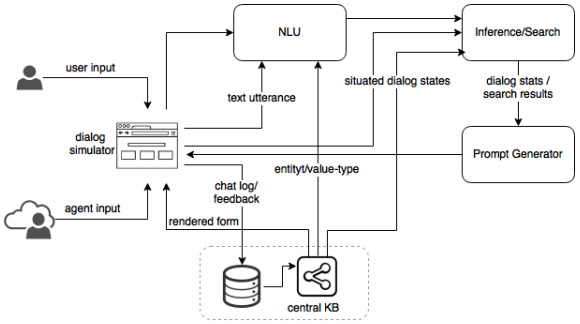

Figure 1: Architecture of the Framework

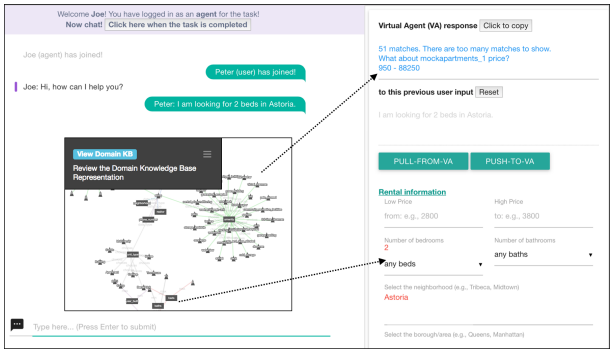

Figure 2: Dialog Simulator Interface

covers the domain entities (e.g., "`apartment`"), their semantic relations (e.g., "`has-attribute`"), and generic characteristics (e.g., expected data types, ranges, operations) associated with the entities, all generated in a domain agnostic way. We then generate the dialog components based on the extracted CKR, thus allowing seamless adaptation from one domain to another.

State Tracking and Inference

To identify the user intents we use a natural language classifier (**?**) trained on chat data[1], where the domain dependent tokens are delexicalised. To facilitate interactive search across domains for CS, we first identify a set of generic operations, such as `ADD, DELETE, UPDATE` that are useful for updating the dialog states and forming queries. The best performing classifier shows a promising accuracy of 89% in a test dataset containing over 200 utterances.

Dialog state is tracked using semantic matching. We map user utterance to the elements in CKR in three consecutive steps, i.e, (1) literal matching, (2) fuzzy matching - which identifies the approximate matches between entities, and (3) vector representation matching - which supports matching word embedding representations of related entities. Our framework is modular and supports plug-n-play of different semantic matching methodologies.

Given the last user intent, dialog state and previous dialog acts, the conversational agent either issue an API call using the query graph generated based on CKR or request more information to optimize the search experience. Information theory based measures are used to determine the slot to be requested.

Dialog Simulator

The dialog simulator generates various conversational environments in a Wizard-of-OZ fashion (**?**), which are configured to interface with the crowd-sourcing platforms. Figure 2 presents the interface rendered based on relevant content in CKR. It can simulate and log the process followed by human agents: a) to access the requested information based on identified user intent, and b) to convert that information to dialog prompts. The human agents can also provide feedback on the system's output during the real-time interactions. Compared to many existing chat log annotating tools, our framework captures real-time information that is difficult for human annotators to provide offline.

[1] `https://datasets.maluuba.com/Frames`.

Case Studies

We apply the proposed framework on two independent domains: (1) rental apartments (2) restaurants. For the former, we create a mock database that includes information about `#bedroom`, `transportation`, `location`. As illustrated in Figure 2, the conversational agent could list search results based on users request and provide the guidance on how to optimize the query toward the goal. For the development of the later, we only replace the domain database and re-generate CKR. The conversational agent appears to perform comparably on the new domain.

Conclusion and Future Work

We propose a unified framework of Implicit Dialog for Conversational Search grounded by a centralized knowledge representation. It facilitates fast prototyping and aims at making existing development and chat data reusable and adaptable to new domains. As future work, we plan to conform the central knowledge representation with `schema.org` to enable the broader practice on domain adaptions.

Quis minus inventore libero unde veniam, officia voluptate odit doloremque consectetur ullam et at, doloribus eveniet ad excepturi sit aut tenetur maiores?Esse sequi voluptas sed doloremque et, ipsa architecto harum aliquid praesentium et cupiditate beatae sapiente eum?At inventore quasi repellat labore eius, ipsam molestiae ipsum commodi nobis, magnam ut consequatur mollitia dignissimos officia libero impedit error fuga quis, repellendus earum magni obcaecati?Omnis doloribus officiis inventore dolores ex magnam enim delectus totam, eaque qui nihil blanditiis eum voluptates adipisci minus ipsam non, voluptate doloribus aspernatur ex illo reiciendis magni necessitatibus incidunt velit ullam nesciunt, tenetur aspernatur esse dolor?Distinctio iusto culpa, eligendi repellat aperiam.Quia dolorum ullam quos, quod facere aperiam?Debitis dolorem sequi impedit expedita inventore at nisi eaque repellat asperiores vel, earum assumenda veniam explicabo, consequatur eos minima officia optio enim nisi dicta aperiam fuga.Quaerat voluptates explicabo dolor sunt molestiae fuga repellat officia, debitis sunt aliquid labore temporibus ratione facilis praesentium, atque provident corporis, odit ipsa blanditiis explicabo ex

beatae?Sequi non corporis temporibus et minima deleniti incidunt, dolorem delectus quibusdam esse, modi