Game of Characters

Group No: 21

Abhinav Reddy Podduturi, Nimish Jindal, Sushmit Dharurkar, Rongrong Wang

1. Motivation

Sometimes while watching a TV series with a pretty huge star-cast and storyline like “Game of Thrones”, a user may want to learn about the background story of a character, relationship between two characters, get a gist of what’s going on, or to remember what had happened before.

With our system at hand, TV fans don’t need to take the time to open the browser, type in the question “Who are Arya’s siblings”, or “Who’s Benjen Stark?” and browse the results one by one filtering out all the irrelevant contents. Instead, they only need to ask the Echo device out loud, saying “Who is Tywin Lannister to Tyrion Lannister?” and the simple and accurate answer, “Parent”, will be returned immediately.

It is not only fun to use, fast and accurate, but also the first dialog system on TV show storyline queries.

2. Background

Essentially what we wish to achieve can be broken down into two simple thoughts -

1. dialogue system should be able to respond to any complex human query.
2. dialogue system should be portable i.e., could be easily used for any other TV series or a general search application as well.

Both our thoughts somewhat relate to some existing use cases or a mixture of these. For example, our first thought can related to a combination of Siri’s wit and persona along with google’s search abilities. On the other hand, our second thought arise out of reducing the effort of making a new dialogue system for every application that exist. Imagine how many applications exist and imagine writing a dialogue system for each! The need for portability is not something new to the computing world. In fact, it lies at the base of World Wide Web, and the underlying idea which it entails, serves as a useful input for solving our portability challenge.

Before the World Wide Web, when a user wanted to use an Internet application, he would install a tool capable of handling specific types of network messages on his machine. For example, if a user wanted to locate users on other networks, he would install an application capable of

utilizing the FINGER protocol. If a user wanted to exchange email across a network, he would install an application capable of utilizing the SMTP protocol. At that time, to write a network application a developer was required to write the communication protocol, besides writing the application logic; and user was required to required to install the communication protocol to start use the app.

The emergence of the Web represented a radical change in how most people used the Internet. The Web shielded users from having to think about the applications handling the Internet messages. All you have to do is install a web browser on your machine, and any application on the Web is at your command. For developers, the Web provided a single, simple abstraction for delivering applications. This standardized infrastructure allows web application developers to focus on how their applications appear to users rather than how application data is transmitted between machines.

At the core, a web browser decouples applications from data through the use of a simple, abstract model(knowledge representation). The model provides constraints(HTTP protocol) on both applications and data, allowing both to evolve independently. This degree of application-data independence promotes “portability”. Any application that understands the model can use the data source.

3. Knowledge Representation

So, from discussion in previous section, it is clear that to design a dialogue system which can be used for multiple applications, the underlying representation of data source should be same across applications. In view of our first thought, the data source of our applications is world wide web itself!

Information on a web page is arranged in many different ways such as a table, an outline, or a multi-page narrative. Currently, we are able to extract the important information and use it to guide further knowledge discovery because we as humans are very flexible data processors. However, this heterogeneity of information would be indecipherable to machines, if they are left to find knowledge from it - all from themselves!

If the diversity of information present on the Web is made available as “semantic data structures” by the web developers, any “application” could access and use this information rich data.

Now what this semantic data structure should look like? We already have many trivial and advanced ways of representing knowledge, from storing in tables across spreadsheets, to storing data a relational database. Given the heterogeneity and amount of data present on the web, these methods cannot be used to represent semantics of the web. For example, merging two relational databases, or even a single table into an existing database involves a lot of issues, namely schema migration. A whole discipline of software engineering has emerged to deal with these issues, using techniques like object-relational mapping (ORM) layers to try to

decouple the underlying schema from the business-logic layer and lowering the cost of schema changes. These techniques are useful, but they impose their own complexities and problems as well.

When dealing with data integration across the entire Web, or even in smaller environments that are constantly facing new datasets, migrating the schema each time a new type of data is encountered is simply not tractable.
This brings us to the question of whether it's possible to define a schema which is flexible enough to handle a wide variety of ever-changing types of data, while still maintaining machine-readability. The schema should such that it easily incorporates addition of new entities and properties of the entities. The answer lies in the three-column format also known as a triple, which forms the fundamental building block of semantic representation. Each triple is composed of a subject, a predicate, and an object. One can think of triples as representing simple linguistic statements, with each element corresponding to a piece of grammar.

This key-value type of representation is not new, and is in fact used by some companies to represent data internally. However, this type of representation frequently hampers database performance(because it ignores the concept of database normalization), and therefore it is generally not considered a best practice for every database application. Given the requirements of our system, key-value representation would be fast and extremely flexible to integrate new information, whether it's about same TV series(Game of thrones) or a different TV series which we may want to add in future.

Once we have achieved the semantic data modelling, a well-designed application would be able to seamlessly integrate new semantic data, and moreover, semantic datasets would be able to work with a wide variety of applications.

4. Data Collection and Preparation

We crawled GameOfThrones site (http://gameofthrones.wikia.com/wiki/Game_of_Thrones_Wiki) for data and stored the information in three column format. All the information stored in three column format is combined to form a directed graph R(Figure-2).

5. Dialogue System Architecture

Our system is using frame-based, mixed initiative dialogue manager, where the NLU and the ASR modules are handled by Alexa in the cloud. Users can ask questions about any character's parents, siblings, spouses and grandparents, or any general information of a character stored in the database. The diagram of dialogue system architecture is shown as below.

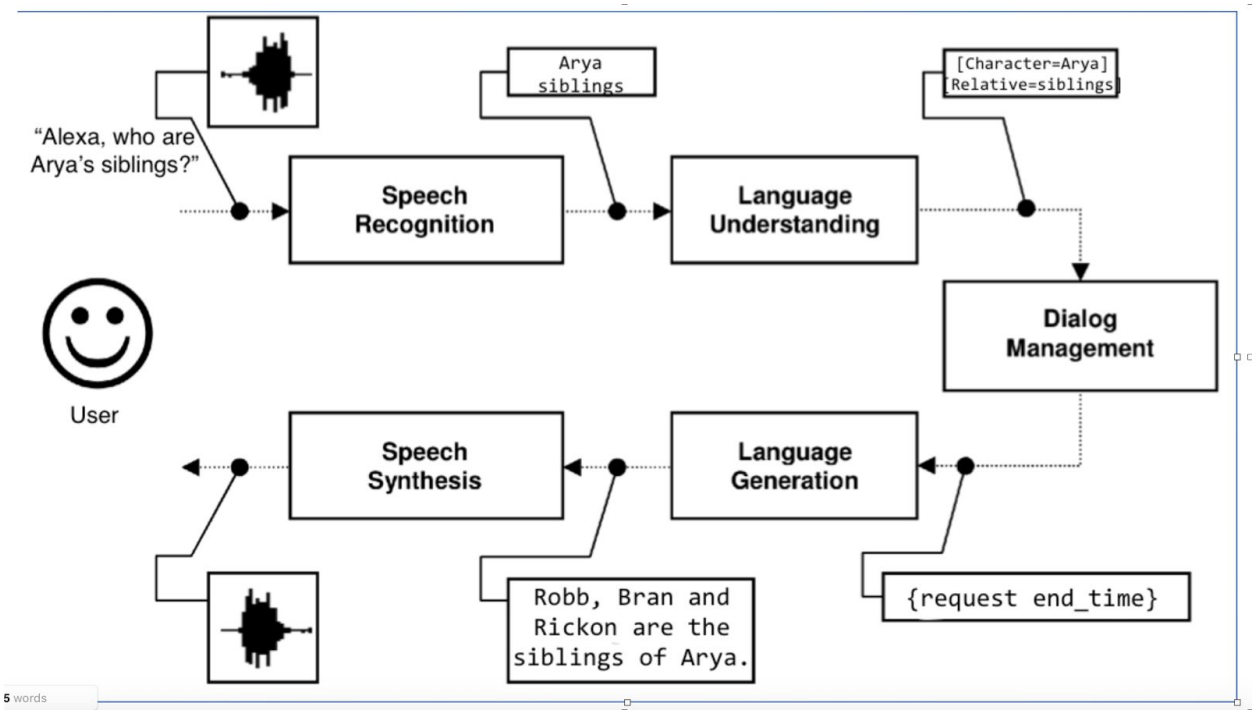


Figure 1. Diagram of dialogue system architecture

6. Dialogue Manager

In the Alexa Skills toolkit, we are using frame-based approach for the dialogue manager, the reasons for doing this is that frame-based DM has higher ASR results, it monitors the dialogue while filling in each slot, and refers to that slot for information. Also, as frame-based DM has no absolute sequences for which slot to fill first, users may ask questions in whatever order they like, or contain multiple concepts (names, relation) in one turn.

The dialogue manager decides what to do next after receiving and interpreting the users input, it will ground inputs like names of the characters and output the information of the stories and character relationships that has been queried.

First we specified a mapping schema between users' spoken input and the intents, which represent actions that fulfills a user's spoken request. There are currently four different intents in our system: 'WhoIsX', 'CharactersRelative', 'RelationOfXandY' and 'WhoPortrayedX'.

For each intent there are:
(1) Intent property that gives the name of the intent, like the 'CharactersRelative'.

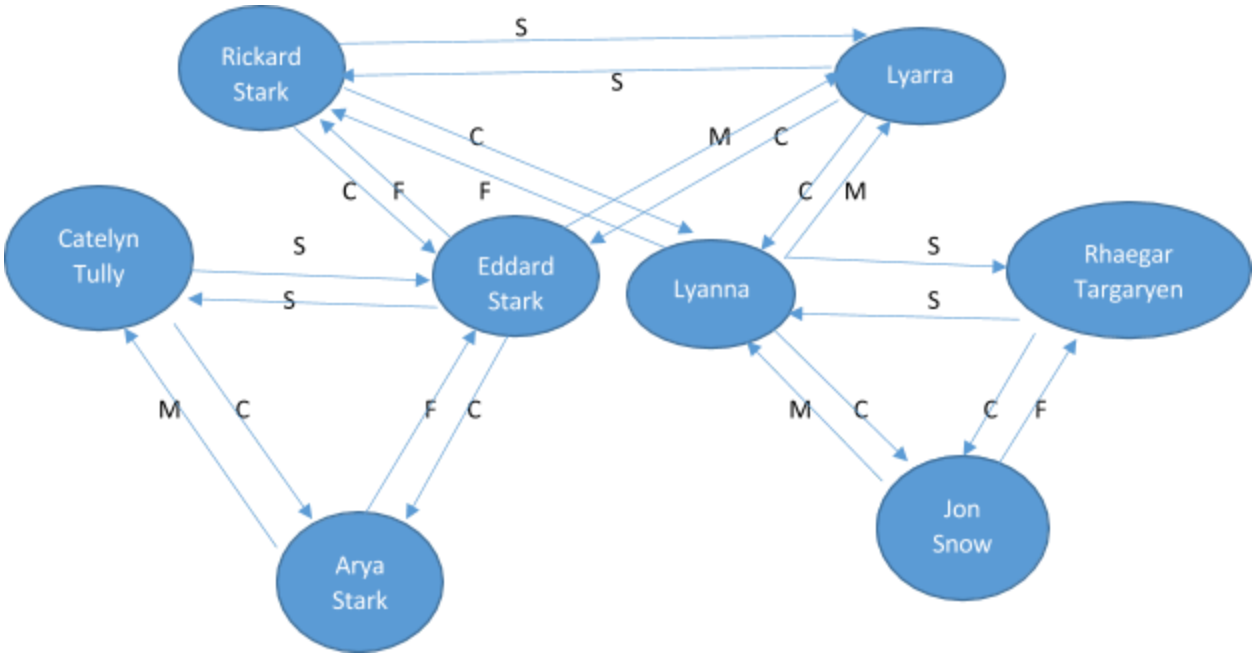
(2) And the slots property that lists the slots associated with that intent, like the ‘Character’ slot and the ‘Relation’ slot.
When a user says “Alexa, who are Tyrion’s parents?”, and the Alexa service sends the ‘CharactersRelative’ intent with the value ‘Tyrion’ in the ‘Character’ slot and ‘father’ in the ‘Relation’ slot in written format. On receiving the ‘CharactersRelative’ intent, the system can then look up the list of characters for the word ‘Tyrion’ and the list of relations for the word ‘parents’, then sends back text to convert to speech.

There are also different types defined in each slot. For instance, the ‘Character’ slot is defined as type ‘LIST_OF_CHARACTERS’, in order for the system referring to a list of character name values provided for the slot (Tyrion, Arya, etc.), and the slot type can be used for multiple slots, like the type ‘LIST_OF_CHARACTERS’ can also be used in ‘WhoIsX’, ‘RelationOfXandY’ or ‘WhoPortrayedX’ slots.

The syntax of the DM that contains the intent and slots are shown as below:

```
"intent": "CharactersRelative",
"slots": [
  {
    "name": "Character",
    "type": "LIST_OF_CHARACTERS"
  },
  {
    "name": "Relation",
    "type": "LIST_OF_RELATIONS"
  }
]
```

7. Processing User Query



Sample subgraph of R

Figure-2 Sample subgraph of R

Definitions:
SAME_LEVEL:
{siblings, spouses, cousins, siblings-in-law, cousins-in-law}
HIGHER_LEVEL:
{grandparent, great grandparent, great great grandparent, great ….. great grandparent}
ANSWERABLE_RELATIONSHIPS:
{parents, children, siblings, cousins, grandparents,
great grandparents, great …... great grandparents}

R is the graph formed by combining all the collected data.
There are four important types of edges in the graph R {F,M,C,S}, F-father, M-mother, C-child, S-spouse. Each type of edge is assigned a value. F and M are assigned a weight of +1, weight of 0 for S and weight of -1 for edge C.

Our system is currently answering four types of queries.

1) How are X and Y related?
2) Who is the X of Y?
3) Who portrayed X?
4) Who is X?

Below we have described the algorithms that were used on graph R to answer queries.

Type one query: How are X and Y related?
Let the query be to find relation between two characters X and Y. We will start a breadth first search in the relationship graph R and find shortest path from node X to node Y where node X represents character X and node Y represents character Y. We will get a path P between X and Y.
Now P contains a sequence of relations with which the characters are connected. An example path P can be {F,M,C,C}. P must be compressed into a single relationship. This is done by following the below logic. Net weight of path P is calculated by adding the edge weights of all edges in P.

a) If net weight is equal to zero then X and Y must be related in any one of the ways in SAME_LEVEL. The exact relationship from SAME_LEVEL is found by checking edges in P for each case.

b) If net weight of path P is greater than zero then we use the following algorithm. If net weight is equal to one then Y must be a parent of X or we find relationship between X's parent and Y using the algorithm presented in a). If net weight of P is greater than one, then Y must be related to X in any one way among HGHER_LEVEL. Number of times “great” is perpended to “grandparent” is decided based on the value of
(net weight of P) - 2.

c) If level is less than zero then X and Y are swapped and relationship between X and Y is found by using the algorithms described in a) and b).

Type two query : Who is the X of Y?
Here X is any relationship from ANSWERABLE_RELATIONSHIPS and Y is any character.
We can directly get information about parents, spouses and children from graph R by traversing the directed edges from node Y. To find siblings of Y, we first find parents of Y and then we find children of Y's parents. To find grandparents of Y we first find the parents of Y and then find the parents of Y's parents. To find great grandparents we find Y's grandparents parents. To find cousins of Y we find all grandchildren of Y's grandparents who are not Y's siblings.

For type 3 and type 4 queries, required information is collected from graph R and answers are generated using templates.

8. Technologies Used

The system is using Alexa Skills Kit, it is Amazon's intelligent voice recognition and natural language understanding service. The major components of the dialogue system, like automatic speech recognition (ASR) and natural language understanding (NLU), are handled by Alexa in the cloud. It also recognizes and responds to user’s voice requests instantly, greatly improve the quality of the user experience.

We built the system for an Alexa skill using AWS Lambda, which runs the code only when it's needed and scales automatically, so there is no need to continuously run servers. We used Python to collect data from a set of data links, like extracting each character's information, and we also built a linked list for the relationship tree among all the characters. To apply the code into Alexa, we uploaded the code into Lambda function, which executes the code and responses to the Alexa's voice interactions, it will also automatically manage the compute resources.

9. Context For Use

As some of dialogue system components like Automatic Speech Recognition and Natural Language Understanding are handled in the cloud, the system can be implemented in any Alexa-enabled device that has a microphone and speaker, as long as it connects to the internet and the Alexa hardware. It can be used in a laptop, a smart phone, or even a car.

10. Capability & Future Scope

Currently, the system mainly answers for 4 kinds of questions:
(1) Backstory of a character X by user asking, "Who is Arya?"
(2) Or the relationship between 2 characters X and Y by user asking, "What's the relationship between Ned Stark to Jon Snow?"
(3) Or questions about a character's relative: "Who are Tyrion's parents?"
(4) Or the name of the actor who portrayed a role, by user asking, "Who portrayed Benjen Stark?"

The system supports up to 1155 characters that currently appeared in the Game of Thrones series, users may get generic information about the characters' age, status, which season they first appeared in by asking the name of them. Also, the relationship of each connected character like their children, spouse and parents can also be obtained by querying the system.
As the system contains more detailed profiles of each character, like the religion and origin, than just their names and relationships, it could be further developed to add more types of queries, like things the characters were doing at some location at certain period of time, or the particular feelings one character holds toward another.

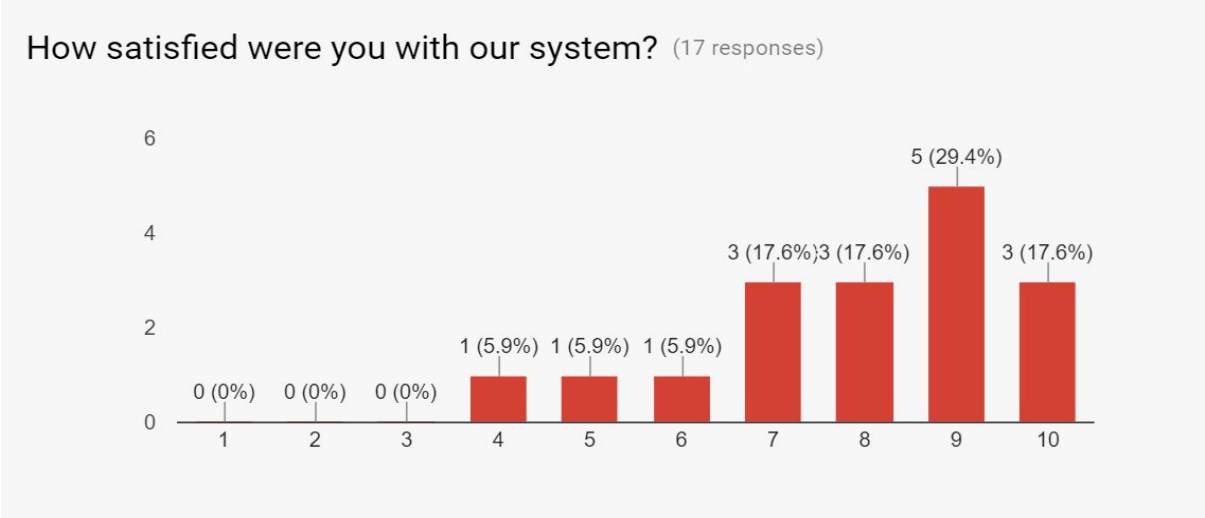
In the future, our system could be expanded to cover more than one TV series, like the Arrow, Breaking Bad, etc. And, for those shows that are not ended yet, we will be updating the system to add newly appeared characters and stories to ensure the database is up-to-date.

11. User Surveys

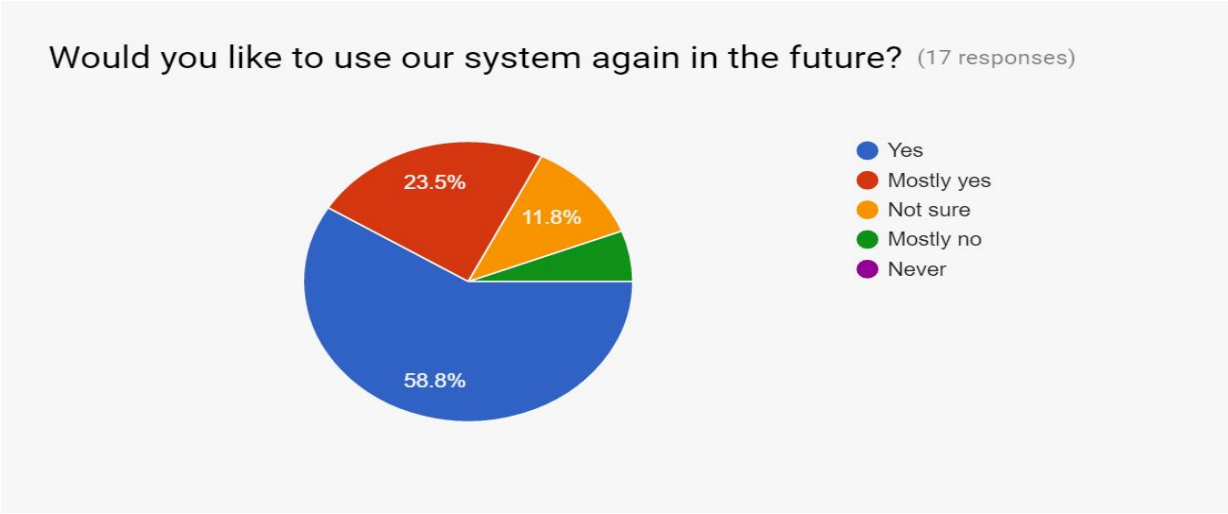
A total of **17 users** used our system. We used Google Forms to do the user surveys. These are the responses which we got from our user surveys.

The first question which we asked the user was, 'How satisfied were you with our system?'. 1 being not very satisfied and 10 being very much satisfied. Here we can see, most of the users

were satisfied by using our system. Around **82.2%** users gave us the rating of 7 or higher and **17.7%** users gave a rating below 6 or below.



The second question which we asked the user was, 'Would you like to use our system again in the future?'. The response of this question was also a positive one. Around **58.8%** users answered **'yes'** to this question, **23.5%** users answered **'mostly yes'** and 11.8% users were 'not sure'. Only 5.9% users answered 'mostly no' and **none** answered **'never'** to this question.

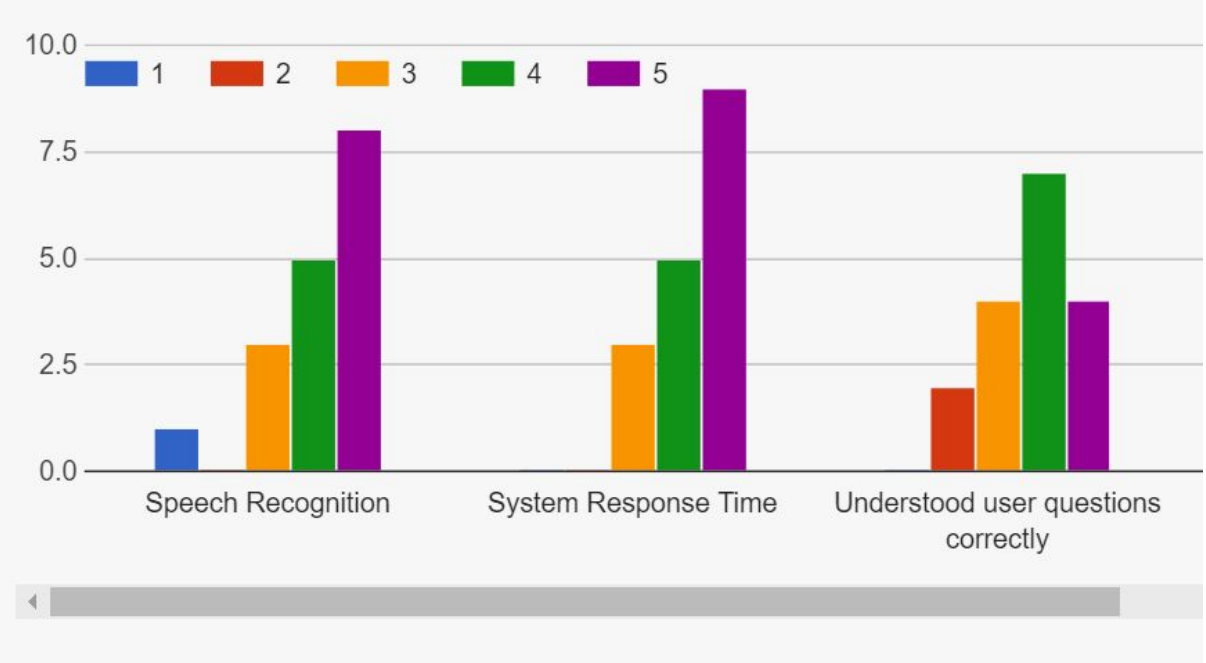


We then asked the user, how satisfied were they with the individual aspects of our system like the Speech Recognition, System Response Time, Understood user questions correctly, Responded with correct answers, Prompts given to assist users and Ease of Use. We used a scale of 1 to 5 to rate each aspect. 1 being very dissatisfied and 5 being very satisfied.

- **Speech Recognition:**
  Speech Recognition means how well the system got the words correctly, what the user was saying. **76.5%** users gave speech recognition a rating of 4 or higher, whereas **23.5%** users gave speech recognition a rating of 3 or lower.

- **System Response Time:**
  Speech Response Time means how quickly could the system answer to the question asked by the user. **82.3%** users gave system response time a rating of 4 or higher, whereas **17.6%** users gave system response time a rating of 3.

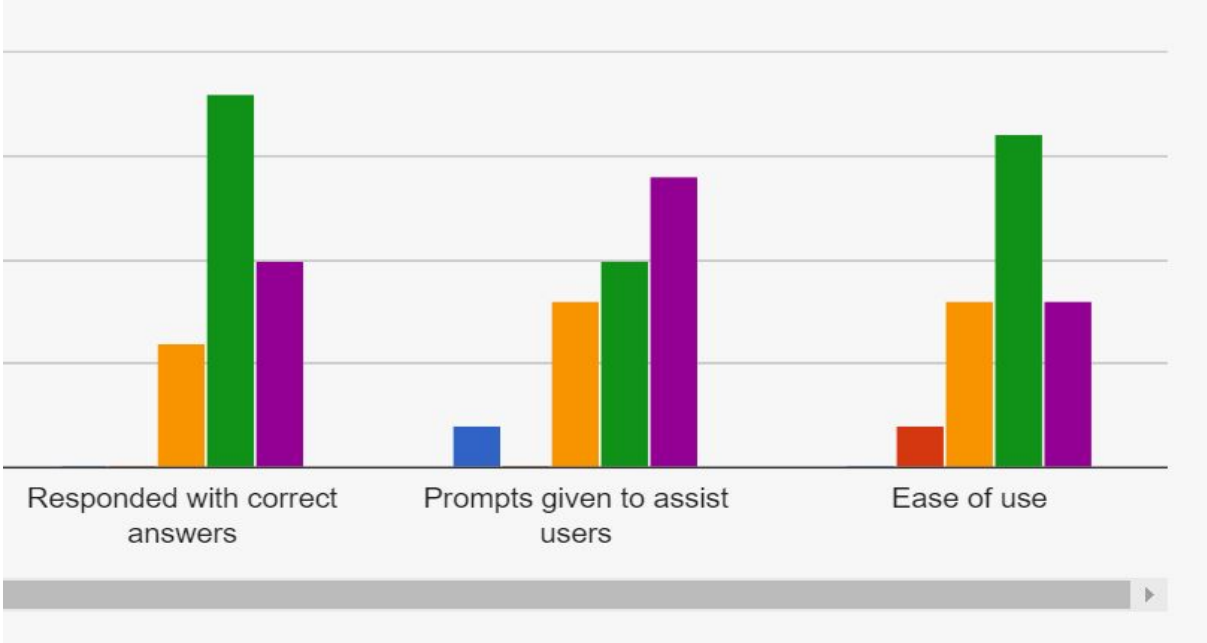
- **Understood user questions correctly:**
  This means how well did the system understand exactly what the user wanted to know. **64.7%** users gave this a rating of 4 or higher, whereas **35.3%** users gave this a rating of 3 or lower.



- **Responded with correct answers:**
  This means how well was the system able to provide the correct answer to the question asked by the user. **82.3%** users gave this a rating of 4 or higher, whereas **17.6%** users gave this a rating of 3.

- **Prompts given to assist users:**
  This means the responses or messages given by the system to help the user in completing his required task. **70.6%** users gave this a rating of 4 or higher, whereas **29.4%** users gave this a rating of 3 or lower.

- **Ease of use:**
  Ease of use means how easy was it to interact with the system and get the task completed. **70.6%** users gave this a rating of 4 or higher, whereas **29.4%** users gave this a rating of 3 or lower.



Here are a few user comments which we got:

- awesome that you can recognize game of thrones names. that seems hard!
- I was not familiar with the system context but I think it would be a nice system in terms of functionality for their curious about the characters.
- awesome application . adding more characters and questions might be helpful

12. Regression Model

We built a multiple regression model for user satisfaction using four variables 1) accuracy of speech recognition (ASR), 2) system response time(SRT), 3) Understood user questions correctly(U), 4) answered user questions correctly(A)

The multi regression model for user satisfaction US is

US = 3.3236 + ASR*0.2534 + SRT*-0.3230 + U*0.5423 + A*0.7430

User satisfaction is much more affected by fourth parameter i.e. A. So to improve user satisfaction we need to improve the accuracy with which our system responds to queries and decrease the response time.

13. Conclusion

In this way, we have built a dialogue system on Alexa which can answer various questions related to the TV series Game of Thrones. The results which we got from the user surveys are very promising. Most of the users were satisfied after using our system and also considered of using it in the future. Our system currently supports only 4 questions, but support for additional questions can be easily added. In the future, we are also thinking about support for additional TV shows.