Handling Multi-Sentence Queries in a Domain Independent Dialogue System

[1]**Prathyusha Jwalapuram** [2]**Radhika Mamidi**
Language Technology Research Center,
Kohli Center on Intelligent Systems,
International Institute of Information Technology,
Hyderabad, India
[1]prathyusha.jwalapuram@research.iiit.ac.in
[2]radhika.mamidi@iiit.ac.in

Abstract

This paper discusses the handling of multi-sentence queries in a mixed-initiative dialogue system based on a hierarchically structured knowledge base, in a way that is domain independent. The system is rule-based and uses dependency relations and part-of-speech tags obtained from the Stanford Parser coupled with the hierarchical structure of the knowledge base to identify the user's goal. The system was tested for its accuracy over answering questions, and also subjective testing was done to evaluate the dialogue flow; primarily over the books domain. We show examples of the system developed over the domains of books, movies and restaurants to demonstrate the domain independence.

1 Introduction

Most dialogue systems focus on processing the user's input and classifying the dialogue in terms of the amount of information it presents and the possible paths the dialogue could take. The idea is to predict a possible goal of the user in order to be able to ask relevant questions if needed in order to fill information gaps, and provide more relevant replies. Using a hierarchically structured knowledge base for a dialogue system helps achieve this. They help us limit the possible paths of the dialogue, and can help us identify irrelevant inputs or topic changes.

Few dialogue systems attempt to process multi-sentence inputs (multiple utterances in a single user turn) that collectively behave as a query. Users of a dialogue system may break up their queries into multiple sentences, and also provide additional information and qualify their initial statements. In such cases it might not suffice to simply find all the relevant keywords; it might be important to understand the relationships between them as well.

Aided with dependency relations, part-of-speech tags (provided by the Stanford parser) and the inherent semantics of some words such as 'and' or 'but', we attempt to link the keywords in complex sentences and multiple sentences to get a clear picture of exactly what the user is looking for. Significant information can be gained by looking at function words and their dependents. We also attempt to look at expressions of negation and negative words indicating the exclusion of certain objects as required by the user.

The system is domain-independent, and we present an implementation over a simple, hand-crafted knowledge base of books, movies, and restaurants, that is essentially structured into a useful hierarchy. We evaluate the system based on whether the replies are relevant or not on multi-sentence queries that were collected through a survey (objective evaluation) and qualitatively through participant interaction and rating (subjective evaluation).

The paper is organised as follows. Section 2 describes related work, section 3 describes the strucure of the knowledge base, section 4 describes the dialogue manager, section 5 shows some examples of dialogue demonstrating domain independence, section 7 is about the evaluation and error analysis and section 8 explores possible future work and the appendix at the end has more examples of dialogue with multiple-sentence inputs.

2 Related Work

The advantages of using an ontological knowledge base for a dialogue system were put forth in Milward and Beveridge (2003) which defined an ontology simply as a network of concepts and

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 122–130

instances related to each other through semantic links. They introduce a mixed-initiative dialogue system that uses part-whole and is-a relationships to drive clarification questions and determine the sequence of the dialogue. Since we seek to further simplify the structure into a more generic relationship set (rather than is-a and part-whole relationships), we refer to our knowledge base as hierarchically structured.

It has been argued that the separation of the dialogue management from the domain knowledge management helps reduce the complexity of the systems and enhance further extensions (Flycht-Eriksson, 2004). Flycht-Eriksson (2004) uses is-a and part-of relations to resolve issues of under and over specification. We use a similar approach to make our implementation domain independent; swapping the knowledge base with another in the same format would produce a working dialogue system for the new knowledge base, even of a different domain.

Dzikovska et al. (2003) describe a system that maintains two ontologies, a domain independent ontology for a parser linked to the lexicon to capture the aspects of dialogue interaction that are common across domains and a domain-specific ontology for knowledge representation, and they integrate the linguistic and domain knowledge by defining a set of mappings between the two.

Division of the task and the dialogue is also explored in Bohus and Rudnicky (2003). A dialogue engine generates domain independent conversational behaviors and the dialogue task specification is handled separately. RavenClaw uses hierarchical task decomposition, which has a tree-like structure. The dialogue task specification is domain specific.

Lee et al. (2009) propose an example-based dialogue modeling that is applicable over different domains, but requires dialogue corpora for each domain; they also do not consider multi-sentence utterances in the course of a dialogue.

Mazuel and Sabouret (2006) propose a generic command interpreter for natural languages that uses an ontology to clarify semantic concepts, coded in a specific language. They use a tokenizer, tagger, lemmatizer and a chunker, but discount the need for a grammar based syntactic parser; they remove stop words and treat the sentences as a bag of words, and use the ontology for concept matching and semantic analysis; they do not take into account the semantic information that is provided by function words such as 'in', 'and' and so on. They assume that the users' commands are unlikely to be complete sentences, and therefore do not consider multiple sentences at all.

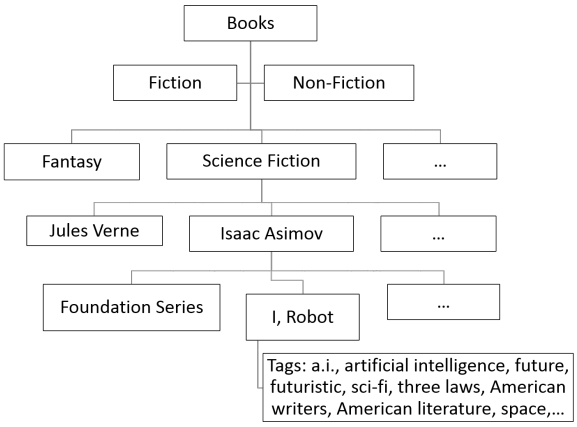

Figure 1: Hierarchical Knowledge Base Structure

Bickmore et al. (2011) describe an ontology based dialogue system that simulates a health counselor, using an RDF-based ontology described in OWL. They maintain a representation of a plan tree, recording the recipes that are actively being used and their goal-subgoal task decomposition relationships. Their dialogue fragments are modeled through a task representation language, and a dialogue planner enacts these fragments. They introduce the notion of adjacency pairs which are logically related, consisting, for example, of an utterance and its response.

Bharati et al. (1995) proposed a Computational Paninian Grammar framework for interpreting natural language queries, for example by creating verb frames of a list of domain-specific verbs in order to identify relationships between the keywords. We try to make it a domain independent implementation by using dependency relations with less significance given to the verbs, allowing us to focus on the relationships between the prepositions, adjectives, nouns/noun phrases, etc.

3 Hierarchically Structured Knowledge Base

The knowledge base is structured in a hierarchical manner for ease of representation, and also to facilitate dialogue flow. For example, in the *books* domain, the books maybe in a hierarchy from *genre* to *author* to *title* and so on. This helps

form a possible path that a dialogue may take; for example the user may specify a *genre* they like, and the system may suggest relevant *authors*; the user may then pick an author and the system suggests *books* by the author, and so on.

Another reason for maintaining a structured knowledge base rather than a standard ontology is the flexibility in defining relationships. In our knowledge base, the entities are related by a *'is-x-of'* relationships; that is, *is-author-of*, *is-genre-of*, etc. This could allow us to structure knowledge bases in domains which may not fit into the traditional ontology structure. We can also define synonyms for the relationships for making search easier *(author, writer)*.

Tag Information

Although typical data about books consists of information like author, year of publishing, genre, etc., in order to answer queries, the system must also be to consider what the books are about. This is problematic since we usually do not have information about the content of the books; the titles of the books are not always informative, and a large set of questions would have to go unanswered.

To resolve this issue and expand the scope of queries being answered, tags provided for books by the general public on popular book sites were collected and added to the knowledge base (or alternatively, high-frequency content words from book reviews can also be scraped). Tags are commonly used in searches for a similar purpose, allowing you to look for a book based on themes, characters, and other classifications which are not necessarily genres or part of a typical knowledge base.

Consider a query like *books with magic and animals in them*. The *Harry Potter* series, for example, would be a good fit for such a query; but nothing in any of the titles of the series would directly imply this. However, some of the tags for the series include *dark magic, witches, wizards, young adult, British fiction, beasts, dragons*, etc, which would allow us to infer that both *magic* and *animals* play a part in these books (synonyms must be considered).

Similarly, *women writers, female writer* are part of the tags for both the *Harry Potter* series and the *Hunger Games* series; this helps us return results to queries like *Do you have books written by female authors?*. All additional information, like prizes won by the books, the time period it is set in (for example, *detective fiction set in the Victorian times* = *Sherlock Holmes*), etc. is available to us through these tags, enabling us to provide results to a wide variety of requirements; including spelling variations (*theatre, theater*).

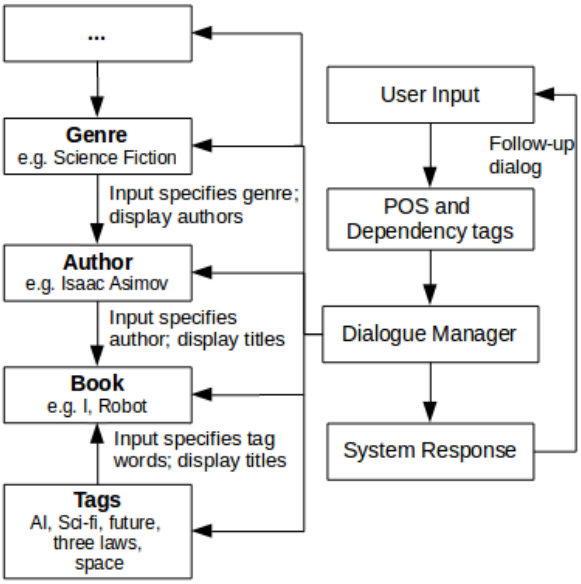

Figure 2: Simplified System Architecture

For the movie domain, similar *'tags'* are available in terms of *'plot points'*, or high-frequency content words can be scraped from movie reviews on popular movie review sites. For the restaurant domain, we use the restaurant menu item descriptions (which are mainly descriptions of ingredients and the like).

4 Dialogue Manager

Our emphasis was to try and identify the objective of a user's query, that is, the data the user is looking for and the constraints pertaining to this data. We use the Stanford POS Tagger and the Stanford Dependency Parser for this purpose while assuming that the queries are free of errors and are fairly grammatically sound. We don't attempt to resolve abbreviations and ambiguities. The system is able to extract information from ungrammatical queries in certain cases where the Stanford Parser is able to generate a fairly accurate dependency parse.

We consider the cases where the user directly specifies what they are looking for, or makes oblique remarks that are intended to help reach the goal. These statements can be simple, complex or span multiple sentences.

An online survey was conducted to obtain questions for the development and testing. A total of 57

participants submitted 118 questions. All domain specific assumptions and rules are based on a development set of 68 questions; the other 50 were separated for testing.

4.1 Motivation behind using Dependency Relations

The advantage of using dependency relations is that they are syntacto-semantic relations, so the same question formulated in different ways can lead to the similar dependency relations (such as a statement in active/passive voice); this allows us to easily group similar user queries without having to anticipate all the possibilities.

Dependency relations also give us an idea of the relationship between the words and therefore the information the user is looking for and the constraints. This information cannot be obtained from simple syntactic parsing, such as Phrase Structure Trees, as they are highly dependent on the word order.

4.2 Processing the parser output

Keywords and Negation

An example of an input by the user could be *I like Oscar Wilde*. Here, the parser should tag *like* as the root, *I* as the nsubj (subject) and *Oscar Wilde* as the dobj (direct object). We see that the object here is the keyword we are looking for. Using such information, keywords in the utterance can be detected (Jwalapuram and Mamidi, 2017).

Similarly, we also consider cases where the user expresses a negative sentiment, such as *I don't like Oscar Wilde*. Here the negation applies to the *root*, and *Wilde* is the object of the root, so we transfer the negation from the root to the keyword; we remember to eliminate *Oscar Wilde* from any results we provide to the user. We also recognize lexicalized negatives, such as *hate, dislike, awful, stupid* etc. through a simple dictionary list of negative emotions. By maintaining a list of rejected candidates, we prune the results tailored to the user's requirements.

Prepositions, Modifiers and Conjunctions

In order to maintain domain independence, we cannot attribute specific interpretations to function words, which are often overloaded. We use prepositions as a clue merely signalling relationships between two keywords (nouns or verb and noun). This relationship does not need to be identified; we use the knowledge base to detect this. For example, if the user says *"I want books by Dan Brown"*, we identify that *"books"* and *"Dan Brown"* are the keywords, and are related. A reference to a topic in *'What books on psychology are available?'* is identified in a similar way, as related keywords *"books"* and *"psychology"*. Useful modifications through adjectives include cases like *I want funny books* or *Do you have scary books?*.

And and *or* are treated somewhat similarly. If a user says *I like magic and animals*, if there are no books which are about both *magic* and *animals*, we would still want to return some results, which are about magic or animals. This would be equivalent to the user saying *'I like magic or animals'*; in this case, all books which have references to either *magic* or *animals* are part of the results (see also Table 11 in Example Dialogs). For *but*, it is a clear case of contrast; any keyword obtained from a *but* clause are included in the reject list. Consider *I want books about vampires but not Twilight* or *All books by Bernard Shaw but Pygmalion*.

Relative Clauses Consider *I would prefer books which have some mystery in them*. Here the user is looking for *books*, which is the argument of the main verb *prefer* and the user specifies that the books need to *have* (secondary verb) *mystery* (object of secondary verb). Note that the prepositional argument here is *them*; since it is a demonstrative pronoun, it can be discarded as irrelevant. Using dependency relations, it is possible to extract the information that our keywords *books* and *mystery* are therefore related (see Table 8 in Example Dialogues).

In case of *but*, the same assumption of contrast applies; we simply identify the relevant keyword as a reject qualification and eliminate it from the results (*I would like books which are about traveling but not those which are focused on Europe*).

4.3 Multiple Sentences

We have a similar assumption here, that the user is likely to state their primary requirement first, followed by further qualifications. Since we add any keywords which have been associated with negative words to a reject list, and simply add any other keywords to a list of possibilities, the explicit order does not really matter. However, it is assumed that the set of sentences which occur together are related; which means that any qualifications that are made are applied collectively.

Consider *I like books with humor. I don't like*

Me: I like science fiction. Maybe Asimov.
System: I found these under Science Fiction: *H. G. Wells* *Jules Verne* *Isaac Asimov* *George Orwell* *Aldous Huxley* I found these under Isaac Asimov: *I, Robot* *Foundation Series* *Caves of Steel* What do you think? Do you want books from the above?
Me: I like science fiction. I think Asimov is awful.
System: You could try: *H. G. Wells* *Jules Verne* *George Orwell* *Aldous Huxley* What do you think? Do you want books from the above?

Table 1: Multiple Sentence Inputs

Wodehouse. Here, we first identify humor as a keyword from the first sentence; since the structure of the ontology classifies authors under genres, in this case we return a list of authors classified under *humor* to the user. The next statement is assumed to be related to the first; it specifies that the user does not like the author *Wodehouse*. This keyword is added to the reject list; when the list is specified to the user, we search for and omit *Wodehouse* from the list (Table 1).

Through the dialogue for this user, the keyword *Wodehouse* stays in the dialogue manager's reject list, so that any future cases are also eliminated. This also helps us omit references in cases where it is not immediately applicable; for example, if the user input was *I like books with humor. I don't like the Jeeves stories.*, we would add that to the reject list, and if the user chose to look at books written by *Wodehouse* in a future dialogue, we omit the books which are about *Jeeves*. Some other pragmatic instances for elimination can also be considered (*I have already read the Jeeves stories*).

Similarly, it is also possible to answer questions which are more descriptive in style. Consider, *I am trying to find a poem I read back in school. There's a man who shoots a bird or something and things start to go wrong. I think he was a sailor.* We get a set of related keywords from each utterance: *poem-school; man-shoots-bird; sailor etc.*

4.4 Identifying Relationships in the Knowledge Base

Given a set of related keywords, the system searches through the knowledge base and matches them to the entities it finds there. In order to make the search efficient and the results relevant, the search is prioritised. For example, the *x-of* relationships are searched for matches first; next the search moves through the entities through the hierarchy in order, ending at tags.

So, in our *"I want books by Dan Brown"*, we found that *"books"* and *"Dan Brown"* are the related keywords; the system finds *books* is an *x-of* relationship as part of the knowledge base, and *Dan Brown* is an entity under *author* which has an *x-of* relationship with *books*. The system therefore returns a list of books under the author *Dan Brown*. In the case of *'What books on psychology are available?'*, related keywords *"books"* and *"psychology"* are identified as an *x-of* and an entity under either *genre* or *tag* respectively, and the system similarly returns books classified or tagged as *psychology*.

In case of multiple sentences, the keyword relationships are identified individually from each sentence and then collated. This helps us retrieve a list (*I like books with humor"*) and then eliminate unwanted results (*I don't like Wodehouse*); or alternatively narrow down possibilities. Consider *"I am trying to find a poem I read back in school. There's a man who shoots a bird or something and things start to go wrong. I think he was a sailor."*. We identify *poem* as a *genre* and *school* as a *tag* or perhaps a *title*; once we add *man/shoot/bird/sailor* we continue to narrow the possibilities down and perhaps finally find all of them as *tags*. The system then returns the relevant title *(Rime of the Ancient Mariner)*.

4.5 Dialogue Flow

The hierarchy of the knowledge base guides the flow of the dialogue. The dialogue manager traverses the hierarchy and locates itself on one of the nodes, and uses the meta information of the node (genre/author, etc) to frame its replies or ask further questions. If user input is unclear and the manager cannot locate itself in the knowledge base, the system asks the user to rephrase.

Typically a user's input is processed, the keywords and the rejects identified, and the relevant results are displayed. The user may then choose to move up or down the hierarchy, or may change the topic entirely. This is identified by the break in the chain of the path being followed that is, if the user is not simply moving up or down the path but breaks the hierarchy chain such that the dialogue manager must relocate itself in the knowledge base, then the topic (or user goal) is assumed to have changed.

Consider a user who asks for some *poetry*. The user is presented with a list of authors, say *Yeats, Shelley, Wordsworth* and *Coleridge*. The chain is now from *genre* to *authors*. Next, the user may choose one of these authors, say *Wordsworth*. Then a list of poems written by *Wordsworth* is presented to the user. Now the chain goes from *genre* to *authors* to *poems*, and so on.

Similarly, a move up the hierarchy is also possible. A user may say that he wants books like, say, *The Time Machine*. The system may then present the user with a list of books written by the same author, i.e., *H. G. Wells*. The chain has now formed from *books* to *authors*. The user may express interest in other authors who write similar books; the chain then moves up to *genre* and the user is presented with a list of *authors* who write *Science Fiction*, and so on.

4.6 Topic Change

The user may explicitly jump around the hierarchy, making the dialogue mixed-initiative. For example, the user may choose to go back to the list of authors under a genre after being presented by a list of books (*go back to fantasy*) or the user may skip a level by directly specifying the book he's interested in when presented with the authors (*do you have The Adventures of Tom Sawyer by Mark Twain?*) (see Table 7 in Example Dialogues). At any point, if the user does not move up or down from the current location in the hierarchical structure, then the chain is broken. A change in user goal is assumed to have occurred, and the dialogue manager relocates the current reference node in the hierarchy. The chain is restarted from the location which is specified by the keywords obtained from the user's input.

4.7 Ambiguities and Clarification Dialogue

In order to resolve certain inherent ambiguities, the system engages the user in a simple clarification dialogue. For example, if the user asks for *books with animals in them*, the keyword relationships set up may match with books with *animals* in the title *('Animal Farm')* or generally books tagged with *animals (Harry Potter, Black Beauty, Animal Farm)*. In this case, the system will ask the user by presenting each option in order of priority until the user accepts the results.

Queries such as *books by Dan Brown and Ayn Rand* can also be considered ambiguous in the sense that the user may be looking for an intersection (books written by *both Dan Brown and Ayn Rand* or a union *books by Dan Brown and books by Ayn Rand* (which is a case of ellipsis). In such cases, the system asks the user if s/he wants the results of the intersection, and if rejected, displays results of the union (see Table 11 in Example Dialogues). In case the system does not find any results for intersection, the system specifies so and automatically presents results of the union.

The system only looks for negative indication in clarification *(no, I don't think so, etc.)* in order to be able to accept a wider range of positive indications *(yes, alright, okay, sure etc.)*.

5 Domain Independence

The system utilises the structure of the data to recognise goals and paths; there is no domain knowledge involved in the dialogue manager itself, or in fact any other module. The knowledge base is therefore replaceable, and the system is able to work in the same way for a different domain. In order to demonstrate this, examples are provided in Tables 2, 3, and 4 (additionally, Tables 9 and 10 in Example Dialogs) to show that the system works just as well with a similarly structured knowledge base for a different domain. Here, in the movies domain the *x-of* relationships are in a hierarchy consisting of *genre, director, title, actors, plot points*, etc. For the restaurant domain the *x-of* relationships are *location, cuisines, name, menu, description*, etc.

6 Evaluation

6.1 Accuracy

The system was tested on a set of questions for the books domain obtained through an online survey. A human evaluation of the system's responses was conducted based on relevance, appropriateness and correctness. The evaluation is based on whether the intended keywords and any associated

Me: I like science fiction.
System: I found these under Science Fiction: *H. G. Wells* *Jules Verne* *Isaac Asimov* *George Orwell* *Aldous Huxley* What do you think? Do you want books from the above?
Me: I will try Isaac Asimov's books. **System:** I found these under Isaac Asimov: *I, Robot* *Caves of Steel* *Foundation Series*
Me: What is in the Foundation Series?
System: I've found a book series: *1. Prelude to Foundation* *2. Forward the Foundation* *3. Foundation* *4. Foundation and Empire* *5. Second Foundation* *6. Foundation's Edge* *7. Foundation and Earth*

Table 2: Books: Moving Through the Hierarchy

System: How can I help you?
Me: I like science fiction.
System: I found these under science fiction: *Steven Spielberg* What do you think? Do you want movies from the above?
Me: I will try Steven Spielberg movies. **System:** I found these under Spielberg: *Indiana Jones* *Schindler's List* *Jurassic Park* *Catch Me If You Can* *Saving Private Ryan*
Me: Please tell me about Indiana Jones.
System: I've found a movie series: *1. Raiders of the Lost Ark* *2. Temple of Doom* *3. Last Crusade* *4. Kingdom of the Crystal Skull*

Table 3: Movies: Moving Through the Hierarchy

constraints are correctly identified. A result is considered to be correct if the relevant or expected output is returned. An overall accuracy of 88% was achieved over 50 questions (Table 5).

A second evaluation was carried out to test the dialogue system qualitatively by means of user responses to questions in a Likert-scaled questionnaire; 8 participants were asked to interact with the system (over about 8-10 turns) and rate it on a scale of 5 on the basis of relevance (whether they thought the system understood their goals), correctness (compare expected response with actual system response) and an overall rating (navigation, task completion, etc). A Wizard-of-Oz benchmark was also collected. The results are in Table 6.

6.2 Error Analysis

There are many errors introduced due to ambiguous structures, grammatically unsound instances, etc. Parsing errors, like wrong identification of root, incorrect argument assignment, etc. also introduce significant error as the system is unable to identify the right keywords.

Parsing Issues: There is a prepositional phrase attachment problem - consider the following queries:

(a) *Do you have Catcher in the Rye?*

(a) *Do you have Of Mice and Men in the library?*

(b) *Do you have Body in the Library?*

In (a), the prepositional phrase *in the Rye* is part of the title and important information; in (b) *in the library* is irrelevant information. It may also not be possible to ignore this phrase on the basis of the domain, as we can see from (c) where the title of the book itself has *in the Library* in it. In such cases, we may end up including irrelevant keywords and retrieving wrong information. Similar problems also occur when the book titles are phrases or sentences themselves (*Do you have Where the Wild Things Are?*) which are considered as part of the sentence by the tagger and assigned incorrect roots or copula tags.

Pragmatics: Consider *'Which is the most popular dish in RESTAURANT-2?'*. Converting *'the most popular'* to mean that we need to group by dish name for the restaurant, then look at the ratings, frequency of orders or other relevant information, is hard to do even if such information is available in the knowledge base, and such an inference includes information not present in the query. Also, the thresholds of such subjective opinions may differ.

Recognizing the User's Goal: Consider the example, *What other books are there have that*

System: How can I help you?
Me: I like chinese food.
System: I found these under chinese food: *RESTAURANT-16* *RESTAURANT-24* What do you think? Do you want menus from the above?
Me: I will try RESTAURANT-24. **System:** I found these under RESTAURANT-24: *Hakka Noodles* *Chicken Fried Rice* *Chilli Mushroom* *Chicken 65*

Table 4: Restaurants: Moving through the Hierarchy

Type of Input	Accuracy
Single line	88.09%
Relative clauses	66.66%
Multiple lines	100%

Table 5: Survey Query Evaluation

were written by the author of The Old Man and the Sea?. Here, we need to first identify the *author* of the *The Old Man and the Sea* as Hemingway, and then retrieve books written by Hemingway. This retrieval may be complex and hard to identify.

7 Future Work

Pre-processing problems involving spelling and grammatical errors, synonyms, missing arguments, abbreviations, etc. need to be handled in order to make the system practically useful. Also, the words must be lemmatized and replaced with a mapped, representative synonym that is part of the knowledge base. A chunker or an NER can be used to identify complete constraints with functions words which the parser might separate (like *'Harry Potter and the Goblet of Fire'*). Ellipsis and anaphora issues need to be identified in depth.

A more robustly domain-independent system which can work despite significant changes in the knowledge base structure must be developed. For example, the system should be capable of handling data having complex networks and interconnections among the entities (instead of a simple hierarchy as we have used).

Instead of a simple dictionary, a better method for analyzing the positively and negatively associated words expressed in queries can be found, instead of a simple dictionary. Keeping track of both positive and negatively associated keywords could prove helpful for providing suggestions.

Category	Woz Benchmark	Avg. Rating out of 5
Relevance	4.625	4
Correctness	4.5	3.5
Overall	4.25	3.5

Table 6: Subjective Evaluation

A corpus with a wide variation in the pattern of questions, along with questions in statement form must be analyzed and considered, in order to make the system more accurate for generic queries. The usefulness of the information gathered through this method and the ease of mapping to a data representation must also be studied. More rigorous testing needs to be done for the descriptive questions.

The possibility of increasing the accuracy of keyword identification by using statistical machine learning methods also needs to be explored.

References

Akshar Bharati, Ashok Gupta, and Rajeev Sangal. 1995. Parsing with nesting constraints. In *Proc of 3rd NLP Pacific Rim Symposium*, pages 4–6.

Timothy W Bickmore, Daniel Schulman, and Candace L Sidner. 2011. A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology. *Journal of biomedical informatics*, 44(2):183–197.

Dan Bohus and Alexander I Rudnicky. 2003. Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda. *Eighth European Conference on Speech Communication and Technology*.

Myroslava O Dzikovska, James F Allen, and Mary D Swift. 2003. Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains. In *Proc. of IJCAI-03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

Annika Flycht-Eriksson. 2004. Design and use of ontologies in information-providing dialogue systems. *Dissertation, Linkping Studies in Science and Technology, Thesis n. 874, School of Engineering, Linkping University*.

Prathyusha Jwalapuram and Radhika Mamidi. 2017. Keyword and constraint identification for question answering. In *15th International Conference of the*

Pacific Association for Computational Linguistics, Yangon, Myanmar, in print.

Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. 2009. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication*, 51(5):466–484.

Laurent Mazuel and Nicolas Sabouret. 2006. Generic natural language command interpretation in ontology-based dialogue systems. In *Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*, pages 347–350. IEEE Computer Society.

David Milward and Martin Beveridge. 2003. Ontology-based dialogue systems. In *Proc. 3rd Workshop on Knowledge and reasoning in practical dialogue systems (IJCAI03)*, pages 9–18.

A Example Dialogues

System: What would you like me to look for?
Me: I want to try fantasy fiction.
System: I've found Fantasy. I found these about/under Fantasy : *J. K. Rowling* *Christopher Paolini* *George R. R. Martin* *Jonathan Stroud*
Me: Do you have books by Rice?
System: I've found Anne Rice. I found these about/under Anne Rice : *The Vampire Chronicles* What would you like me to look for?
Me: Go back to fantasy.
System: I found these about/under Fantasy : *J. K. Rowling* *Christopher Paolini* *George R. R. Martin* *Jonathan Stroud*

Table 7: Books: Topic Change

System: What would you like me to look for?
Me: I like books which are about vampires.
System: I've found vampires. Would you like Dracula?
Me: I would prefer books which have some mystery in them.
System: I've found Mystery, Thriller, Suspense in the database.

Table 8: Books: Relative Clauses

Me: I like Indian food. Like Chicken Tikka.
System: I found these under Indian: *RESTAURANT-13* *RESTAURANT-22* *RESTAURANT-26* I found these for Chicken Tikka: *RESTAURANT-22* *RESTAURANT-26*
Me: I like Indian food. But I don't like RESTAURANT-22 or RESTAURANT-26.
System: I think these restaurants are relevant: *RESTAURANT-13*

Table 9: Restaurants: Multiple Sentence Input and Rejection

Me: I like Tom Hanks.
System: I think these movies are relevant: *Catch Me If You Can* *Saving Private Ryan*
Me: Recommend some Matt Damon movies. **System:** I think these movies are relevant: *Saving Private Ryan*
Me: I like Hanks. I don't like Damon.
System: I think these movies are relevant: *Catch Me If You Can*

Table 10: Movies: Multiple Sentence Input and Rejection

System: What would you like me to look for?
Me: I want Neil Gaiman and Terry Pratchett books.
System: Do you want an intersection of Neil Gaiman and Terry Pratchett?
Me: Yes.
System: Would you like Good Omens?
System: What would you like me to look for?
Me: I like Jules Verne and Isaac Asimov.
System: Do you want an intersection of Jules Verne and Isaac Asimov?
Me: No.
System: I found these under Jules Verne *Twenty Thousand Leagues Under the Sea* *Around the World in Eighty Days* *Journey to the Center of the Earth* I found these under Isaac Asimov: *I, Robot* *Caves of Steel* *Foundation Series* What do you think? Do you want books from the above?

Table 11: Books: Clarification Dialogue