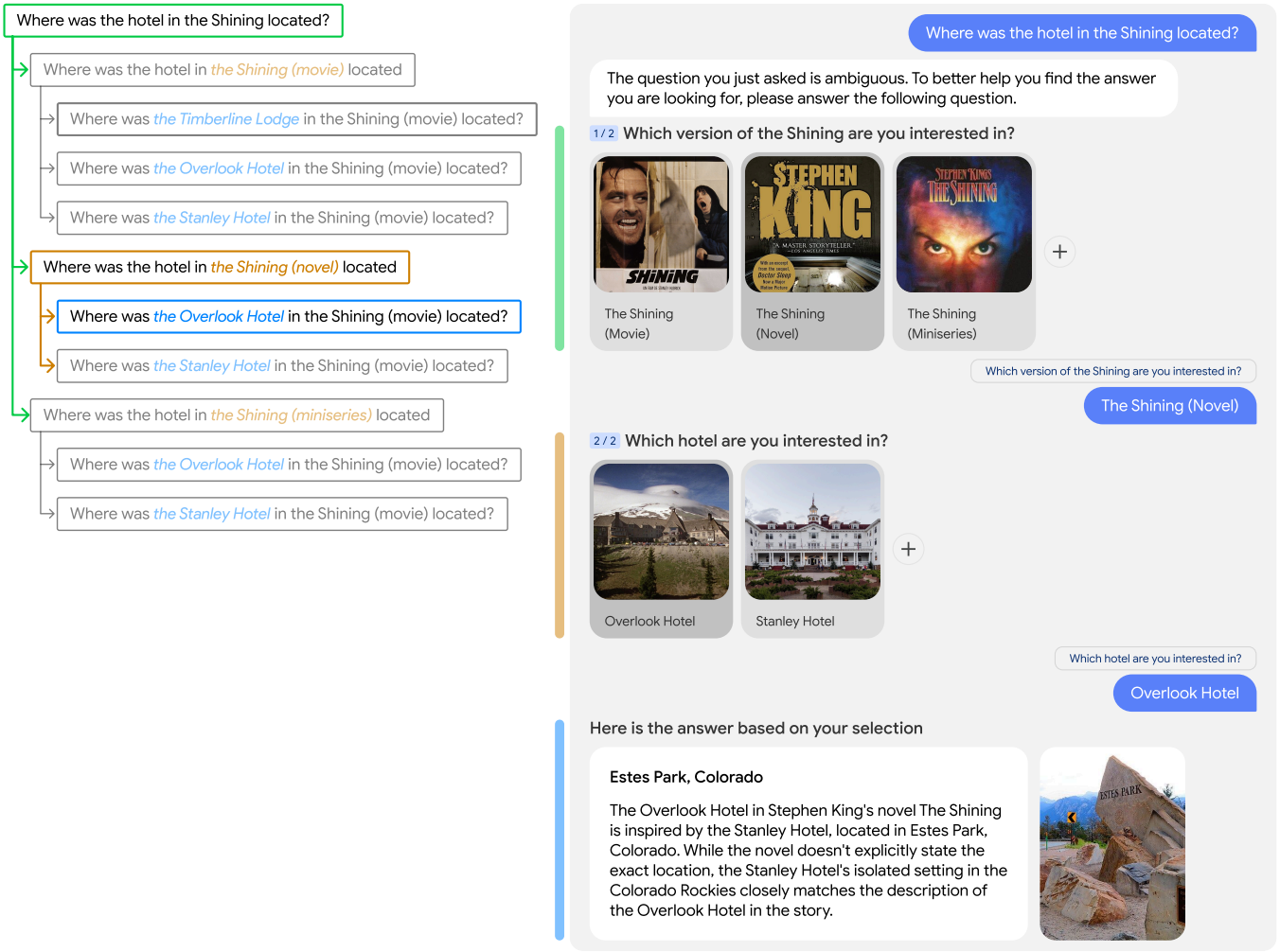


**Figure A.1: AmbigChat identified two ambiguous facets in the query "*Where was the hotel in The Shining located?*". The Overlook Hotel is a fictional location in *The Shining* and is based on the real-world Stanley Hotel. This poses a question as to whether these two hotels should be disambiguated as two distinct entities. *Image Credit: Wikimedia Commons, The Shining (Movie) / The Shining (Novel) / The Shining (Miniseries) / Overlook Hotel / Stanley Hotel / Estes Park, CO.***

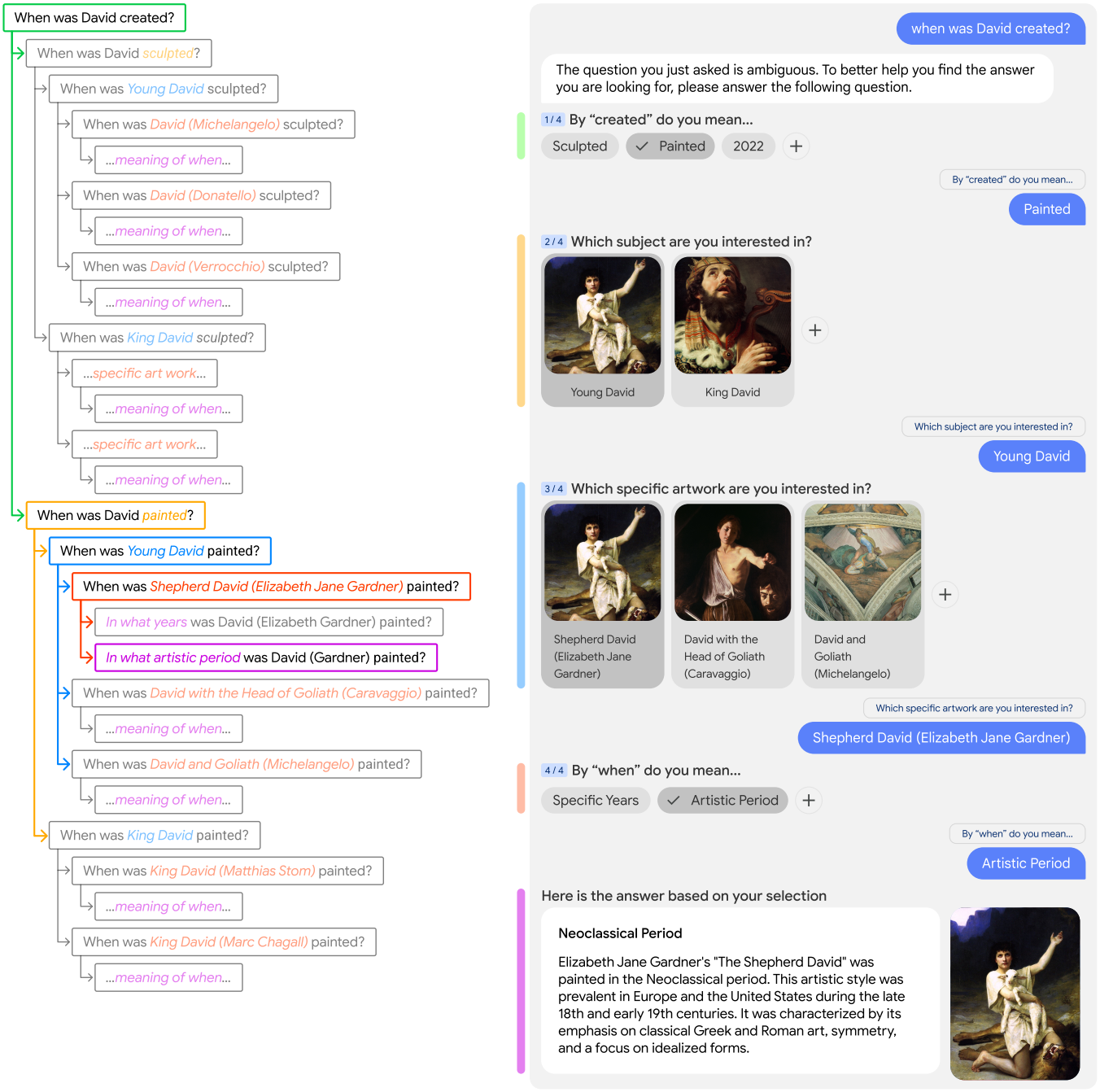


**Figure A.2: AmbigChat identified four layers of ambiguity in the query "*When was David created?*" (*left*). In this conversation enabled by AmbigChat, the user finds a lesser known painting of young David as a shepherd, instead of the more famous sculpture *David* by Michelangelo. We used this as one of the queries in our user evaluation. *Image Credit: Wikimedia Commons, Shepherd David / David with the Head of Goliath / David and Goliath / King David.***