Natural Language Processing Spark Magic

By Michaella Steinruck

Introduction

- The entire Harry Potter series - 1,0804, 170 words
- 7 books over 10 years
- Over 80 spells
- Over 700 characters
- Overlapping plot/topics

https://5.imimg.com/data5/SELLER/Default/2020/8/PE/PX/MO/54836353/harry-potter-books-collection-j-k-rowling-bloo

Areas of Interest

SparkNLP

SparkML

Topic Modelling

The Data

- 7 .txt files
- Each a single line
- Formatted using:
  - DocumentAssembler()
  - SentenceDetector()
  - Tokenizer()

```
root
 |-- text: string (nullable = true)
 |-- titles: string (nullable = true)
```

SparkNLP

Exploration

- 84978 sentences across all books
- RAM issues in Google Colab
- Dependency issues with SparkNLP on local Spark instance and DataBricks
- Decided to finish cleaning and exploring using regular SparkML

Sentences

- By default SparkNLP SentenceDetector() returns and array of sentences

```
setExplodeSentences(True)
```

- Allows each sentence to become own row
- Sentences can be referenced by

```
df = pipeline_model.withColumn('sentence', pipeline_model.sentences['result'])
```

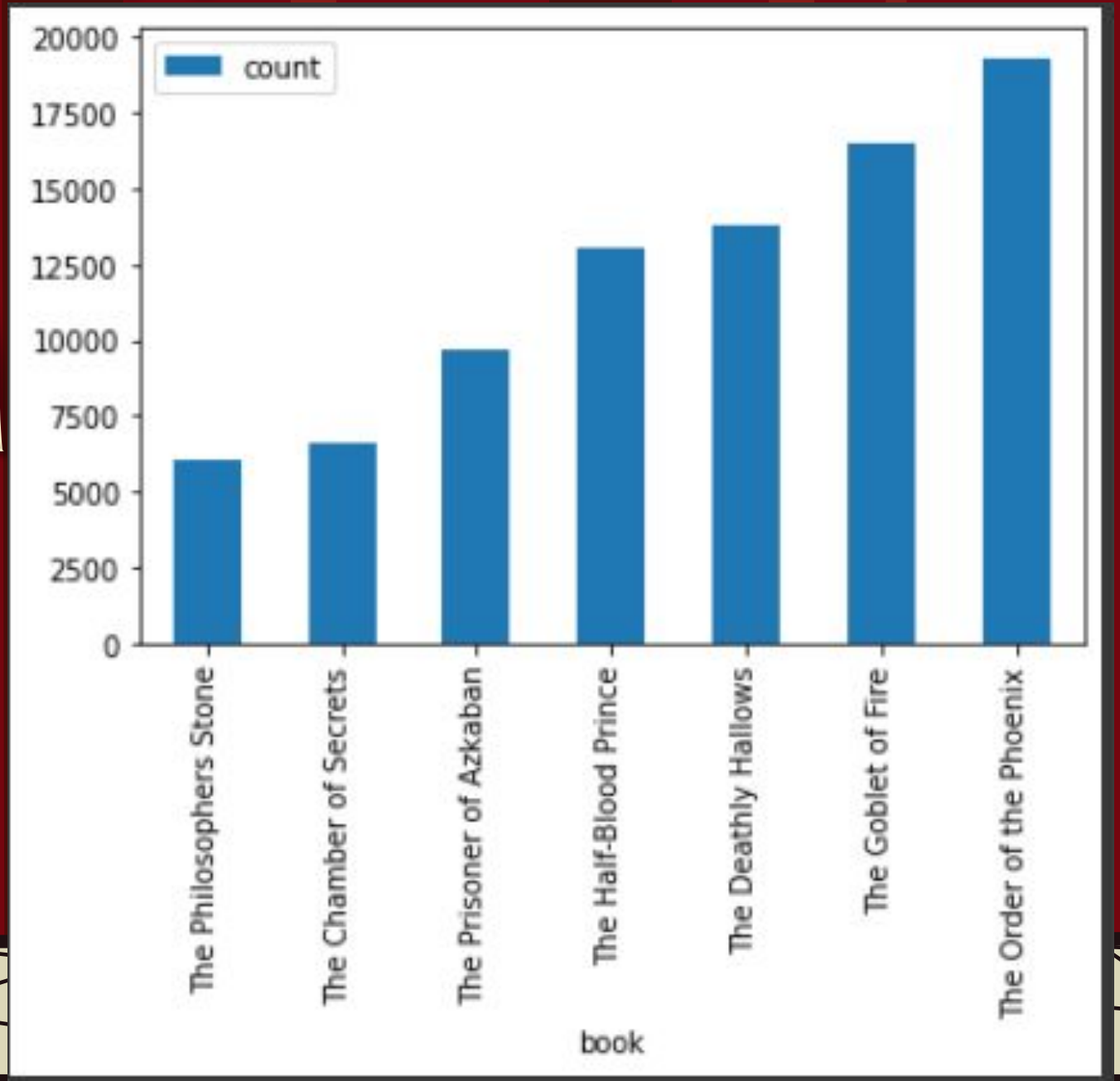
Or

```
pipeline_model.first()['sentences'][0].asDict()
```

```
+-----------------------+-----+
|book                   |count|
+-----------------------+-----+
|The Philosophers Stone |6023 |
|The Chamber of Secrets |6611 |
|The Prisoner of Azkaban|9664 |
|The Half-Blood Prince  |13033|
|The Deathly Hallows    |13795|
|The Goblet of Fire     |16532|
|The Order of the Phoenix|19320|
+-----------------------+-----+
```

```
{'annotatorType': 'document',
 'begin': 1,
 'embeddings': [],
 'end': 137,
 'metadata': {'sentence': '0'},
 'result': 'the boy who lived mr. and mrs. dursley of number four
```

Sentences Cont.



The Pipeline That Didn't Live

- Pretrained model for lemmatization
- Pretrained model for Named Entity Recognition (NER)
- Deep Learning techniques for NLP

```
root
 |-- text: string (nullable = true)
 |-- titles: string (nullable = true)
 |-- document: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- annotatorType: string (nullable = true)
 |    |    |-- begin: integer (nullable = false)
 |    |    |-- end: integer (nullable = false)
 |    |    |-- result: string (nullable = true)
 |    |    |-- metadata: map (nullable = true)
 |    |    |    |-- key: string
 |    |    |    |-- value: string (valueContainsNull = true)
 |    |    |-- embeddings: array (nullable = true)
 |    |    |    |-- element: float (containsNull = false)
 |-- sentences: array (nullable = false)
 |    |-- element: struct (containsNull = true)
 |    |    |-- annotatorType: string (nullable = true)
 |    |    |-- begin: integer (nullable = false)
 |    |    |-- end: integer (nullable = false)
 |    |    |-- result: string (nullable = true)
 |    |    |-- metadata: map (nullable = true)
 |    |    |    |-- key: string
 |    |    |    |-- value: string (valueContainsNull = true)
 |    |    |-- embeddings: array (nullable = true)
 |    |    |    |-- element: float (containsNull = false)
 |-- tokens: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- annotatorType: string (nullable = true)
 |    |    |-- begin: integer (nullable = false)
 |    |    |-- end: integer (nullable = false)
 |    |    |-- result: string (nullable = true)
 |    |    |-- metadata: map (nullable = true)
 |    |    |    |-- key: string
 |    |    |    |-- value: string (valueContainsNull = true)
 |    |    |-- embeddings: array (nullable = true)
 |    |    |    |-- element: float (containsNull = false)
```

SparkML

Words

- Removing stop words with SparkML
  - No easy way, used a stop words list https://www.ranks.nl/stopwords
  - StopWordsRemover()
    - InputCol
    - OutputCol
    - stopWords
- Clean using regex_replace
  - Special characters
  - Single letters
  - Extra spaces

Words Cont.

- This: `'the boy who lived mr. and mrs. dursley of number four privet drive were proud to say that they were perfectly normal thank you very much.'`

  Becomes

- This: `'boy', 'lived', 'dursley', 'number', 'privet', 'drive', 'perfectly', 'normal'`

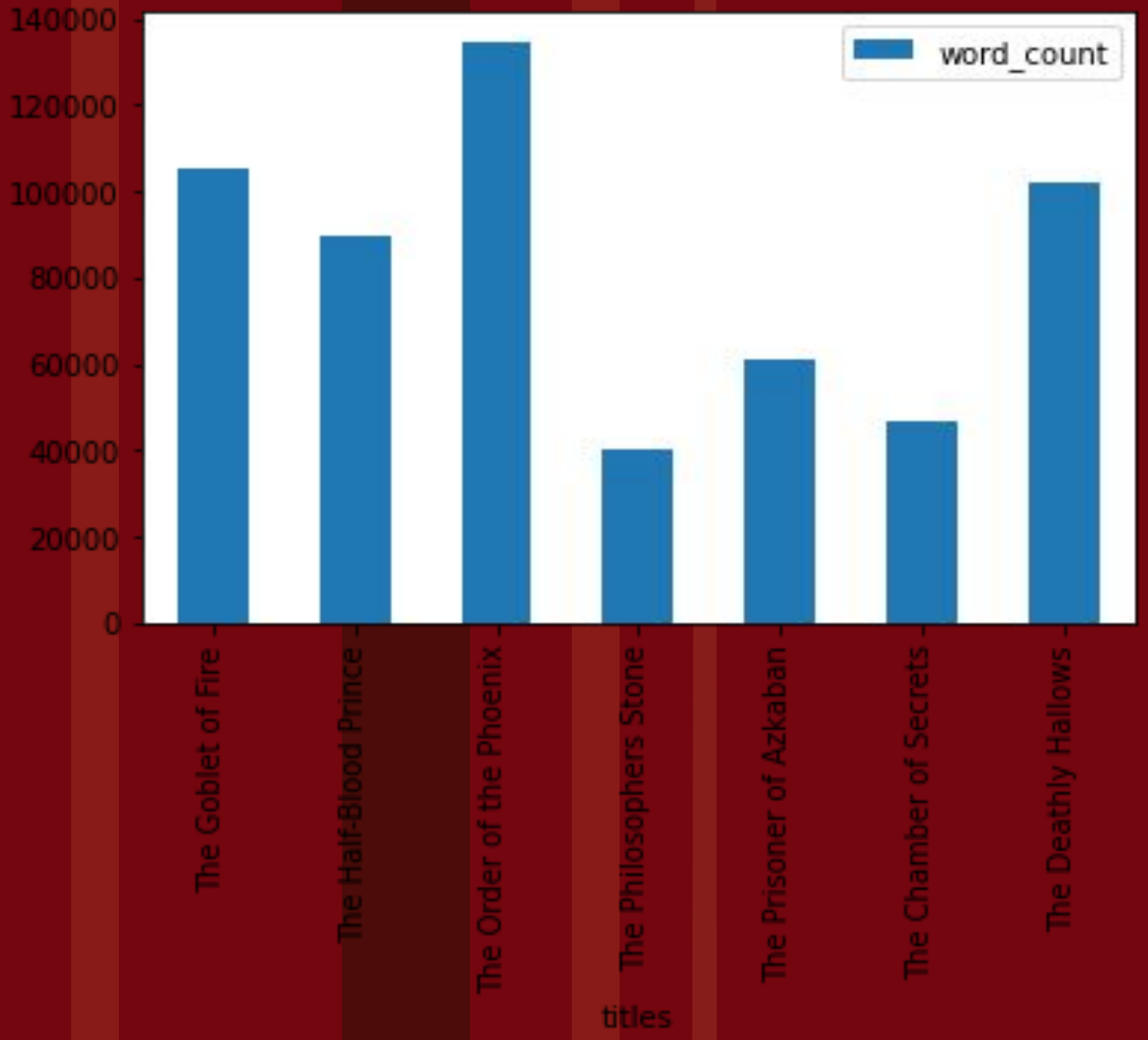
Before Stop Word Removal

titles	words	word_count
The Goblet of Fire	[the, riddle, hou...	224250
The Half-Blood Pr...	[j, the, other, m...	196841
The Order of the ...	[harry, potter, i...	299097
The Philosophers ...	[the, boy, who, l...	90600
The Prisoner of A...	[owl, post, harry...	127006
The Chamber of Se...	[j, ., k, ., r, o...	99933
The Deathly Hallows	[i, the, dark, lo...	227858

After Stop Word Removal

titles	words	word_count
The Goblet of Fire	[riddle, house, v...	106558
The Half-Blood Pr...	[minister, nearin...	90920
The Order of the ...	[harry, potter, d...	136715
The Philosophers ...	[boy, lived, durs...	40978
The Prisoner of A...	[owl, post, harry...	61658
The Chamber of Se...	[, , , , harry, p...	47250
The Deathly Hallows	[dark, lord, asce...	103288

Words Cont.


140000
120000
100000
80000
60000
40000
20000
0
word_count
The Goblet of Fire
The Half-Blood Prince
The Order of the Phoenix
The Philosophers Stone
The Prisoner of Azkaban
The Chamber of Secrets
The Deathly Hallows
titles

Words Cont.

- Harry : 21,920
- Ron: 6,329
- Hermione: 5,357
- Dumbledore: 3,365
- Hagrid: 2,042
- Professor: 2,034
- Snape: 1,827
- Time: 1,732
- Wand: 1,660

- Remaining issues:
  - Not all empty strings were removed
  - Still have single letters
  - Words like ‘rowling’

Topic Modelling

CountVectorizer

Determine the frequency (TF) of each term in the document

Get vocab during fit(), and counts during transform()

Inverse Frequency of Documents (IDF)

Accounts for words frequent across all documents

Uses Pyspark IDF estimator

Topic Modeling Using LDA (Latent Dirichlet allocation)

“a generative model that assumes that documents are represented by a distribution of topics and topics, in turn, are represented by a distribution of words”

Topics:
Selected arbitrary numbers and tried them out

topic	topicWords
0	[harry, wand, hermione, dumbledore, whispered, eyes, hand, sir, riddle, dobby]
1	[harry, ron, hermione, chamber, secrets, ginny, yeah, fred, desk, george]
2	[harry, time, uncle, vernon, ron, didn, told, aunt, haven, hermione]
3	[harry, well, professor, dark, hallows, deathly, ron, hermione, going, potter]
4	[harry, hermione, ron, potter, looked, k, hagrid, weasley, stone, malfoy]
5	[potter, k, harry, order, phoenix, half, fire, goblet, azkaban, blood]
6	[harry, ron, quietly, asked, looked, hermione, neville, dumbledore, er, hear]

Future Considerations

- Unique words per book
- Most common words per book
- Count occurrences of character appearances/mentions for each book
- Topic by book rather than

- Count character combinations in each sentence
  - Relationship estimations
- Fix instances of empty strings/random words
- lemmatization


Thank you!

Credits

- Presentation Template: SlidesMania
- https://medium.com/trustyou-engineering/topic-modelling-with-pyspark-and-spark-nlp-a99d063f1a6e
- https://medium.com/spark-nlp/spark-nlp-101-document-assembler-500018f5f6b5
- https://www.ranks.nl/stopwords
- https://github.com/formcept/whiteboard/tree/master/nbviewer/notebooks/data/harry potter