Semantic Parsing for Location Intelligence

Voicebox’s advanced system for helping you navigate the world

Voicebox’s Location Intelligence system combines cutting-edge Semantic Parsing for natural language understanding, sophisticated query processing, and Voicebox’s deep experience with context and voice interaction. The result is a system that gives users an unmatched level of natural interaction for navigating the world and accomplishing what they need to get done.

The Location Intelligence system is like having someone by your side who knows where everything is—from businesses to restaurants, museums, parking, and more—and understands how real people talk about those things. Unlike the faux-intelligence of older systems that can only help you if you already know exactly what you need, Location Intelligence links a wide range of real-world information to Voicebox’s industry-leading natural language understanding software.

This paper provides an overview of what the system is and how it works, with examples that demonstrate why Voicebox’s Location Intelligence system is unmatched in the industry.
Location Intelligence

Location Intelligence is Voicebox’s system for helping users find what they need and get where they’re going. Simple computer-assisted navigation is now a well-solved problem, with many offerings from many different vendors. Voicebox’s Location Intelligence system goes beyond navigation to enable a wide range of previously unsolvable problems:

- Navigation, even when users can’t give the specific address of their intended destination (“Get me directions to the Bartell’s drugstore in Redmond”),
- Helping users determine a destination based on potentially vague criteria (“Are there any hotels near the train station?”)
- Find and interact with “points of interest” (POIs) based on attributes such as business name, type of business, services available, hours of operation, and more (“Find parking near the Japanese restaurant across the street from Macy’s.”)
- Supporting queries that rely on proximity to other POIs or combinations of POIs (“What’s that bank near First Avenue and Pine Street?”)
- Multi-stage operations such as finding a POI based on certain criteria, then performing another action such as sending the details to a contact or looking up the operating hours. (“Send the address of the nearest dentist to Tim”)
- Performing such multi-stage operations in a single query or split across consecutive queries. (“How many coffee shops are in Bellevue?” “Show me the nearest one”)

To handle queries such as these, the Location Intelligence system combines cutting-edge natural language understanding and query optimization methods to bridge the gap between what users say and the many disparate sources of data needed to answer their questions.
Semantic Parsing

Semantic Parsing is the natural language understanding technique behind Location Intelligence. Semantic Parsing converts natural language expressions into symbolic “logical forms” that represents their meanings. Logical forms are recursively-structured, meaning that they can support arbitrarily complex or nested queries, such as “Find the address and phone number of the Italian restaurants near 4th Avenue and Virginia Street and send them to Bob.”

Logical Forms

The first step in Location Intelligence is converting the user’s utterance into a logical form. For example, Semantic Parsing reduces the complex query “Find the address and phone number of the Italian restaurants near 4th Avenue and Virginia Street and send them to Bob” to the following logical form:

```plaintext
answer([A, B, C, D] ( send(A, E), location(B, A), amenity(B, italian), near(B, F), intersection(F, G, H), street(G, '4th avenue'), street(H, 'virginia street'), person(E, bob) send(C, I), phone_number(D, C), amenity(D, italian), near(D, J), intersection(J, K, L), street(K, '4th avenue'), street(L, 'virginia street'), person(I, bob) ))
```

This logical form means, roughly, that the set of items {A, B, C, D} constitute an answer to the user’s query if:

- A is the location of B
- B matches Italian Restaurants (a type of amenity) near F
- F is a spatial intersection of G (4th Avenue) and H (Virginia Street)
- C is the phone number of D
- D matches Italian Restaurants near J
- J is a spatial intersection of K (4th Avenue) and L (Virginia Street)

And having found such an answer, send both A and C to Bob.

Query Optimization

Notice that the above logical form contains two send clauses, and that each one replicates the same underlying search for restaurants. This is due to the particular phrasing the user gave and their request to send two pieces of information to Bob.
Query optimization is the step which identifies any redundant predicates in the logical form so it can build an optimal set of queries to any underlying data sources. This is important for overall system responsiveness, cost management, and resource efficiency. Some predicates, such as the spatial intersection predicate, may be compute-intensive. Others, such as amenity or phone_number, may require queries to high-latency web APIs such as Yelp or to paid data sources. Query optimization ensures that users’ queries are resolved as efficiently as possible.

**Cross-Service Querying**

In general, there is no guarantee that all the predicates in a logical form will relate to the same data source. The street predicate, for example, could be satisfied by OpenStreetMaps or any number of other map data services. The phone_number predicate may require a call to a business directory service such as Yelp. The person predicate requires access to the user’s personal address book, which varies per-user.

Because the data necessary to answer the user will come from many different sources, Location Intelligence can implement a framework for “handlers” that enables cross-service querying by isolating the processing for each predicate in a logical form. The handlers take care of any differences in the underlying data sources.

In this way, the Location Intelligence system can merge information from web service APIs, traditional databases, local data sources, cloud services, and more into a single seamless experience. The logical form also allows the system to process logically-independent predicates in parallel, while allowing logically dependent predicates to be deferred until the results from other predicates are available.

This architecture also makes the Location Intelligence service highly extensible, since custom data sources can be added as easily as creating a handler for them. In principle, Location Intelligence can interface with a user's e-mail, scheduling, banking information, family calendar, stock quotes, or anything else developers write a handler for.

**List Management**

Many user queries return lists of potentially applicable items, rather than single results. Voicebox has over a decade of experience in voice-based list management, which the Location Intelligence system leverages. Lists are a fully supported data type, allowing users to page through lists by voice, sort, filter, and all the other natural list interactions users expect.
Context

Users don’t always know exactly what they want when they make a query. Further, the results of a preliminary query often affect their subsequent queries. Many voice interaction systems fail when faced with these cross-query scenarios; those systems only work on one query at a time, and cannot remember the results of previous queries to make use of them later.

Voicebox also has deep experience with and industry-leading capabilities for managing context in voice interactions. This, too, is fully leveraged by the Location Intelligence system, allowing it to support queries which can only be understood relative to previous queries or their results.

Where other voice interaction systems fail on multi-utterance sequences such as the following, Voicebox’s Location Intelligence system handles them without trouble:

User: “Get me directions to Bartell’s Drugstore in Redmond”
Location Intelligence (LI): shows list of three matching locations
User: “The one by Redmond Town Center”

In this sequence, the phrase “the one” refers to some item on the list; the whole utterance can only be understood if the system can understand “the one” as a reference to an earlier result.

User: “What’s the closest pizza place?”
LI: shows the matching result
User: “Does it do delivery?”

Similarly, in this example the word “it” refers to an earlier result. The context system also handles multiple objects and can distinguish references to them by type, as in this example:

User: Where is my two o’clock sales meeting?
LI: Shows the location
User: Is Bob free then?
LI: Shows Bob’s free/busy status
User: Send that to him.

By inferring the user’s intended data types throughout the conversation, the Location Intelligence system’s context system understands “then” as a reference to two o’clock, “that” as a reference to the location of the sales meeting, and “him” as Bob.