

Regex Chatbot for Detective Novels

Team: Anirudh Gajjala, Bryson Gullett,
Oleksiy Selyutin, Anderson Walsh



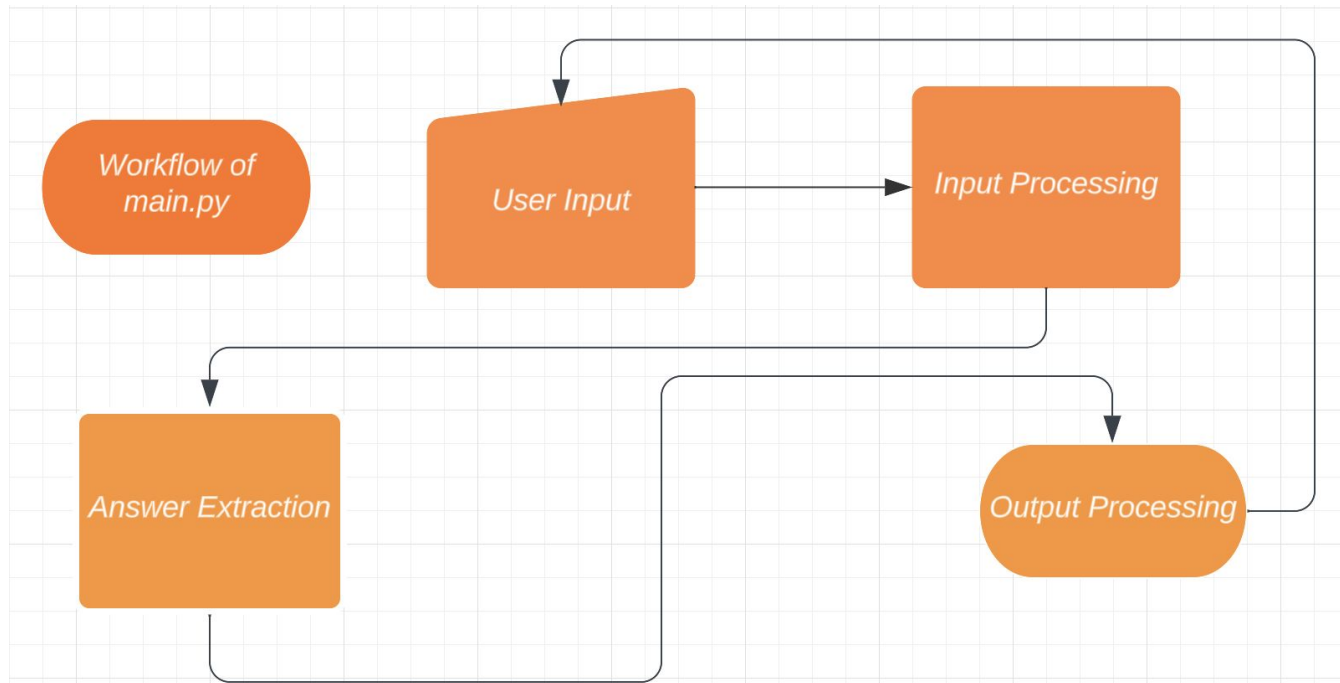
Regex Chatbot Pipeline Overview

The background of the slide is a solid orange color. In the top right corner, there are three decorative elements: a small circle with a pie chart, a larger circle with a pie chart, and another small circle with a pie chart, all in varying shades of orange.



Implementation and Development

- Ani
 - Output
- Bryson
 - Data
 - Architecture
- Alex
 - Answer Extraction
 - Text Processing
- Andy
 - UI
 - Concurrency
 - Main.py





Regex Chatbot Pipeline

- Input Processing
- 



Input Processing - Macroscopic

- Utilizes regular expressions for signaling answer extraction
 - Generalized detection of words associated with each prompt
 - Verbiage one could anticipate in this context
 - Specific keyword detection based on novel selection
 - *Murder on the Links*
 - *The Sign of the Four*
 - *Mysterious Affair*
- Stemming employed to detect broadest usages of words
 - Catch most cases
 - Account for distinct manners in which users might interact
- Unique re for each potential prompt of 6
 - Investigator
 - Crime
 - Perpetrator
 - Words adjacent to perpetrator
 - Concurrence of investigator and perpetrator
 - Suspects

```
def detInvest(line):  
    return re.findall('invest|detect|protag|main|hercule|poiro|arthur|hastings|giraud|  
    sherlock|holmes|john|watson|inform|sleuth|officer|agent|inspect|police', line)  
  
def detCrime(line):  
    return re.findall('crime|thef|murder|steal|stole|kill|stab|thiev|kni|theft|murder|  
    prison|escape|conspir|shock|poison|stryc|felony|scandal|wrong|lawless|misdeed|  
    assass|homici|shoot|slay|burgli|burgle', line)  
  
def detPerp(line):  
    return re.findall('commit|crimin|perp|murderer|killer|thief|bad|evil|marthe|  
    daubreuil|tonga|small|sholto|alf|inglethorp|felon|crook|robber|burglar', line)  
  
def detAdj(line):  
    return re.findall('adjac|around|near|close|next|border|neigh|surround', line)  
  
def detConc(line):  
    return re.findall('meet|concur|clash|fight|talk|speak|capture|detain|arrest|  
    apprehend|convers|contact|face|see|altercat|conflict|brawl|exchang|skirmi|scuffl|  
    chat|exchang|imprison|jail|incarcer|interrogat|pair|co.?occur', line)  
  
def detSus(line):  
    return re.findall('poss|inter|susp|potent|jack|bella|duveen|morstan|small|jonathan|  
    john|thaddeus|evelyn|howard|cynthia|murdoch|mary|cavendish|lawrence|conceiv|feas|  
    plaus|proba|viabl|likel', line)
```

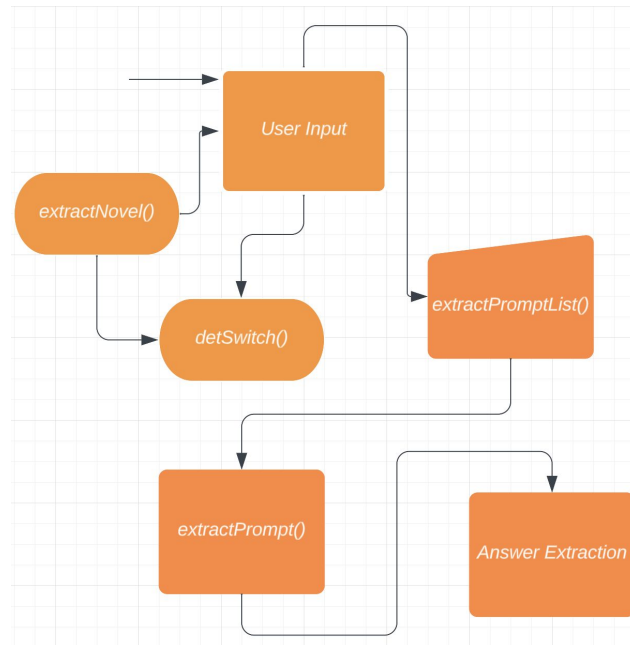


Input Processing - Implementation Details

- Extracts every match in user input to any regular expression
- Check matches against potential queries with implicit decision tree
 - Handle complex cases first; simultaneous flags
 - Then checks descending order of most specific regular expressions
- Passes best match to answer extraction pipeline

Implications, Challenges, Outcomes

- Accurate query detection can be misidentified
 - Bias of assumptions in decision tree
 - Intersection is accounted for, but not universally
- Intrinsic limitation in keyword detection from a finite list
- Difficulty in more complex cases such as concurrence
 - Many prompts could indicate concurrence case
 - Detection of this case suffers more from finite list
 - Detecting intersection of 3 cases
 - Many possible ways to inquire about simultaneous character appearance
- Primary challenge: consistently detecting sentiment of user accurately
- Outcome
 - Tends to signal answer extraction accurately
 - But, iterative development, addition of vocabulary highlights shortcomings





Regex Chatbot Pipeline

- Data Preprocessing
- 



Human Analysis

- We first answered five of the six required questions about the three novels ourselves
 - Read detailed plot/chapter summaries on LitCharts, OpenAI's ChatGPT, and Wikipedia
 - Gave us details and approximate locations for first appearances of:
 - Crime
 - Investigators
 - Perpetrators
 - All major suspects
 - Investigators together with suspects
 - Followed up by finding exact answers to these questions in the Project Gutenberg novel text
 - Last question on words around the perpetrator was left temporarily unanswered (best left for programmatic analysis)



**PROJECT
GUTENBERG**

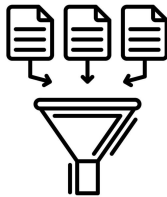
<https://stevelaube.com/wp-content/uploads/2021/11/gutenberg.jpg>

 **LitCharts**

<https://d2xp6zio4g87q8.cloudfront.net/logo-and-wordmark.png>



Basic Data Cleaning



- Data Collection

- Project Gutenberg - used as a data source for all four novels.
Selected a uniform format for input files
- Requests Python package - used for website scraping
- BeautifulSoup API - used for HTML parsing, text extraction
- Robust data collection can be used in the future for other books by simply repopulating a list of links to the novels of interest

```
r = requests.get(WebsiteList[i])  
soup = BeautifulSoup(r.text, 'html.parser')  
textExtracted = soup.get_text().lower()
```

- Text Cleaning

- Analyzed the input format, removed HTML tags
- Used RegEx to remove unnecessary characters for all novels
- Create new text files with chapter and sentence data.

```
textExtracted.split(str("*** START OF THE PROJECT GUTENBERG EBOOK ").lower())
```

```
wordsInBook[j] = re.sub('[!,"&|:@,<>()*$\\\/;="\'"]', '', wordsInBook[j])  
wordsInBook[j] = re.sub('[^0-9\.\']*$', '', wordsInBook[j])
```

The result - relevant novel text for each book to be separated into chapter, sentence objects



Chapter Preprocessing

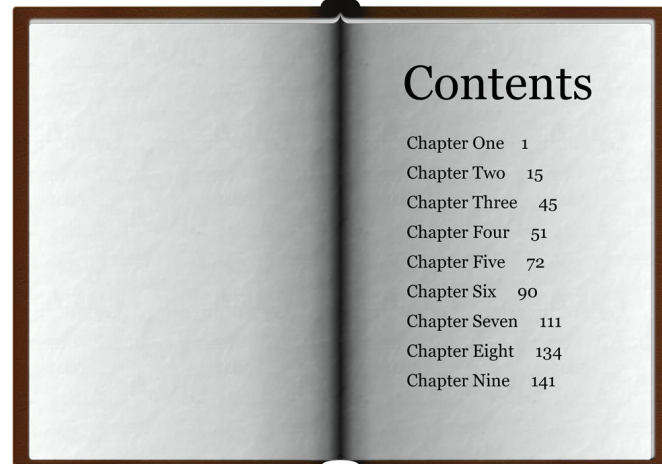
- Chapter Detection and Representation:
 - Different chapter separators based on the novel
 - “Murder on the Links”: exception to the traditional Roman numbered chapters

```
CHAPTER_KEYWORDS = ('chapter i\\W', 'chapter ii\\W', 'chapter iii\\W',  
                    'chapter ix\\W', 'chapter x\\W', 'chapter xi\\W',
```

```
'1\\W*\\.\\W*\\.ellow\\W*\\.raveller',  
'2\\W*\\.n\\W*\\.ppeal\\W*\\.or\\W*\\.elp',  
'3\\W*\\.t\\W*\\.he\\W*\\.illa\\W*\\.enevi\\.ve',
```

- Each chapter is represented as an element in a **list**
- Each element contains full chapter text
- Population of the list happens through **RegEx text splitting** based on the chapter separators.

```
chapters = re.split(chaptersPattern, self.getFullText())
```





Sentence Preprocessing

- For each chapter, split sentences into a list
 - End up with a nested list where:
 - First index keys into chapter
 - Second index keys into sentence
 - Value is sentence string without punctuation
- Avoided using Python NLP package for automatic sentence splitting
 - Manually split sentences using RegEx
 - For more RegEx practice
- Split sentences using comma, question mark, and exclamation point
 - Had to account for abbreviations
 - Mr., Mrs., Ms., Dr., Prof., a.m., p.m., etc.
 - French abbreviation M. is in *Murder on the Links*
 - Kept abbreviation period by temporarily replacing them with '<tmp_prd>'

```
separators = [',', '?', '!']  
pattern = '|'.join(map(re.escape, separators))  
novelChapters = self.getChaptersText()
```

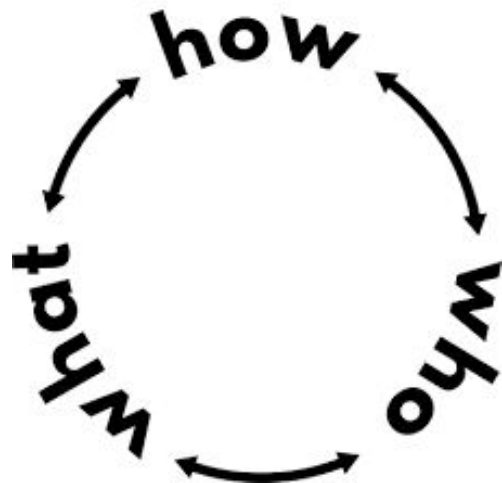
```
novelChaptersAbbrevReplaced = re.sub(r'([aApP])\.([mM])\.', r'\1<tmp_prd>\2<tmp_prd>', novelChapters[chapterNumber])  
novelChaptersAbbrevReplaced = re.sub(r'(\W)(mr|mrs|ms|dr|st|m|prof|capt|cpt|lt|mme|mlle)\.', r'\1\2<tmp_prd>', novelChaptersAbbrevReplaced)  
chapterSentences = re.split(pattern, novelChaptersAbbrevReplaced)
```



Answer Extraction: Prompts [1,3,and 6], [4]

- Motivation for grouping:
 - Similar nature of answer extraction
 - The choice of sentence data structure allows us to identify chapter and sentence number
 - Finding the first occurrences of particular individuals in a sorted list of sentences is trivial
- When does the investigator (or a pair)/ the perpetrator / the suspects occur for the first time ?
 - Use the names of the individuals to conduct a search
 - Stop after first mention of an individual of interest occurs
 - For #6, return a list of individual mentions with the rest of the details
 - Expand the names of individuals to the one that conveys most information in the process
 - Holmes-> Sherlock Holmes
- Three words around the perpetrator on each mention
 - Search the sentence list for each occurrence of the identified perpetrator
 - Create lists of three words before and after the mention
 - Potential for more intelligent type of crime detection
 - Expand the range of search to capture several sentences, maybe a paragraph

```
precedingWords = splitSentence[0].split()
preceding3Words = precedingWords[-3:]
preceding3WordsJoined = ' '.join(preceding3Words)
```



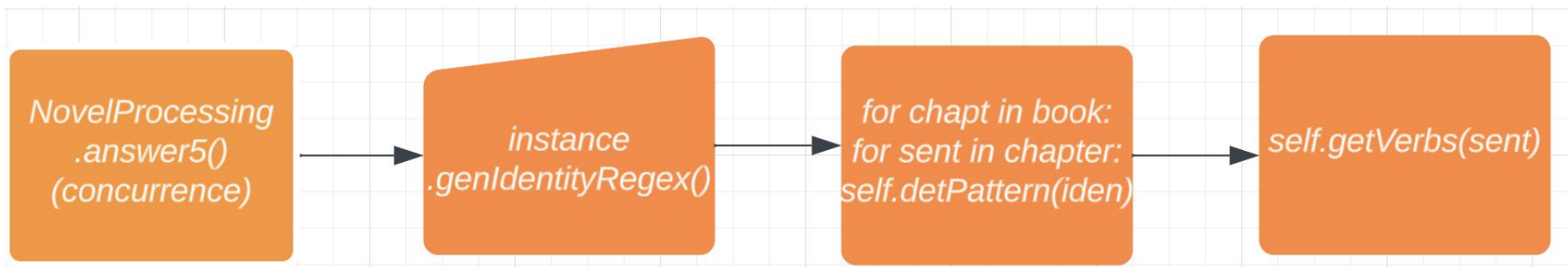


Answer Extraction: Prompt 2 (Crime)

- Use following as anchors:
 - Hardcoded victim names for each novel
 - Causes of death for murder
 - Poisoned
 - Stabbed
- Use RegEx to search each sentence in each novel for the structure:
 - (she|he|it|[victim name]) (was|is|(has|had) been|means) (murder(ed)|killed|poisoned|stabbed)
 - Matches dramatic proclamations of murder common in detective novels:
 - “‘I believe she has been poisoned’” (*The Mysterious Affair At Styles*)
 - “‘It means murder,’ said he, stooping over the dead man” (*The Sign of the Four*)
 - “‘But be careful, for it is poisoned’” (*The Sign of the Four*)
 - “‘Renauld was murdered this morning’” (*Murder On the Links*)
 - “‘[...]M. Renauld’s body had been discovered, stone dead, stabbed in the back’” (*Murder On the Links*)
 - Also extracts the correct cause of death from each novel



Answer Extraction: Prompt 5 (Concurrency)



- Based on novel, produce patterns based on novel-specific identities
- Find the first occurrence of those patterns chapter and sentence wise
- Extract the words indicative of actions in occurrence
 - Generalized and verb specific detection of interaction between protagonist and antagonist

Pitfall: depending on interpretation of problem definition, detecting indirect references by name could be considered a false positive



Regex Chatbot Pipeline

- Output Processing
- 



Output Processing

- Convert all the data analysis done on novels into clear and understandable sentences
- Composed of three components
 - Output Interface
 - Formats each field of the extracted answer into a dictionary
 - A different dictionary for each prompt
 - Response Templates
 - Contains templates of answers for each prompt
 - Has variables from Output Interface's dictionary that are replaced with the extracted answers during runtime
 - Multiple response wordings for each prompt
 - Output Processor
 - Driver that puts together the data and the templates into a readable response
 - Does some post-processing such as formatting word tense and capitalization
 - Randomly selects response template for a given prompt
 - Challenges
 - Fitting data into templates
 - Consistency in answer generation



Results





Results

```
(cs524) brysongullett@bgullet1macbook 524Project1 % python3 main.py
```

```
Hi, I'm the Detective Novel re based chatbot for "Murder on the Links", "Sign of the Four", and "Mysterious Affair". You can ask me questions about the criminals, detectives and crime. When you're done, just say "exit". If you'd like to change which novel we're discussing at any time, just say so! What story would you like to start with?
```

```
I'd like to talk about the novel Sign of the Four. Can you tell me about it?
```

```
What would you like to know?
```

```
First off, who was the detective in the story?
```

```
The investigator Sherlock Holmes first appears in Chapter 1 and Sentence 1.
```

```
What would you like to know?
```

```
What crime was being investigated?
```

```
The crime where Bartholomew was poisoning was first discussed in Chapter 5, Sentence 171.
```

```
What would you like to know?
```

```
Who committed the act?
```

```
Chapter 11, Sentence 14 is the first mention of the perpetrator, whose name is Tonga.
```

```
What would you like to know?
```

```
Who were the other suspects?
```

```
The other suspects were first introduced as follows: Arthur Morstan in Chapter 1, Sentence 204, and Jonathan Small in Chapter 1, Sentence 22.
```

```
What would you like to know?
```

```
When did Tonga and Holmes first clash? What were the circumstances of their encounter?
```

```
The detective and perpetrator appear together in Chapter 3, Sentence 379. The action words that are used to describe the interaction are consulting, this, died, file s, times, suggests.
```

```
What would you like to know?
```

```
What words surrounded the perpetrator throughout the novel, characterizing them more broadly?
```

```
Around the perpetrator's mention in Chapter 2, Sentence 47, we have these words: Before - know of major. After - of his own.
```

```
In Chapter 3, Sentence 11, the three words preceding perpetrator are: times that major, and the three words following perpetrator are: of upper norword.
```

```
Around the perpetrator's mention in Chapter 3, Sentence 16, we have these words: Before - visited is major. After - .
```

```
When looking at the perpetrator's mention in Chapter 3, Sentence 17, the words in the immediate vicinity are: major (before) and denies having heard (after).
```

```
Around the perpetrator's mention in Chapter 3, Sentence 18, we have these words: Before - four years later. After - dies.
```



Results

```
What would you like to know?  
Alright, that's all I wanted to know. Can we transition to another novel, say Murder on the Links?  
Ok, we'll switch to that one.  
What would you like to know?  
exit  
(cs524) brysongullett@bgullet1macbook 524Project1 %
```