Team DaC

“Divide and Conquer”
John Grosh & Chris Valant
Agenda

Introduction

Topic and Goals

Topic Aspects

Sequential & Parallel Component

Questions
Team DaC

John Grosh
• Computer Science BS
• Writes software for online chat services
• Interested in automated human-computer interaction

Chris Valant
• Physics BS
• Currently in Data Science MS
• Interested in Artificial Intelligence and Machine Learning
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)
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
Overview of Aspects

Natural Language Processing

Artificial Conversation

Real-time responses

Human-like responses
Natural Language Processing (NLP)

A way for computers to understand human language in a useful way.

Common aspects of NLP:

• Vocabulary - All words used in the process.
• Stop-words - Words like “and”, “the”, “a”, “to”, “how”, “or”, etc.
• Stemming - Reduce words to their root, or stem.
Artificial Conversation

Rule based

• Responds to specific commands.
• Will not know how to respond to anything else.

Machine Learning

• Understands language.
• Don’t have to be specific.
• Can be trained on the conversations it has to become smarter.
Real-time Responses & Human-like Responses

• Normal human responses are on the order of seconds.
• Accomplished by making the search for a response the parallel part of the program.

Two parts:

• Traverse a tree to generate responses
• Evaluate the responses and select the best one
  • Does it make sense?
  • Does it answer the question?
The Sentence Generation Tree

Sequential Program

Depth-first sequentially

Parallel Program

Breadth-First sequentially

Depth-First in parallel
Generating the Tree

**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>today</td>
<td>END</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>
Evaluating a candidate response

Heuristic to determine the quality of a response

• Related words
• Common sentence structure
• Sentence length

In the parallel program, each thread will keep track of its own ‘best’ response according to the heuristic, and then the results will be merged together to find the absolute best.
Sequential Components

• Training
  • Example conversation data from online chat communities
  • Possible example sentence structure and vocabulary from published literature if a broader tree is needed for better results

• I/O for what the user says, and what the bot replies
  • “REPL”-style (Read-Eval-Print-Loop)
Parallel Component

• Generating a response
  • Training will generate a tree of word associations.
  • Evaluation will do an exhaustive search of the tree.
  • Heuristic searches can be done if the tree is very large, which will depend on the size of the vocabulary.
  • We will start with exhaustive searches and move to heuristic if the problem size is unreasonably large.
Questions?