SCS-3666-03 Applied Natural Language Processing

ChatBot using Sherlock Holmes Corpus

December 13, 2021

Kiranpal Sohi, Lawrence Man

Objective

Our team had two main objectives:

- Conduct a Data Exploration
- Create a ChatBot

Data Set:

- Sherlock Holmes corpus consisting of sixty (60) books was used for the project
- Link: https://sherlock-holm.es/ascii/

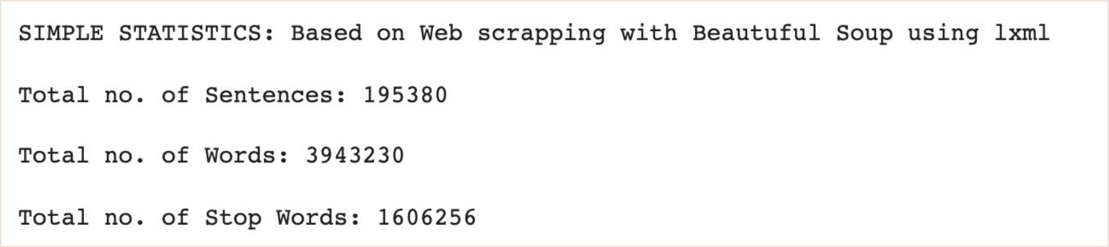
In Data Exploration, the team completed the following:

- Calculated Simple Statistics
- Created Word Cloud
- Developed a Social Network Graph
- Presented a Topic Cloud

In ChatBot, the team attempted to create:

- Rule-Based ChatBot
- Use of Recommender

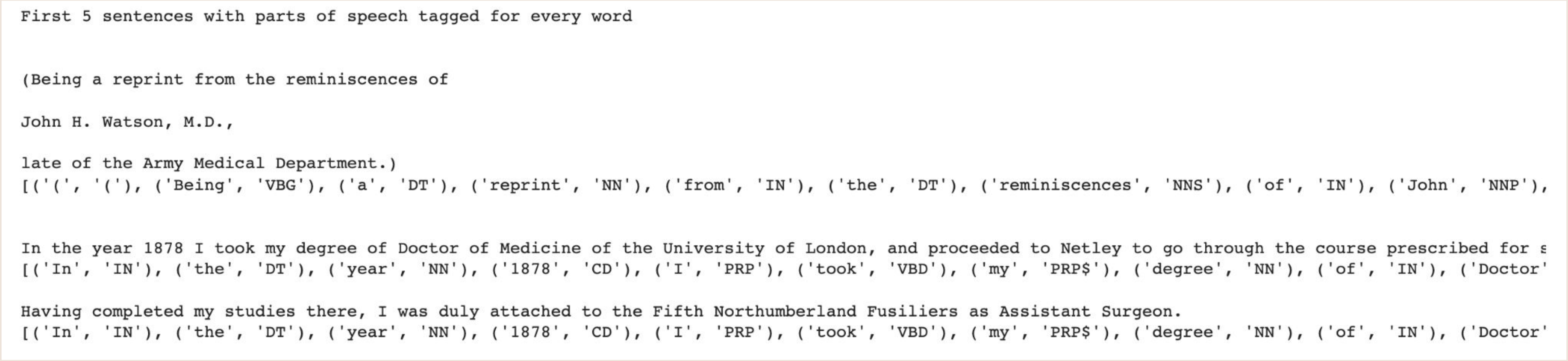
Data Exploration – Simple Statistics



```
SIMPLE STATISTICS: Based on Web scrapping with Beautuful Soup using lxml

Total no. of Sentences: 195380

Total no. of Words: 3943230

Total no. of Stop Words: 1606256
```

Simple statistics provided information on the number of sentences and words.

Almost half the corpus is made of Stop Words.

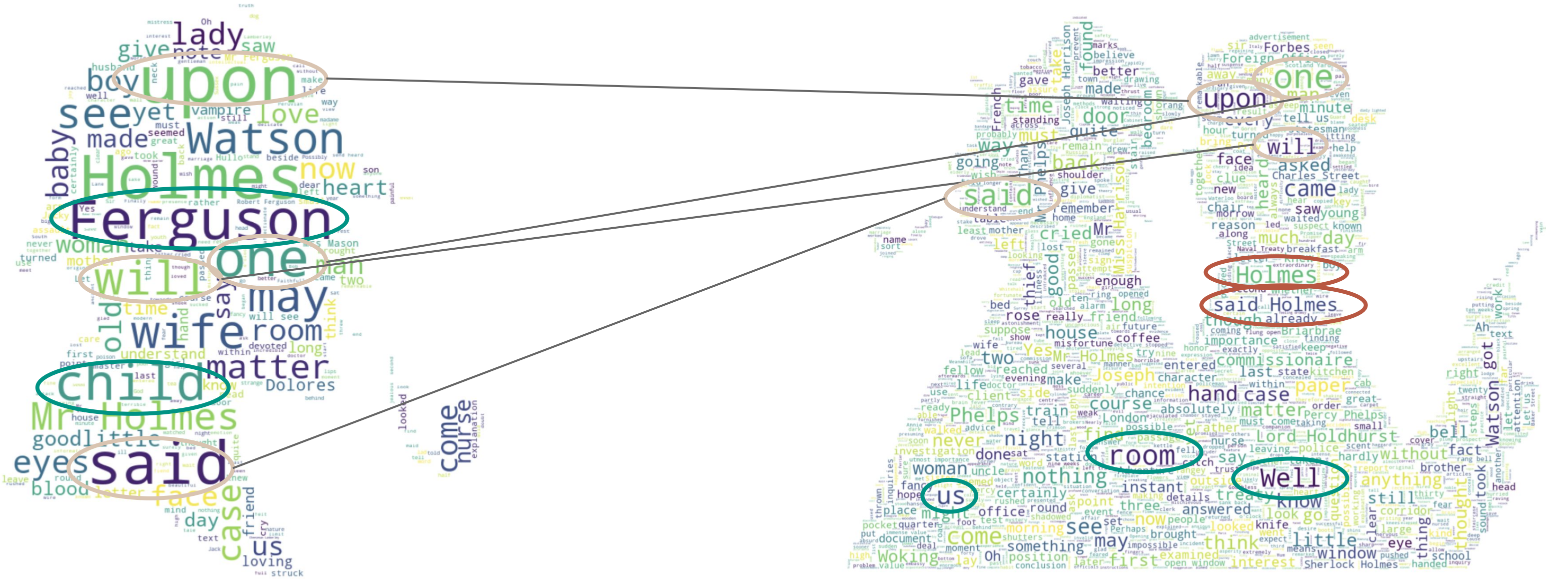


```
First 5 sentences with parts of speech tagged for every word


(Being a reprint from the reminiscences of

John H. Watson, M.D.,

late of the Army Medical Department.)
[('(', '('), ('Being', 'VBG'), ('a', 'DT'), ('reprint', 'NN'), ('from', 'IN'), ('the', 'DT'), ('reminiscences', 'NNS'), ('of', 'IN'), ('John', 'NNP'),


In the year 1878 I took my degree of Doctor of Medicine of the University of London, and proceeded to Netley to go through the course prescribed for s
[('In', 'IN'), ('the', 'DT'), ('year', 'NN'), ('1878', 'CD'), ('I', 'PRP'), ('took', 'VBD'), ('my', 'PRP$'), ('degree', 'NN'), ('of', 'IN'), ('Doctor'

Having completed my studies there, I was duly attached to the Fifth Northumberland Fusiliers as Assistant Surgeon.
[('In', 'IN'), ('the', 'DT'), ('year', 'NN'), ('1878', 'CD'), ('I', 'PRP'), ('took', 'VBD'), ('my', 'PRP$'), ('degree', 'NN'), ('of', 'IN'), ('Doctor'
```

Data Exploration – Word Cloud

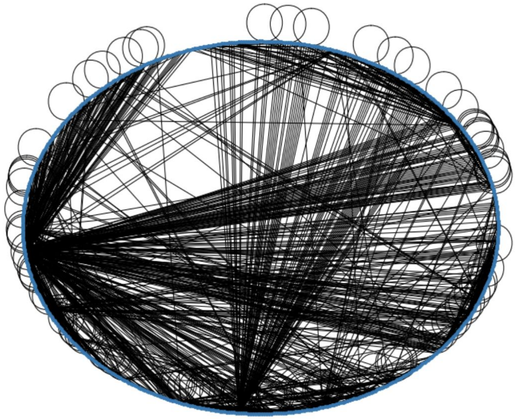
The Sussex Vampire

The Naval Treaty

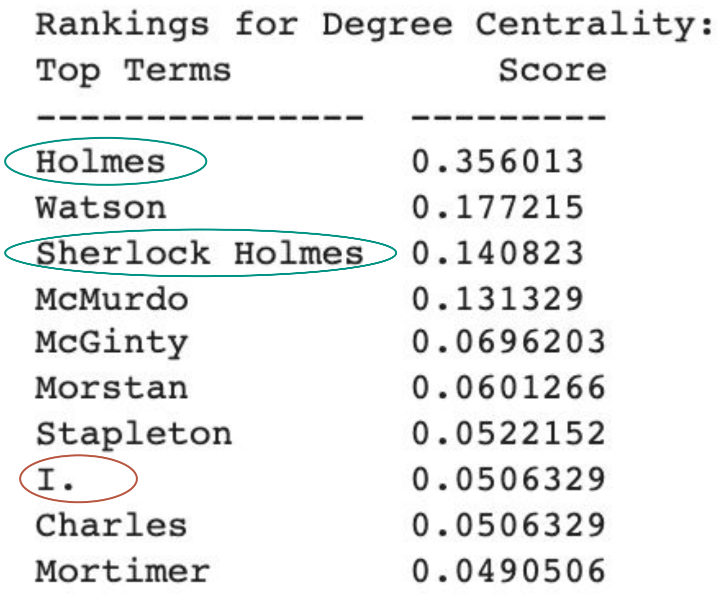


Data Exploration – Social Network Graph

Social Network Graph was created using first ten (10) books in the corpus. As expected, Holmes and Watson have the highest ranking for Degree Centrality.

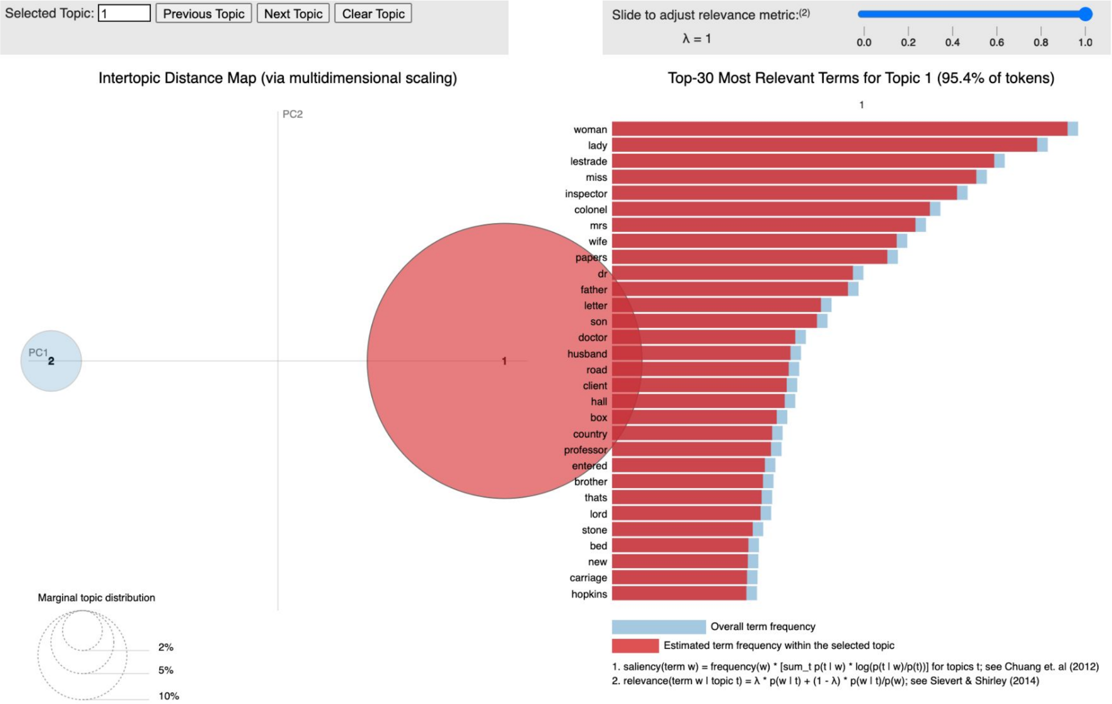


Note: Using all sixty (60) books crashed colab notebook and using thirty (30) books did not create a more meaningful graph.



```
Rankings for Degree Centrality:
Top Terms                  Score
----------------     ---------
Holmes               0.356013
Watson               0.177215
Sherlock Holmes      0.140823
McMurdo              0.131329
McGinty              0.0696203
Morstan              0.0601266
Stapleton            0.0522152
I.                   0.0506329
Charles              0.0506329
Mortimer             0.0490506
```

Data Exploration – Topic Model



Selected Topic: 1 Previous Topic Next Topic Clear Topic

Slide to adjust relevance metric:(2)

λ = 1

Intertopic Distance Map (via multidimensional scaling)

Top-30 Most Relevant Terms for Topic 1 (95.4% of tokens)

🟦 Overall term frequency

🟥 Estimated term frequency within the selected topic

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

A number of different combinations were run for Topic Model using Number of Stories and Number of Topics.

The best option seems to be:

- Use all sixty (60) stories
- Use only two (2) topics

For almost all combinations, one topic consisting of terms “woman”, “lady”, “miss”, mrs.” were highly prevalent in the Sherlock Holmes corpus.

ChatBot (aka Sherlock Holmes Trivia Detective) #1

Problem Statement

- This chatbot takes input questions and retrieves the stories that are most relevant to the terms (nouns) specified

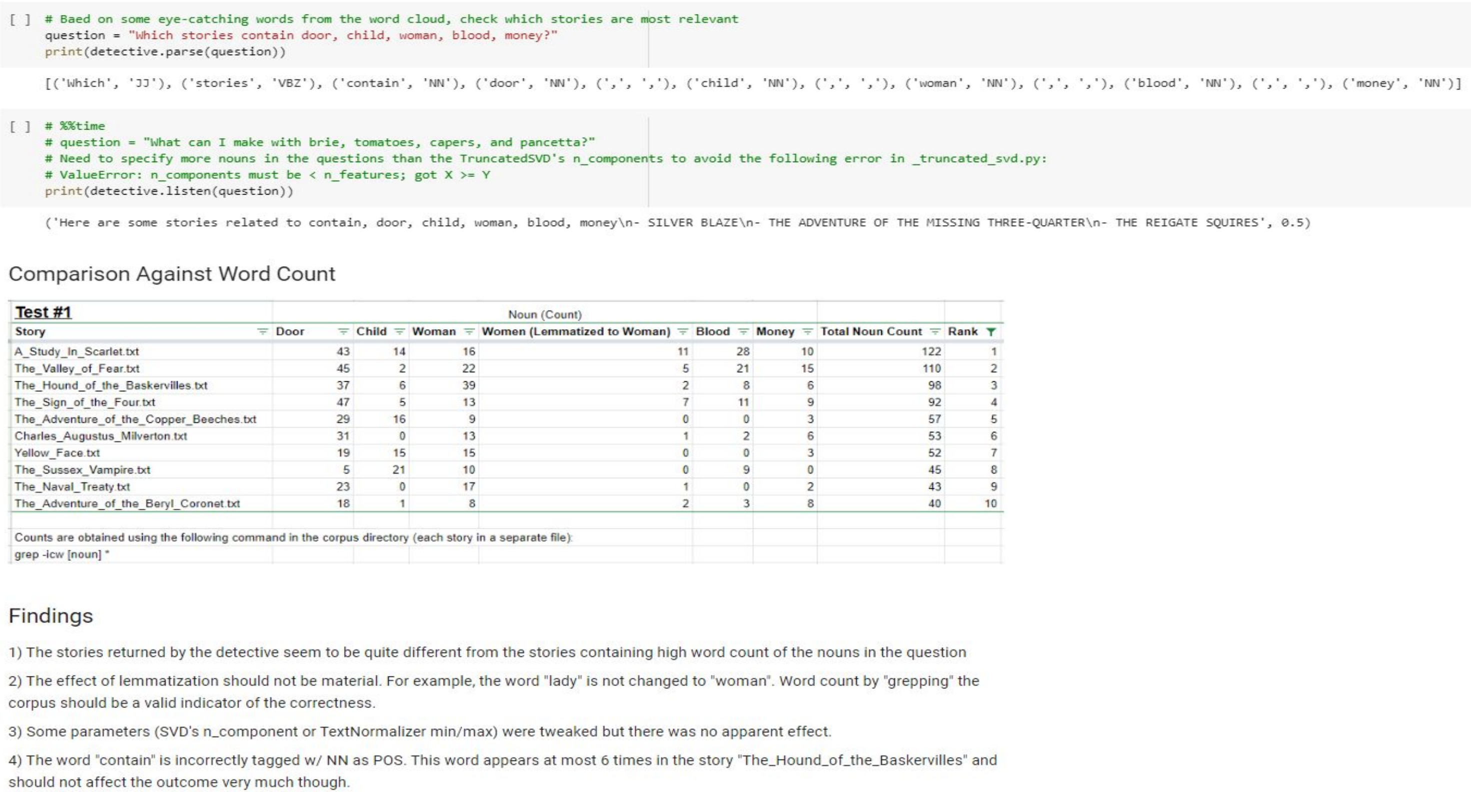
Methodology

- Started from the chatbot in textbook Chapter 10 (used in module 8) as baseline
- Tweaked the recipe recommender to retrieve most relevant story from the corpus, based on the input keywords
- A notebook (pickle_Sherlock_Holmes_corpus.ipynb) is created to pre-process text corpus (60 stories, total size ~3.7MB)
- The chatbot is implemented in another notebook (Sherlock_Holmes_trivia_detective.ipynb)

ChatBot (aka Sherlock Holmes Trivia Detective) #2

Results:

Test #1



```
[ ]  # Baed on some eye-catching words from the word cloud, check which stories are most relevant
     question = "Which stories contain door, child, woman, blood, money?"
     print(detective.parse(question))

     [('Which', 'JJ'), ('stories', 'VBZ'), ('contain', 'NN'), ('door', 'NN'), (',', ','), ('child', 'NN'), (',', ','), ('woman', 'NN'), (',', ','), ('blood', 'NN'), (',', ','), ('money', 'NN')]
```

```
[ ]  # %%time
     # question = "What can I make with brie, tomatoes, capers, and pancetta?"
     # Need to specify more nouns in the questions than the TruncatedSVD's n_components to avoid the following error in _truncated_svd.py:
     # ValueError: n_components must be < n_features; got X >= Y
     print(detective.listen(question))

     ('Here are some stories related to contain, door, child, woman, blood, money\n- SILVER BLAZE\n- THE ADVENTURE OF THE MISSING THREE-QUARTER\n- THE REIGATE SQUIRES', 0.5)
```

Comparison Against Word Count

Test #1	Noun (Count)							
Story	Door	Child	Woman	Women (Lemmatized to Woman)	Blood	Money	Total Noun Count	Rank
A_Study_In_Scarlet.txt	43	14	16	11	28	10	122	1
The_Valley_of_Fear.txt	45	2	22	5	21	15	110	2
The_Hound_of_the_Baskervilles.txt	37	6	39	2	8	6	98	3
The_Sign_of_the_Four.txt	47	5	13	7	11	9	92	4
The_Adventure_of_the_Copper_Beeches.txt	29	16	9	0	0	3	57	5
Charles_Augustus_Milverton.txt	31	0	13	1	2	6	53	6
Yellow_Face.txt	19	15	15	0	0	3	52	7
The_Sussex_Vampire.txt	5	21	10	0	9	0	45	8
The_Naval_Treaty.txt	23	0	17	1	0	2	43	9
The_Adventure_of_the_Beryl_Coronet.txt	18	1	8	2	3	8	40	10

Counts are obtained using the following command in the corpus directory (each story in a separate file):

```
grep -icw [noun] *
```

Findings

1) The stories returned by the detective seem to be quite different from the stories containing high word count of the nouns in the question

2) The effect of lemmatization should not be material. For example, the word "lady" is not changed to "woman". Word count by "grepping" the corpus should be a valid indicator of the correctness.

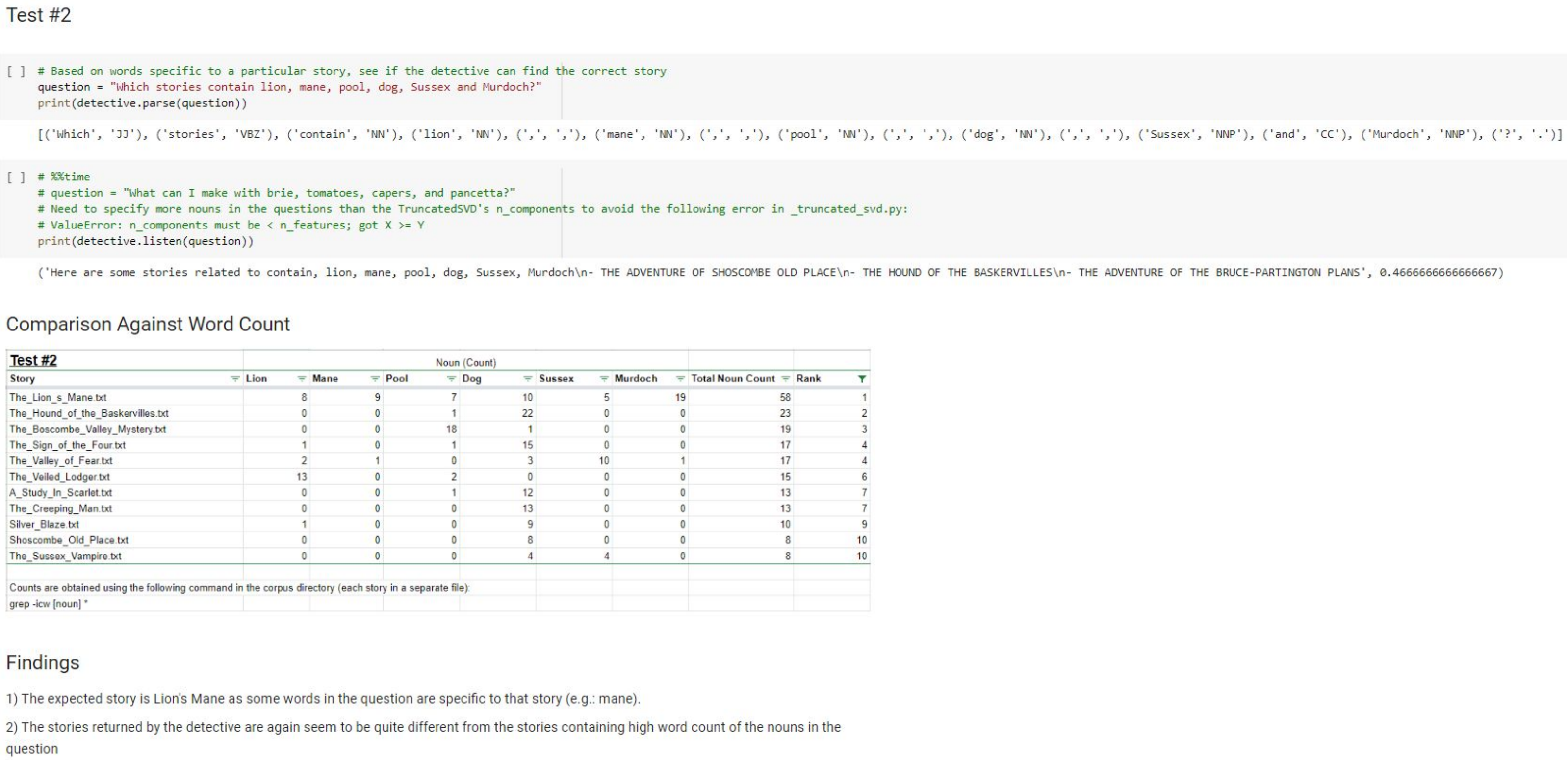
3) Some parameters (SVD's n_component or TextNormalizer min/max) were tweaked but there was no apparent effect.

4) The word "contain" is incorrectly tagged w/ NN as POS. This word appears at most 6 times in the story "The_Hound_of_the_Baskervilles" and should not affect the outcome very much though.

ChatBot (aka Sherlock Holmes Trivia Detective) #3

Results:

Test #2



Test #2

```
[ ] # Based on words specific to a particular story, see if the detective can find the correct story
    question = "Which stories contain lion, mane, pool, dog, Sussex and Murdoch?"
    print(detective.parse(question))

    [('Which', 'JJ'), ('stories', 'VBZ'), ('contain', 'NN'), ('lion', 'NN'), (',', ','), ('mane', 'NN'), (',', ','), ('pool', 'NN'), (',', ','), ('dog', 'NN'), (',', ','), ('Sussex', 'NNP'), ('and', 'CC'), ('Murdoch', 'NNP'), ('?', '.')]
```

```
[ ] # %%time
    # question = "What can I make with brie, tomatoes, capers, and pancetta?"
    # Need to specify more nouns in the questions than the TruncatedSVD's n_components to avoid the following error in _truncated_svd.py:
    # ValueError: n_components must be < n_features; got X >= Y
    print(detective.listen(question))

    ('Here are some stories related to contain, lion, mane, pool, dog, Sussex, Murdoch\n- THE ADVENTURE OF SHOSCOMBE OLD PLACE\n- THE HOUND OF THE BASKERVILLES\n- THE ADVENTURE OF THE BRUCE-PARTINGTON PLANS', 0.4666666666666667)
```

Comparison Against Word Count

Test #2			Noun (Count)					
Story	**Lion**	**Mane**	**Pool**	**Dog**	**Sussex**	**Murdoch**	**Total Noun Count**	**Rank**
The_Lion_s_Mane.txt	8	9	7	10	5	19	58	1
The_Hound_of_the_Baskervilles.txt	0	0	1	22	0	0	23	2
The_Boscombe_Valley_Mystery.txt	0	0	18	1	0	0	19	3
The_Sign_of_the_Four.txt	1	0	1	15	0	0	17	4
The_Valley_of_Fear.txt	2	1	0	3	10	1	17	4
The_Veiled_Lodger.txt	13	0	2	0	0	0	15	6
A_Study_In_Scarlet.txt	0	0	1	12	0	0	13	7
The_Creeping_Man.txt	0	0	0	13	0	0	13	7
Silver_Blaze.txt	1	0	0	9	0	0	10	9
Shoscombe_Old_Place.txt	0	0	0	8	0	0	8	10
The_Sussex_Vampire.txt	0	0	0	4	4	0	8	10

Counts are obtained using the following command in the corpus directory (each story in a separate file):

```
grep -icw [noun] *
```

Findings

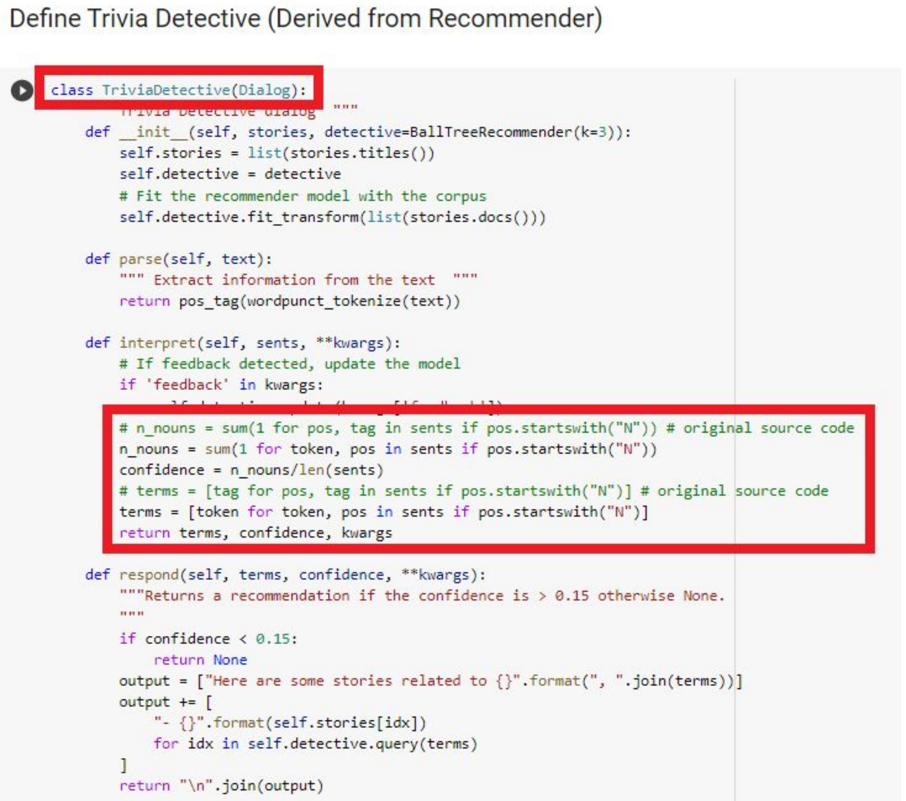
1) The expected story is Lion's Mane as some words in the question are specific to that story (e.g.: mane).

2) The stories returned by the detective are again seem to be quite different from the stories containing high word count of the nouns in the question

ChatBot (aka Sherlock Holmes Trivia Detective) #4

Discussion #1

- The original source code from textbook’s RecipeRecommender always returns None and a confidence value of 0 regardless of input
- The highlighted modifications were made to rectify the situation



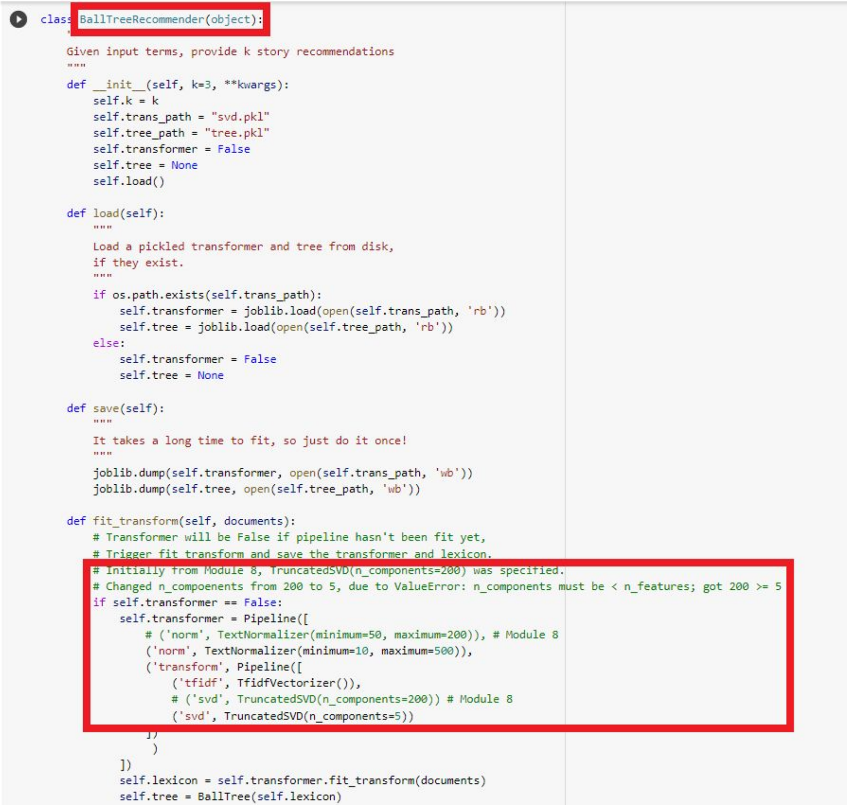
Define Trivia Detective (Derived from Recommender)

```
class TriviaDetective(Dialog):
    """Trivia Detective dialog  """
    def __init__(self, stories, detective=BallTreeRecommender(k=3)):
        self.stories = list(stories.titles())
        self.detective = detective
        # Fit the recommender model with the corpus
        self.detective.fit_transform(list(stories.docs()))

    def parse(self, text):
        """ Extract information from the text  """
        return pos_tag(wordpunct_tokenize(text))

    def interpret(self, sents, **kwargs):
        # If feedback detected, update the model
        if 'feedback' in kwargs:
            self.detective.update(kwargs['feedback'])
        # n_nouns = sum(1 for pos, tag in sents if pos.startswith("N")) # original source code
        n_nouns = sum(1 for token, pos in sents if pos.startswith("N"))
        confidence = n_nouns/len(sents)
        # terms = [tag for pos, tag in sents if pos.startswith("N")] # original source code
        terms = [token for token, pos in sents if pos.startswith("N")]
        return terms, confidence, kwargs

    def respond(self, terms, confidence, **kwargs):
        """Returns a recommendation if the confidence is > 0.15 otherwise None.
        """
        if confidence < 0.15:
            return None
        output = ["Here are some stories related to {}".format(", ".join(terms))]
        output += [
            "- {}".format(self.stories[idx])
            for idx in self.detective.query(terms)
        ]
        return "\n".join(output)
```

ChatBot (aka Sherlock Holmes Trivia Detective) #5

Discussion #2

- After modification, the chatbot could return some responses, but were incorrect
- The highlighted parameter values in the BallTreeRecommender were adjusted (with more other values tried but not shown here). There was no apparent improvement.



```
class BallTreeRecommender(object):
    Given input terms, provide k story recommendations
    """
    def __init__(self, k=3, **kwargs):
        self.k = k
        self.trans_path = "svd.pkl"
        self.tree_path = "tree.pkl"
        self.transformer = False
        self.tree = None
        self.load()

    def load(self):
        """
        Load a pickled transformer and tree from disk,
        if they exist.
        """
        if os.path.exists(self.trans_path):
            self.transformer = joblib.load(open(self.trans_path, 'rb'))
            self.tree = joblib.load(open(self.tree_path, 'rb'))
        else:
            self.transformer = False
            self.tree = None

    def save(self):
        """
        It takes a long time to fit, so just do it once!
        """
        joblib.dump(self.transformer, open(self.trans_path, 'wb'))
        joblib.dump(self.tree, open(self.tree_path, 'wb'))

    def fit_transform(self, documents):
        # Transformer will be False if pipeline hasn't been fit yet,
        # Trigger fit transform and save the transformer and lexicon.
        # Initially from Module 8, TruncatedSVD(n_components=200) was specified.
        # Changed n_compoenents from 200 to 5, due to ValueError: n_components must be < n_features; got 200 >= 5
        if self.transformer == False:
            self.transformer = Pipeline([
                # ('norm', TextNormalizer(minimum=50, maximum=200)), # Module 8
                ('norm', TextNormalizer(minimum=10, maximum=500)),
                ('transform', Pipeline([
                    ('tfidf', TfidfVectorizer()),
                    # ('svd', TruncatedSVD(n_components=200)) # Module 8
                    ('svd', TruncatedSVD(n_components=5))
                ])
                 )
            ])
            self.lexicon = self.transformer.fit_transform(documents)
            self.tree = BallTree(self.lexicon)
```

ChatBot (aka Sherlock Holmes Trivia Detective) #6

Discussion #3

With two unsuccessful test cases, the possible causes of discrepancies are:

- A story is much longer than a cooking recipe and contains a richer variety of words. Its vector representation has a much higher dimension.
- On the other hand, the SVD (single value decomposition) algorithm reduces the vector to a much lower dimension (n_components=5). A greater n_components is not suitable as this in turn requires specifying more terms in the question used in searching. Otherwise, the message "ValueError: n_components must be < n_features" will be prompted. n_features correspond to the number of nouns in the question.

Challenges

Data Exploration

- As data type used by individual exploration analysis were different, the data had to be manipulated to serve the analysis parameters' purpose. This was difficult even though Sherlock Holmes corpus was provided in various formats.
- Deciding on the "perfect" parameters was a challenge as the results were difficult to compare specifically for Topic Model.

Chatbot

- Need to debug the errors in the original sample code from the textbook.
- After fixing runtime errors, the chatbot could not return comparable results that are based on simple word count. The values of the attributes and the outputs of method of various classes (e.g. TextNormalizer) appear to be reasonable.
- The call to BallTreeRecommender is a black box. No apparent improvement is achieved by tweaking the parameters.

Lessons

Data Exploration

- Simple Statistics: A frequency distribution of words would have added to the understanding of the corpus
- Word Cloud: A cloud with all sixty (60) books may have provided some common usage of words for the corpus
- Social Network Graph: Cleaning of entities would have helped clean up the results
- Topic Modeling: More fine-tuning may improve the variability of topics within the corpus

ChatBot

- To provide better response, it is believe that the response should be returned at paragraph level (together with the title) rather than story level. The vectorized dimension will be much lower, and the response can be readily verified.
- With the above improvement, the chatbot can be applied to ingest say a textbook. Students can then make use of this to locate relevant paragraphs that contain answer to their questions. Lemmatization and confidence level ordering improve the search result over simple plain text search.
- Transfer learning appears promising to achieve the objective. However, after trying out, it took too long (>1 hour) and did not respond.

Q&A