Agenda

Topic Summary
Design and Operation
  Sequential
  Parallel
Demonstration
Questions
Project Topic: Conversational “Chatbot”

Our Goal

Generate “intelligent” conversational responses to user input without relying on selecting random numbers, and instead exhaustively or heuristically search the trained vocabulary for the best response.

Parallel Programming

Write the portion of the program that deals with finding a response in parallel to increase response time (strong scaling) or increase the search space (weak scaling)
High Level View

Inputs
- Number of Responses to Evaluate
- Input Prompts
- Training Conversations

Vocabulary Population

Responses
- Response 1
- Response 2
- Response 3
- ... (up to Response N)

Output Response
Towards a method for evaluating naturalness in conversational dialog systems

Author: Victor Hung, Miguel Elvir, Avelino Gonzalez, Ronald DeMara


Date: 11-14 Oct. 2009

Pages: pp111

URL: http://ieeexplore.ieee.org/document/5345904/
Problem addressed by this paper

- Focus is determining naturalness
  - How well a chatbot can maintain a natural conversation flow
- Quantitative methods can not be the only metric in assessing performance
- There is a great deal of subjectivity in assessing the performance of conversation bots
- There is no general method for judging how well it performs
Applications

Functional

• Customer Service
• Tool for automating part of a business
• Weather, News, and Daily Event reminders
• Grocery bot
• Scheduling bots
• Clevertweet

Fun

• Have a conversation
  • Some applications are of specific people, like the William Shakespeare bot.
• Advice, fortunes, fun facts
Paper 2
Making a clever intelligent agent: The theory behind the implementation

Author: Raine, R.
Date: 20-22 Nov. 2009
Pages: 398 - 402
URL: http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=5358137
Fuzzy logic in natural language processing

Author: Vilém Novák


Date: 9-12 July 2017

Pages: 1-6

URL: http://ieeexplore.ieee.org/document/8015405/
How We Will Use This

- Utilize some elements of FTT for classifying words
- While we might not separate nouns and adjectives, we might use other characteristics such as usage percentage, word structure, or location within sentences
Program Start
Arguments:
- Number of responses to test per prompt (N)
- File containing prompts
- Files containing training data

Read files
Populate ‘Words’ object (the program’s known language)
Repeat for every prompt (simulate a conversation):

Initialize a reduction variable
For each value of N (sequential or parallel):
Generate the response that corresponds to the provided value of N
Calculate the fitness of the response based on heuristics
Reduce with the reduction variable (or a thread-local reduction variable)
Print best (reduced) response
Program End
Clever Sequential

• Read inputs
  → Number of responses to test
  → Input prompts
  → Training files

• For each of the input prompts
  - Make a heuristic reduction variable
  - For the number of responses to test
    - Compute weights
    - Generate response
    - Reduce

• Print response
Clever Parallel

• Read inputs
  → Number of responses to test
  → Input prompts
  → Training files

• For loop over input prompts
  - Make a heuristic reduction variable
  - Parallel for the number of responses to test
    Make thread local heuristic reduction variable
    - Compute weights
    - Generate response
    - Reduce

• Print response
‘Words’ Class

Represents the program’s language.

Populated once at the start of the program

Input:
Hello, how are you?
How are you today?
How is your day going?

<table>
<thead>
<tr>
<th>Key</th>
<th>Values</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>hello</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>how</td>
<td>2</td>
</tr>
<tr>
<td>hello</td>
<td>how</td>
<td>1</td>
</tr>
<tr>
<td>how</td>
<td>are</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>is</td>
<td>1</td>
</tr>
<tr>
<td>are</td>
<td>you</td>
<td>2</td>
</tr>
<tr>
<td>you</td>
<td>END</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>today</td>
<td>1</td>
</tr>
<tr>
<td>today</td>
<td>END</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>your</td>
<td>1</td>
</tr>
<tr>
<td>your</td>
<td>day</td>
<td>1</td>
</tr>
<tr>
<td>day</td>
<td>going</td>
<td>1</td>
</tr>
<tr>
<td>going</td>
<td>END</td>
<td>1</td>
</tr>
</tbody>
</table>
Sentence Generation

Each sentence can be generated from an integer or hash (n)

- Start with START
- Next word is found in the values for the current word/key at position n%(# of values).
- Change n to n/(# of values)
- Repeat until END or 25 words

<table>
<thead>
<tr>
<th>Key</th>
<th>Values</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>hello</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>how</td>
<td>2</td>
</tr>
<tr>
<td>hello</td>
<td>how</td>
<td>1</td>
</tr>
<tr>
<td>how</td>
<td>are</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>is</td>
<td>1</td>
</tr>
<tr>
<td>are</td>
<td>you</td>
<td>2</td>
</tr>
<tr>
<td>you</td>
<td>END</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>today</td>
<td>1</td>
</tr>
<tr>
<td>today</td>
<td>END</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>your</td>
<td>1</td>
</tr>
<tr>
<td>your</td>
<td>day</td>
<td>1</td>
</tr>
<tr>
<td>day</td>
<td>going</td>
<td>1</td>
</tr>
<tr>
<td>going</td>
<td>END</td>
<td>1</td>
</tr>
</tbody>
</table>
Weighted Heuristics

Weight of a response =

\( (0.2 \times \text{Sentence weight from Words}) \)
+ \( (0.35 \times \text{Relevance weight}) \)
+ \( (0.1 \times \text{Word mixture weight}) \)
+ \( (0.15 \times \text{Sentence length weight}) \)
+ \( (0.2 \times \text{Usage mixture weight}) \)
Heuristic Class

• Calculate relevance
  - Based on the number of shared words
• Calculate word mix
  - Higher weighting for a broader mix of long and short words
• Calculate sentence length weight
  - Based on how similar in length the prompt and response are
• Calculate word usage mix
  - Higher weighting for a broader mix of common and uncommon words
Heuristic Reduction Variable Class

implements Vbl

• Fields:
  → response
  → fitness

• Get response (to print after reduction)
• Reduction:
  → Take the new response and fitness if the new fitness (weight) is greater than the current fitness
Parallel Programming

Write the portion of the program that deals with finding a response in parallel to increase response time (strong scaling) or increase the search space (weak scaling)

• Strong Scaling
  □ Increase the number of cores

• Weak Scaling
  □ Potential variables that control the problem size
    1  Number of responses to evaluate ✔
    2  Number training documents → Sequential overhead
    3  Number of prompts → Must be sequential in a chat scenario
Demonstration
Questions?