Chatbots

SHRIMAI PRABHUMOYED 
ALAN W BLACK 
SPEECH PROCESSING 11-[468]92
Overview

- Chatbots
- Task Oriented
- Non-Task Oriented Dialog Systems
- Building Dialog Systems
  - Retrieval Based
  - Similarity Metric
  - Generative models
Chatbots

- Designed to simulate how a human would behave as a conversational partner, thereby passing the Turing test.
- Chatbots are used for various practical purposes like customer service, personal assistants or information acquisition.
Chatbots

- **Personal Dialog Assistants**
  - Siri, Alexa

- **Helpline Chatbots**
  - Hotel booking, airline reservation

- **Conversational bots**
  - Zo, Tay, Xiaoice, Facebook M
Eliza was based on keyword matching.

Parry was Eliza with an attitude.
Aspects to think about
Aspects to think about

- **Persona**
  - voice, age, gender, background
- **Domain**
- **Scenarios it can handle**
- **Response for other scenarios**
- **Variation in input and output**
- ...
Chatbots - Types

Task Oriented

• Clear and explicit intentions
• The system should have the capability to process the intents.

Non-Task Oriented

• No particular agenda
• “What’s up”, “How are you doing?”
Task Oriented

- **Intents**: actions that the user wants the system to perform
- **Slots**: arguments you need to fulfill the intent
- **Actions**: system performs the task
  - Eg: “Adds a meeting to your calendar”
- **Responses**: the utterance said by the system
  - Eg: “Meeting has been added”
**Task Oriented - Example**

- **User:** I want to add one reminder to switch off the light.
- **System:** Intent → Add Reminder
- **System:** Slot → date, time and purpose
- **System:** When would you like to set the reminder?
- **User:** In an hour
- **System:** Slot → 12 Sept at 4.30pm
- **System:** Action → Set a reminder at 4.30pm on 12 Sept to switch off the light
- **System:** I have set a reminder to switch off the light in an hour.
**Task Oriented - Example**

- **User:** I am getting late to class, so I need the next bus to CMU. I want to reach as fast as possible.

- **System:** Intent → to go

- **System:** Slot → date, time, departure_stop, arrival_stop (today, in the next 10 minutes, -, CMU)

- **System:** Where would you like to leave from?

- **User:** I live on Whitman and bartlett

- **System:** departure_stop → whitman and forbes (Wh-Fo)

- **System:** Action → Find the next bus from Wh-Fo to CMU

- **System:** The next bus is 61A in 12 minutes from Wh-Fo
Task Oriented

**Evaluation Metric:**
- Task completion success
- Eg: The system was able to give the correct bus information, the system was able to set an alarm for the right time.
- Length of dialog
- Eg: The system should not ask too many questions, too many repetitions etc
User: I am getting late to class, so I need the next bus to CMU. I want to reach as fast as possible.

System: Intent → to go

System: Slot → date, time, departure_stop, arrival_stop (-, -, -, CMU)

System: What date would you like me to check the bus for?

User: Today Sept 12!

System: When would you like to leave?

User: In the next 10 min!!
Non Task Oriented

- **User1**: Hey, what’s up?
- **User2**: Nothing much. What about you?
- **User1**: Nothing, just the usual hectic life of CMU.
- **User2**: Yes, it’s been so long since we caught up. We should maybe grab some coffee sometime...
- **User1**: Ya, that’s true. We should definitely meet up sometime. Ok, I need to run for a class, ping me!
- **User2**: Sure, bye!
Non Task Oriented

- **User1**: Hey, have you seen the new Kingsman movie?
- **User2**: No, what’s it about?
- **User1**: It’s a science fiction thriller movie. Do you want to go watch it sometime?
- **User2**: Ya sure, I like sci-fi movies.
- **User1**: Let’s watch it over the weekend
- **User2**: Ok 😊
Non Task Oriented

- Intents and slots are hard to design
- Can have multiple responses
- Evaluation:
  - Engagement
  - User satisfaction
  - Length of dialog
  - ...
Building Chatbots

► **Retrieval Techniques**
  - Used very often to build helpline chatbots.
  - Examples: “How do I install Ubuntu on my machine?”, “I cannot connect to network. How can I connect to wifi?”

► **Generative Models**
  - Used very often to build conversational chatbots.
  - Example: “How are you doing?”, “Can you tell me a secret?”

► **Hand – Written Rules**
  - Used very often to build some aspects of personal assistants. Eg: “Add ‘Meet Alan’ in my Calendar”
Retrieval Techniques

- Fixed set of query-response pairs in the database.
- Representation of the query and the database.
- Metric to compare and evaluate the best fitting response.
Representation

- Words themselves!
- N-grams
  - Unigram: $P(w)$
  - Bigram: $P(w_1, w_2)$ such as $P(“I”,”am”) and $P(“I”, “is”)$
  - Trigram: $P(w_1, w_2, w_3)$
- Term Frequency – Inverse Document Frequency (Tf-Idf)
- Word Vectors
TF-IDF

Term Frequency (TF): measures how frequently a term occurs in a document. The term frequency is often divided by the document length.

\[
tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}
\]

Inverse Document Frequency: measure of how much information the word provides, that is, whether the term is common or rare across all documents.

\[
idf(t, D) = \log \frac{N}{1 + |\{d \in D : t \in d\}|}
\]
Document (d) \( \rightarrow \) 100 words, term “dog” appears 5 times in d.

\[
\text{tf} \left( \text{“dog”}, d \right) = \frac{5}{100}
\]

Suppose, \( D = 10 \) million and “dog” appears in 1000 of them

\[
\text{idf} \left( \text{“dog”}, D \right) = \log \frac{100000000}{1000} = 4
\]

Tf-idf score: \( 0.05 \times 4 = 0.12 \).

For dialog system, we consider the entire database of “query-response” pairs as one document.
## TF-IDF Representation

### Vocabulary Table

<table>
<thead>
<tr>
<th>Vocab</th>
<th>Tf-Idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;the&quot;</td>
<td>0.8</td>
</tr>
<tr>
<td>&quot;dog&quot;</td>
<td>0.3</td>
</tr>
<tr>
<td>&quot;and&quot;</td>
<td>0.5</td>
</tr>
<tr>
<td>&quot;play&quot;</td>
<td>0.6</td>
</tr>
<tr>
<td>&quot;UNK&quot;</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### Representation of the input

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>dog</th>
<th>and</th>
<th>the</th>
<th>cat</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0.8</td>
<td>0.3</td>
<td>0.5</td>
<td>0.8</td>
<td>0.1</td>
<td>0.6</td>
</tr>
</tbody>
</table>

### Limitations
- Cannot work for synonyms
- Does not take context into account
Similarity Metric

- Jaccard Similarity Coefficient

\[ J(A, B) = \frac{|A \cap B|}{|A \cup B|} \]

- measures similarity between finite sample sets
- \( 0 \leq J(A, B) \leq 1 \)
Similarity Metric

- Cosine Similarity

\[ \cos(\theta) = \frac{A \cdot B}{||A||_2 ||B||_2} \]

- Here, \( A \) = representation of the input and \( B \) = representation of the query in the database.
- For each query in the database, we calculate these scores and select the query which has max score.
- We return the response of this query.
Complete Example

“How can I connect to WiFi”
“Go to Settings → Wifi. Select ...”

“How do I install Ubuntu 16.04”
“Download Ubuntu image ...”

“How can I install Java”
“Download the jdk ...”

“Which NVIDIA driver do I need for GTX 1080 Ti”
“sudo apt install nvidia-381”

<table>
<thead>
<tr>
<th>Vocab</th>
<th>Tf-Idf</th>
<th>Vocab</th>
<th>Tf-Idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>How</td>
<td>3/22*\log(1/2)</td>
<td>Java</td>
<td></td>
</tr>
<tr>
<td>can</td>
<td></td>
<td>Which</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td>NVIDIA</td>
<td>driver</td>
</tr>
<tr>
<td>connect</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td></td>
<td>need</td>
<td></td>
</tr>
<tr>
<td>Wifi</td>
<td></td>
<td>for</td>
<td></td>
</tr>
<tr>
<td>do</td>
<td></td>
<td>GTX</td>
<td></td>
</tr>
<tr>
<td>install</td>
<td></td>
<td>1080</td>
<td></td>
</tr>
<tr>
<td>Ubuntu</td>
<td></td>
<td>Ti</td>
<td></td>
</tr>
<tr>
<td>16.04</td>
<td></td>
<td>UNK</td>
<td></td>
</tr>
</tbody>
</table>

Total Query Words = 22
Complete Example

Input Representation

<table>
<thead>
<tr>
<th>How</th>
<th>do</th>
<th>I</th>
<th>connect</th>
<th>to</th>
<th>WiFi</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How</th>
<th>can</th>
<th>I</th>
<th>connect</th>
<th>to</th>
<th>WiFi</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How</th>
<th>do</th>
<th>I</th>
<th>install</th>
<th>Ubuntu</th>
<th>16.04</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.2</td>
<td>0.4</td>
<td>0.35</td>
<td>0.1</td>
<td>0.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How</th>
<th>can</th>
<th>I</th>
<th>install</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.3</td>
<td>0.4</td>
<td>0.35</td>
<td>0.15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Which</th>
<th>NVidia</th>
<th>driver</th>
<th>do</th>
<th>I</th>
<th>need</th>
<th>for</th>
<th>GTX</th>
<th>1080</th>
<th>Ti</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.1</td>
<td>0.06</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Limitations of Retrieval Systems

- We have a constrained set of responses.
- No variance in the response.
- Cannot handle novel queries.
Summary

- Task Oriented
  - Intents, Slots, Responses. Evaluation by task completion.
- Non-Task oriented
  - Intents and evaluation are hard to define.
- Retrieval Techniques
  - TF-IDF representation and cosine similarity
- Limitations of Retrieval Techniques
Generative Models

Next Class!