Using text reviews to guide book marketing strategies

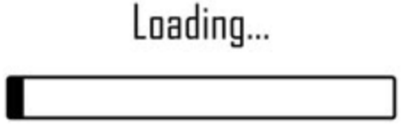
D. Defforey

Photo credit: Susan Yin

Photo credit: CLU/Getty Images

Should retailers and publishers develop distinct marketing strategies for books from different genres?

Overall Project Goals



Load the LibThing dataset from UCSD[1,2]

Web scraping & APIs

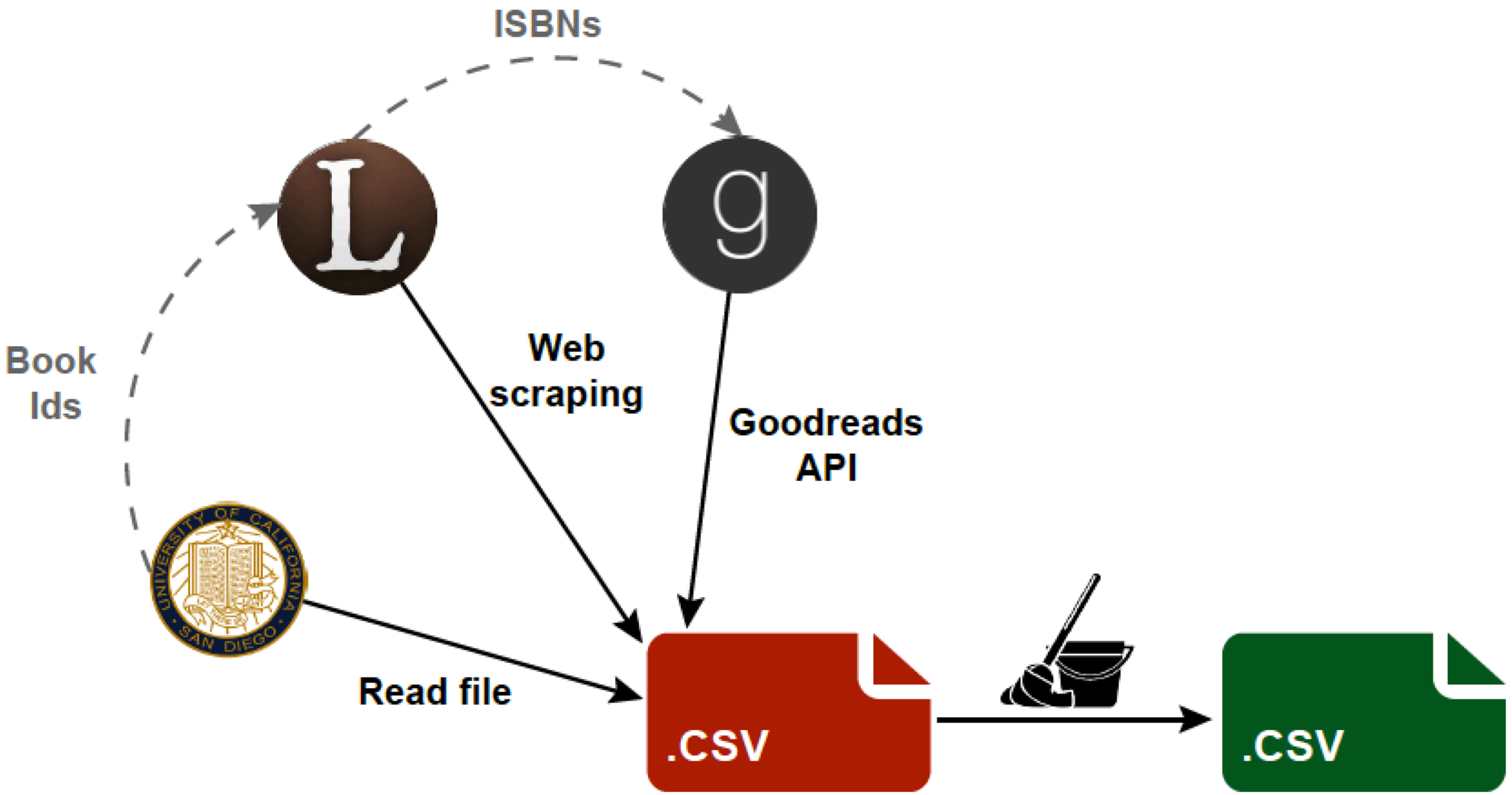
Data preprocessing & LDA

Interpretation & business outcomes

Images from: www.videoblocks.com, www.grepsr.com, www.udemy.com, www.brainfacts.com

[1,2] McAuley *et al.*, 2015, 2017

Data collection & cleaning



Regex & .replace()

Dealing with a pseudo-JSON file

“

re.sub(r"(?<={|\s)'|'(?=,|:|})", "", line)

Replace single quotes around keys/values with double quotes

\

re.sub(r'(?<!{)(?<!,\s|:\s)\"(?!,|:|})', '\\\"', line)

Add a backslash in front of all double quotes within the text

‘

line.replace("\\\"", "'")

Remove all backslash in front of single quotes

```
{
‘work’: ‘3067’,
‘flags’: [ ],
‘stars’: 4.5,
‘nhelpful’: 0,
‘time’: ‘Oct 8, 2006’,
‘comment’: ‘It\’s a great book with “revolutionary” ideas but gets confusing at times’.
‘user’: ‘EvilPlatypus’
}
```

Regex & .replace()

Dealing with a pseudo-JSON file

“ **re.sub(r"(?<={|\s)'|'(?=,|:|})", "", line)**

Replace single quotes around keys/values with double quotes

\ **re.sub(r'(?<!{)(?<!,\s|:\s)\"(?!,|:|})', '\\\"', line)**

Add a backslash in front of all double quotes within the text

‘ **line.replace("\\\"", "\"")**

Remove all backslash in front of single quotes

{

‘work’: ‘3067’,

‘flags’: [ ],

‘stars’: 4.5,

‘nhelpful’: 0,

‘time’: ‘Oct 8, 2006’,

‘comment’: ‘It\’s a great book with “revolutionary” ideas but gets confusing at times’.

‘user’: ‘EvilPlatypus’

}

Regex & .replace()

Dealing with a pseudo-JSON file

“ **re.sub(r"(?<={|\s)'|'(?=,|:|})", "", line)**

Replace single quotes around keys/values with double quotes

\ **re.sub(r'(?<!{)(?<!,\s|:\s)\"(?!,|:|})', '\\\"', line)**

Add a backslash in front of all double quotes within the text

‘ **line.replace("\\"", "")**

Remove all backslash in front of single quotes

{

“work”: “3067”,

“flags”: [ ],

“stars”: 4.5,

“nhelpful”: 0,

“time”: “Oct 8, 2006”,

“comment”: “It\’s a great book with “revolutionary” ideas but gets confusing at times”.

“user”: “EvilPlatypus”

}

Regex & .replace()

Dealing with a pseudo-JSON file

“

re.sub(r"(?<={|\s)'|'(?=,|:|})", "", line)

Replace single quotes around keys/values with double quotes

\

re.sub(r'(?<!{)(?<!,\s|:\s)\"(?!,|:|})', '\\\"', line)

Add a backslash in front of all double quotes within the text

‘

line.replace("\\"", """)

Remove all backslash in front of single quotes

{

“work”: “3067”,

“flags”: [ ],

“stars”: 4.5,

“nhelpful”: 0,

“time”: “Oct 8, 2006”,

“comment”: “It\’s a great book with \“revolutionary\” ideas but gets confusing at times”.

“user”: “EvilPlatypus”

}

Regex & .replace()

Dealing with a pseudo-JSON file

“ **re.sub(r"(?<={|\s)'|'(?=,|:|})", "", line)**

Replace single quotes around keys/values with double quotes

\ **re.sub(r'(?<!{)(?<!,\s|:\s)\"(?!,|:|})', '\\\"', line)**

Add a backslash in front of all double quotes within the text

‘ **line.replace("\\"", "")**

Remove all backslash in front of single quotes

{

“work”: “3067”,

“flags”: [ ],

“stars”: 4.5,

“nhelpful”: 0,

“time”: “Oct 8, 2006”,

“comment”: “It’s a great book with \“revolutionary\” ideas but gets confusing at times”.

“user”: “EvilPlatypus”

}

Regex & .replace()

Dealing with a pseudo-JSON file

“ **re.sub(r"(?<={|\s)'|'(?=,|:|})", "", line)**

Replace single quotes around keys/values with double quotes

\ **re.sub(r'(?<!{)(?<!,\s|:\s)\"(?!,|:|})', '\\\"', line)**

Add a backslash in front of all double quotes within the text

‘ **line.replace("\\"", "")**

Remove all backslash in front of single quotes

{

“work”: “3067”,

“flags”: [ ],

“stars”: 4.5,

“nhelpful”: 0,

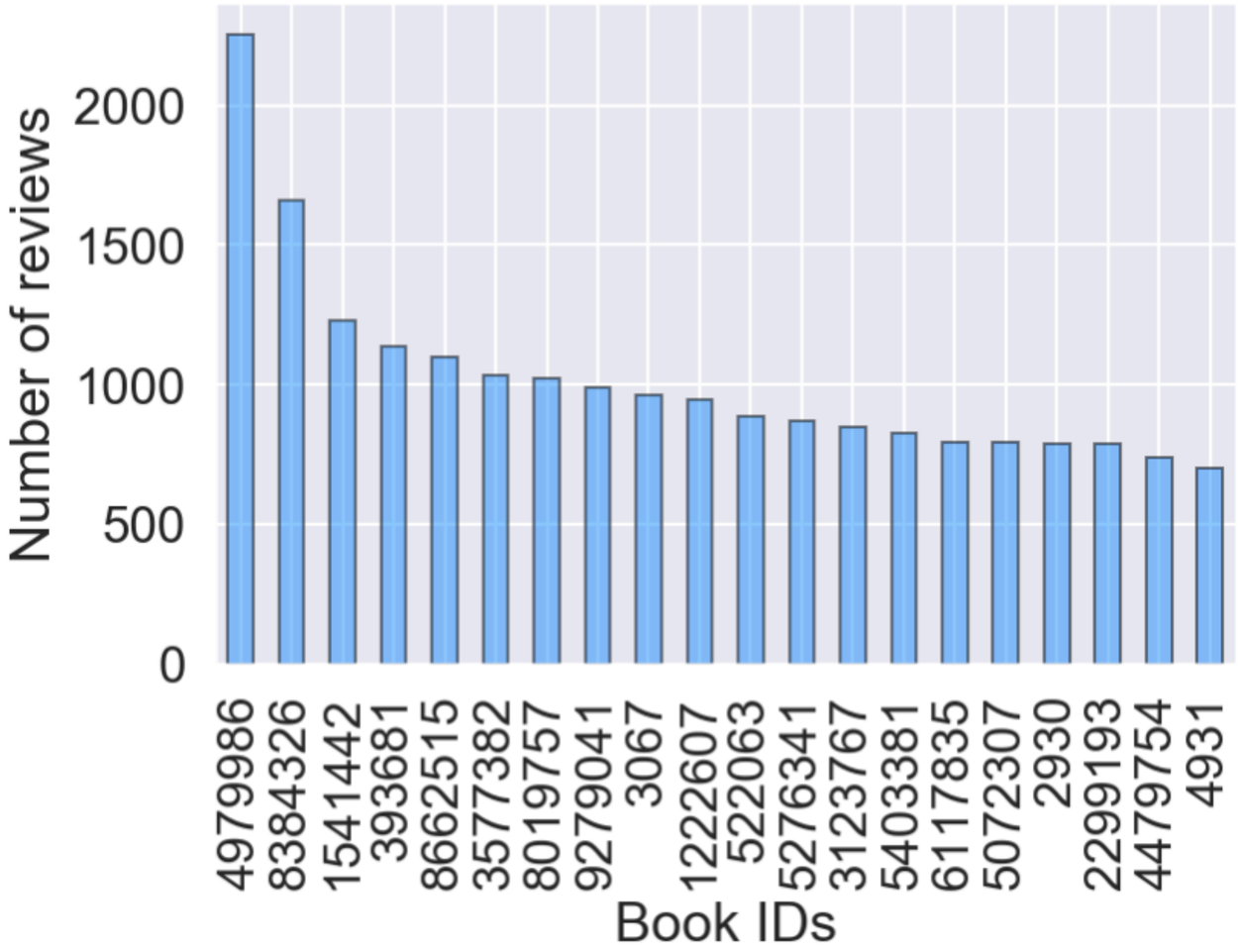
“time”: “Oct 8, 2006”,

“comment”: “It’s a great book with \“revolutionary\” ideas but gets confusing at times”.

“user”: “EvilPlatypus”

}

Top 5K most-reviewed books

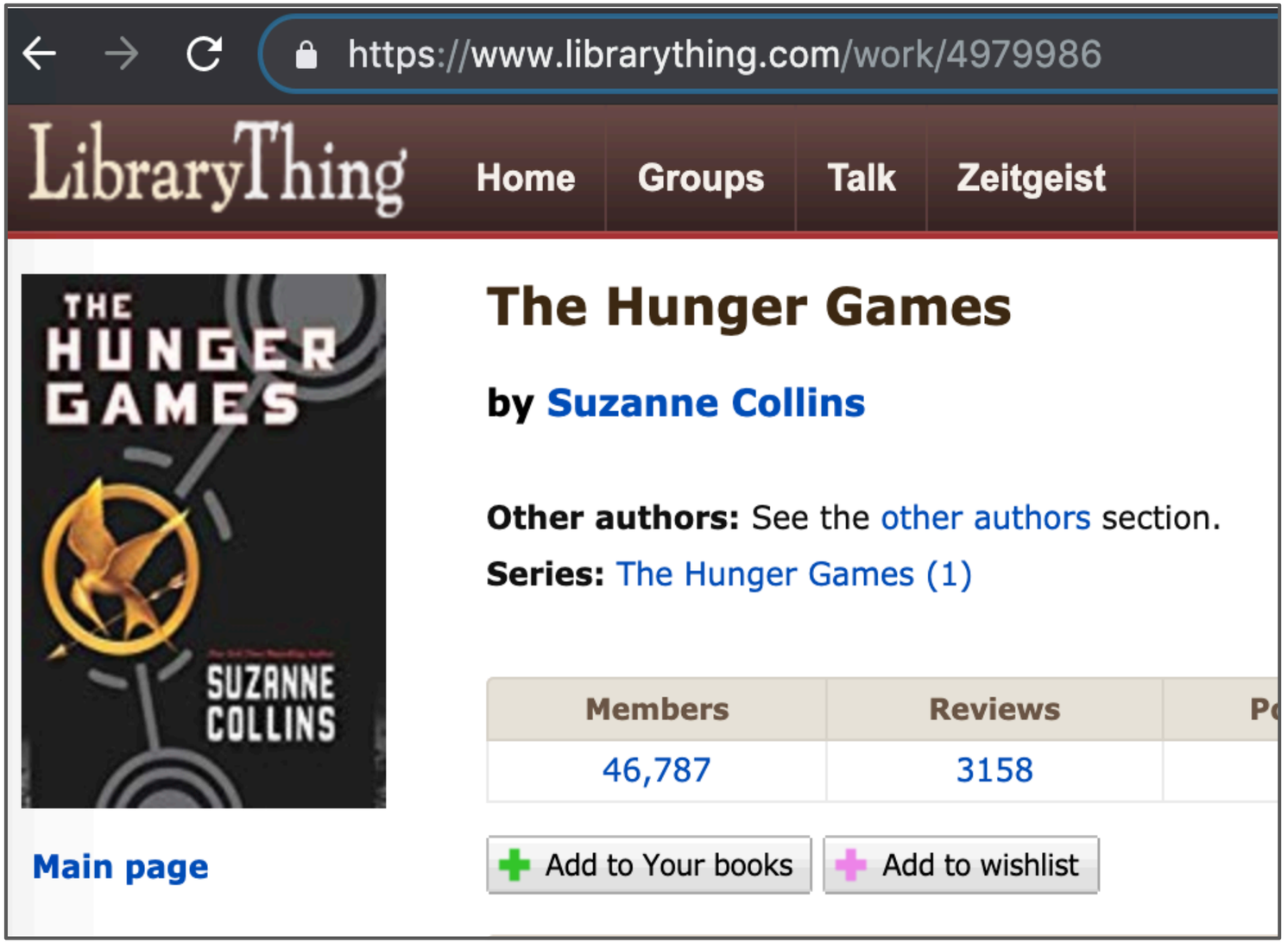



```
top_5k_books = (
    library_thing
    .id
    .value_counts()
    .head(5000)
    .reset_index()
    .rename(columns={‘index’:
‘book_id’})
)
```

Web scraping

Getting more info from LibraryThing.com

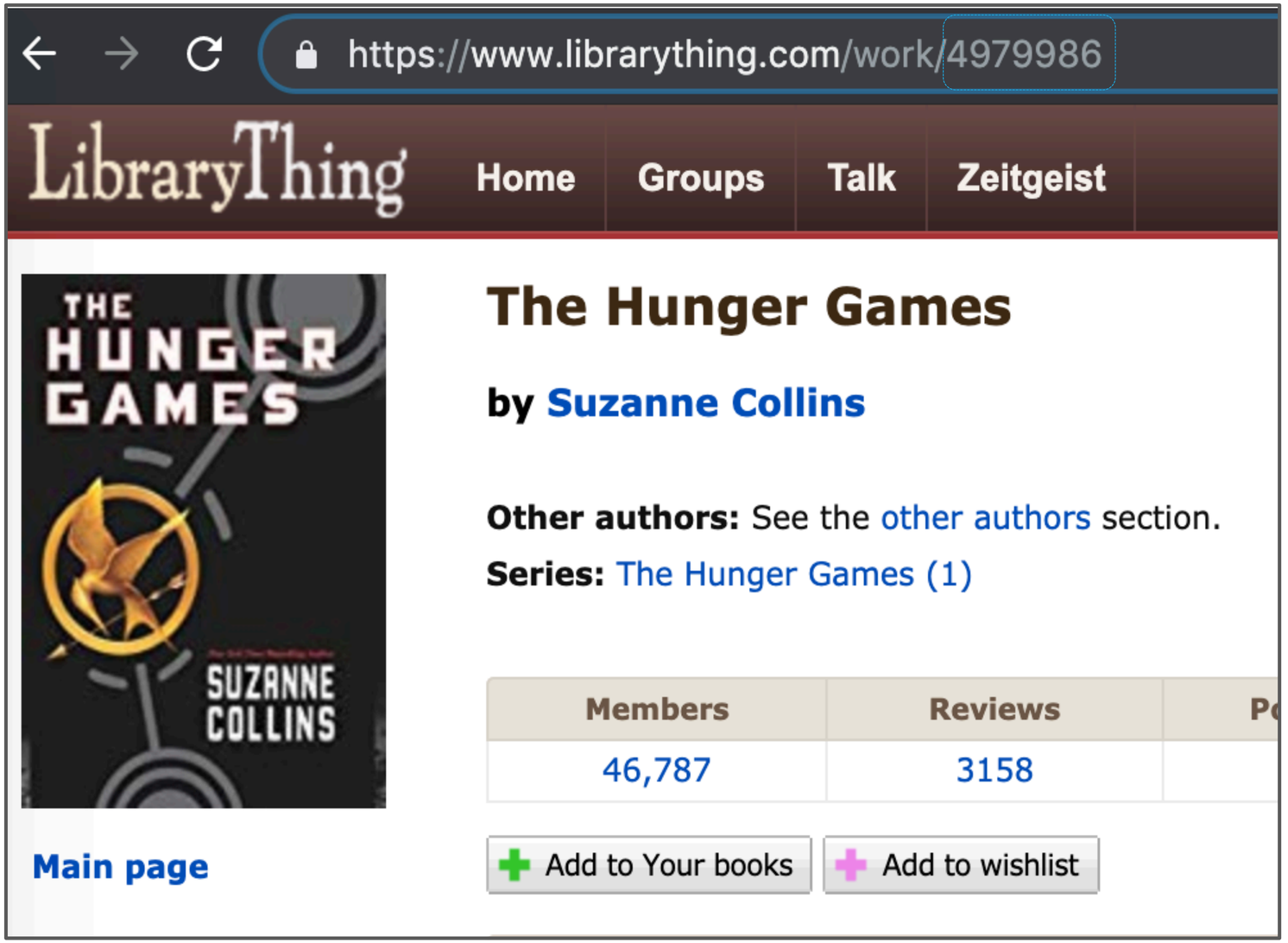
- Web requests
- Wrote scraped htmls to CSVs
- Parsed them with BeautifulSoup
- More cleaning with regex



Web scraping

Getting more info from LibraryThing.com

- Web requests
- Wrote scraped htmls to CSVs
- Parsed them with BeautifulSoup
- More cleaning with regex



Goodreads API

To get book genre information

01

Use ISBNs to get Goodreads IDs

These IDs are needed to look up Goodreads book shelves

02

Use Goodreads IDs to get book titles

This is a precaution to check that the IDs are correct

03

Use Goodreads IDs to get book shelves

These include book genres, but also unnecessary categories that need to be removed

Example of bookshelves for one book

[{‘@name’: ‘to-read’, ‘@count’: 870985},

{‘@name’: ‘currently-reading’, ‘@count’: 21136},

{‘@name’: ‘young-adult’, ‘@count’: 18033},

{‘@name’: ‘fantasy’, ‘@count’: 16141},

{‘@name’: ‘romance’, ‘@count’: 12163},

{‘@name’: ‘fiction’, ‘@count’: 10147},

{‘@name’: ‘vampires’, ‘@count’: 8283},

{‘@name’: ‘bella’, ‘@count’: 4421}]

Goodreads API

To get book genre information

01

Use ISBNs to get Goodreads IDs

These IDs are needed to look up Goodreads book shelves

02

Use Goodreads IDs to get book titles

This is a precaution to check that the IDs are correct

03

Use Goodreads IDs to get book shelves

These include book genres, but also unnecessary categories that need to be removed

Example of bookshelves for one book

[{‘@name’: ‘to-read’, ‘@count’: 870985},

{‘@name’: ‘currently-reading’, ‘@count’: 21136},

{‘@name’: ‘young-adult’, ‘@count’: 18033},

{‘@name’: ‘fantasy’, ‘@count’: 16141},

{‘@name’: ‘romance’, ‘@count’: 12163},

{‘@name’: ‘fiction’, ‘@count’: 10147},

{‘@name’: ‘vampires’, ‘@count’: 8283},

{‘@name’: ‘bella’, ‘@count’: 4421}]

Goodreads API

To get book genre information

01

Use ISBNs to get Goodreads IDs

These IDs are needed to look up Goodreads book shelves

02

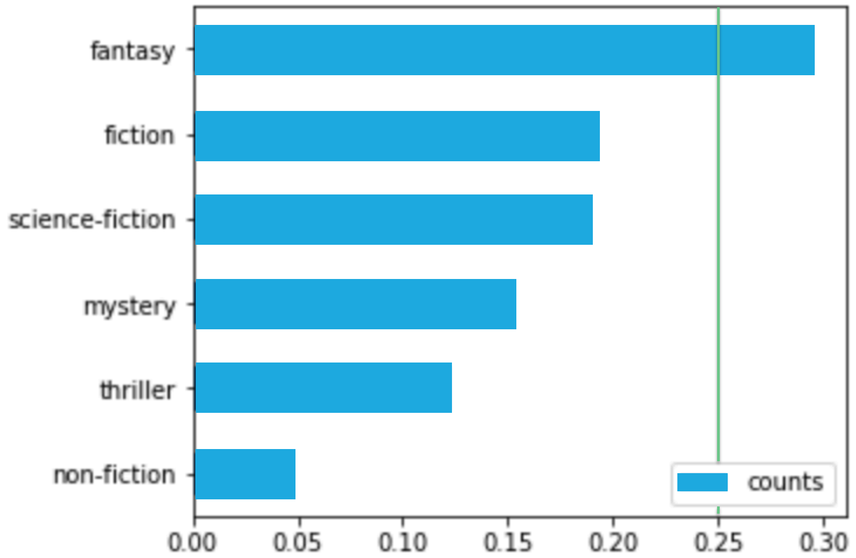
Use Goodreads IDs to get book titles

This is a precaution to check that the IDs are correct

03

Use Goodreads IDs to get book shelves

These include book genres, but also unnecessary categories that need to be removed



Final dataset

After much cleaning

	reviews	n_helpful	time	user	id	book_title	author	isbn_x	book_genres
0	This a great book for young readers to be intr...	0	Nov 7, 2007	van_stef	3206242	The Hobbit (1937)	J. R. R. Tolkien	0618260307	fantasy
1	Not as great a literary work as Tolkien's famo...	1	Aug 10, 2007	EvilPlatypus	3206242	The Hobbit (1937)	J. R. R. Tolkien	0618260307	fantasy
2	Bilbo Baggins becomes an accidental hero in th...	0	Dec 3, 2012	ewyatt	3206242	The Hobbit (1937)	J. R. R. Tolkien	0618260307	fantasy
3	I found the Hobbit when I was in the 5th grade...	0	May 11, 2007	Nikkles	3206242	The Hobbit (1937)	J. R. R. Tolkien	0618260307	fantasy

272,599 rows

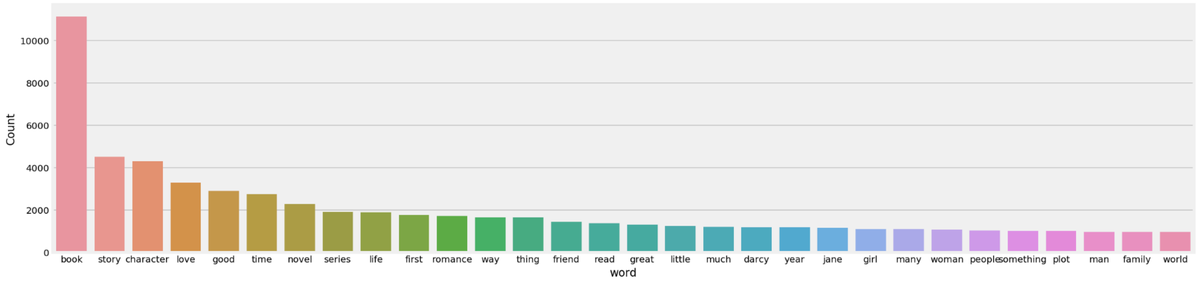
&

9 columns

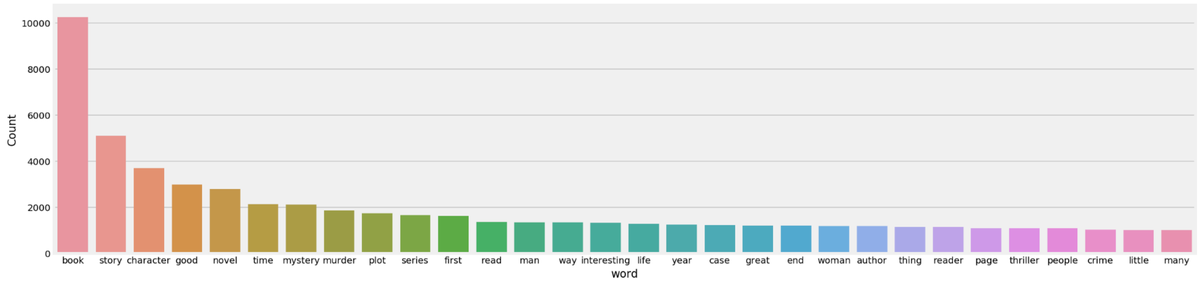
Exploratory Data Analysis

A brief overview

Romance

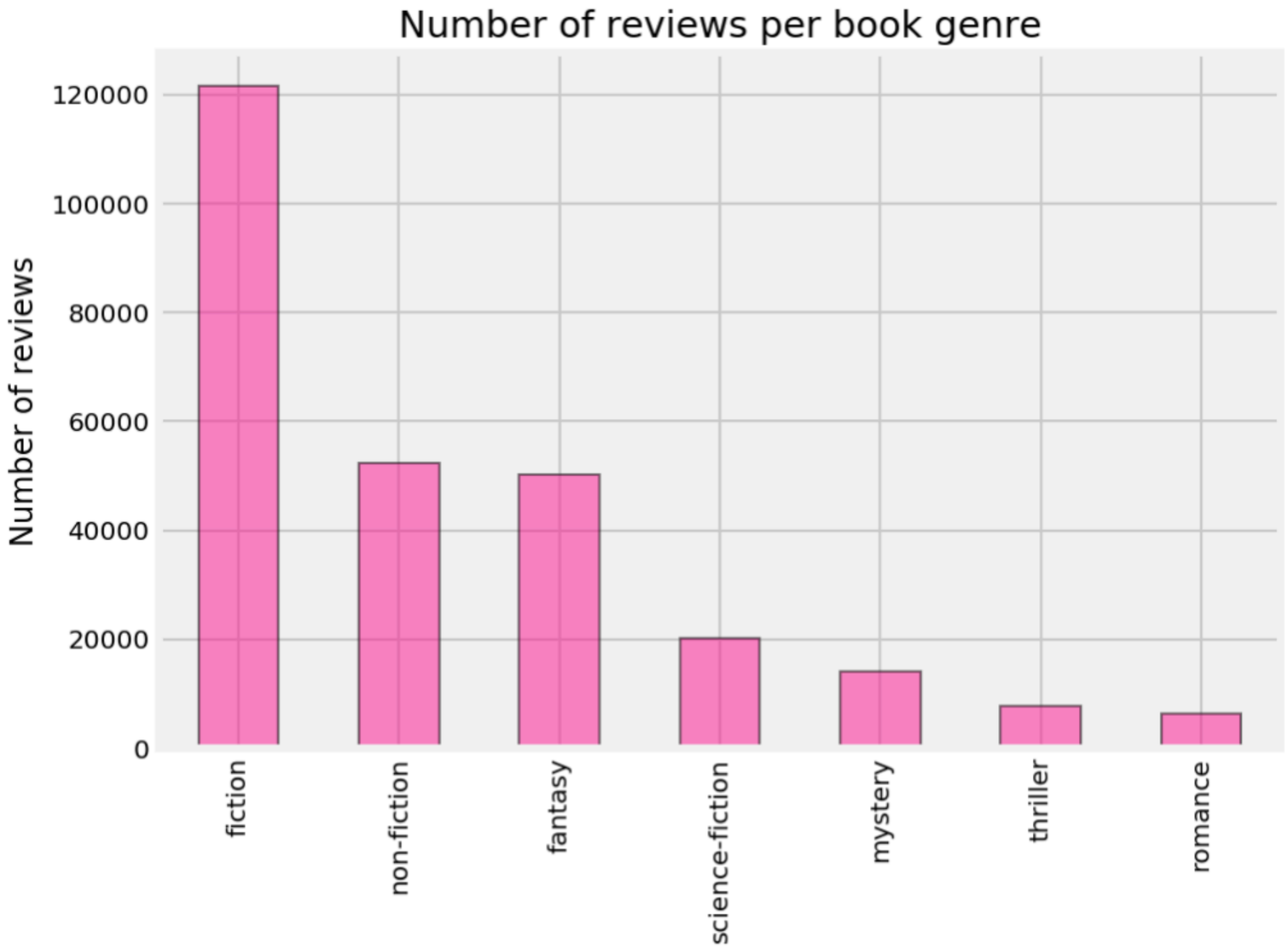


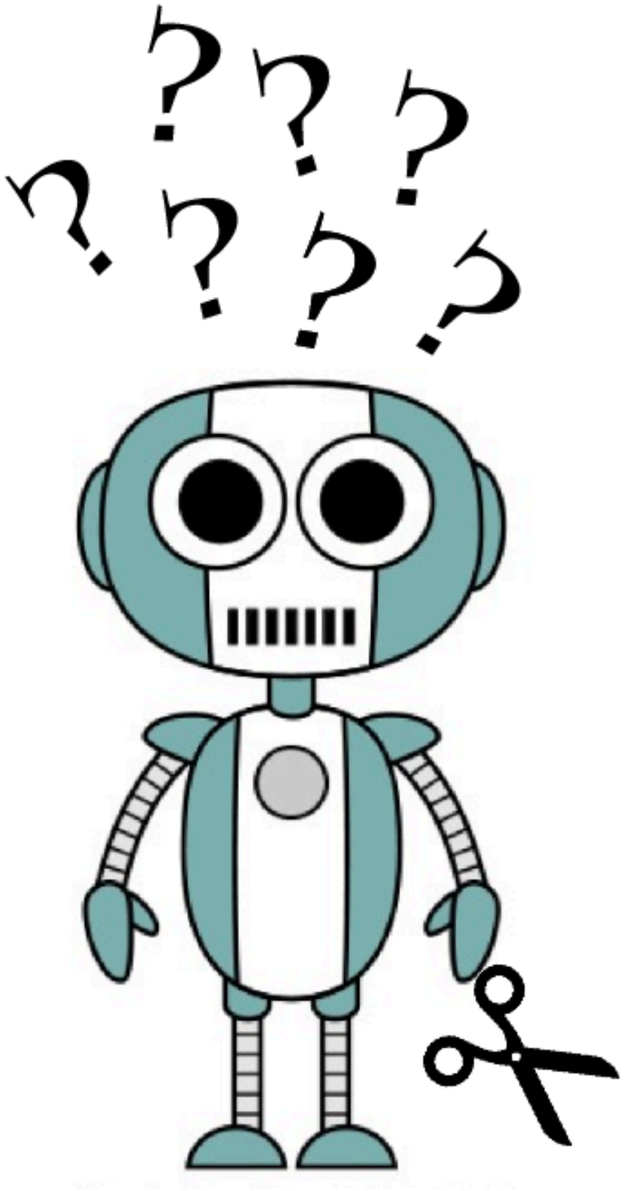
Thriller

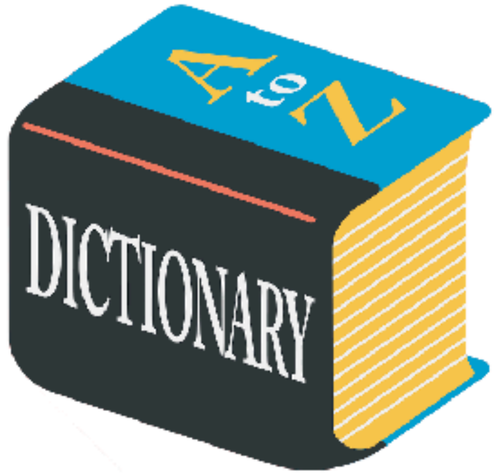


Exploratory Data Analysis

A brief overview







Text preprocessing

Tokenisation

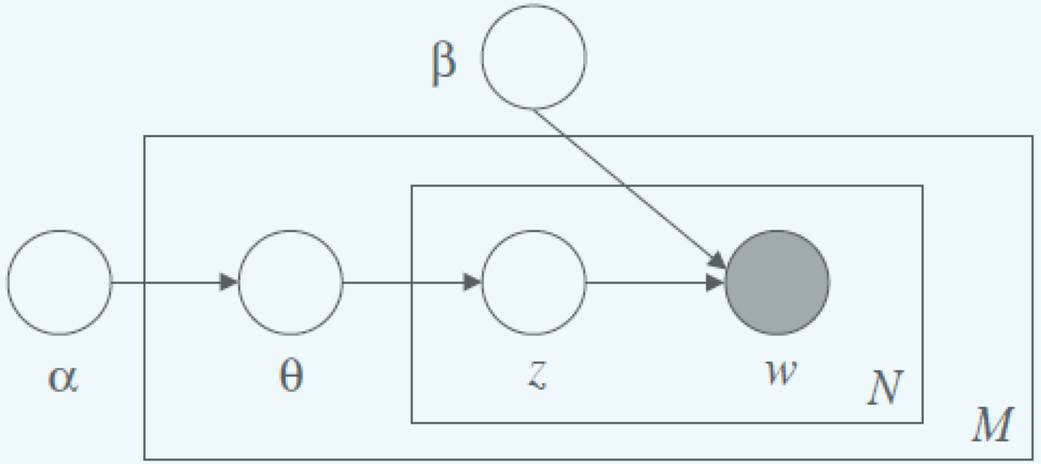
Stopwords removal

Lemmatisation

Robot from: www.how-to-draw-funny-cartoons.com, dictionary from: www.appbajar.com

Latent Dirichlet Allocation[3]

In a nutshell



Why?

Primarily concerned with topic modelling of text. Humans can easily glean topics in documents, but this is difficult to automate.

What?

LDA is a generative statistical model capable of generating documents of text. In practice, most often used to extract topics.

How?

Fit the LDA model to a specific corpus, then inspect its parameters (including topics).

[3]Blei *et al.*, 2003

This may help…

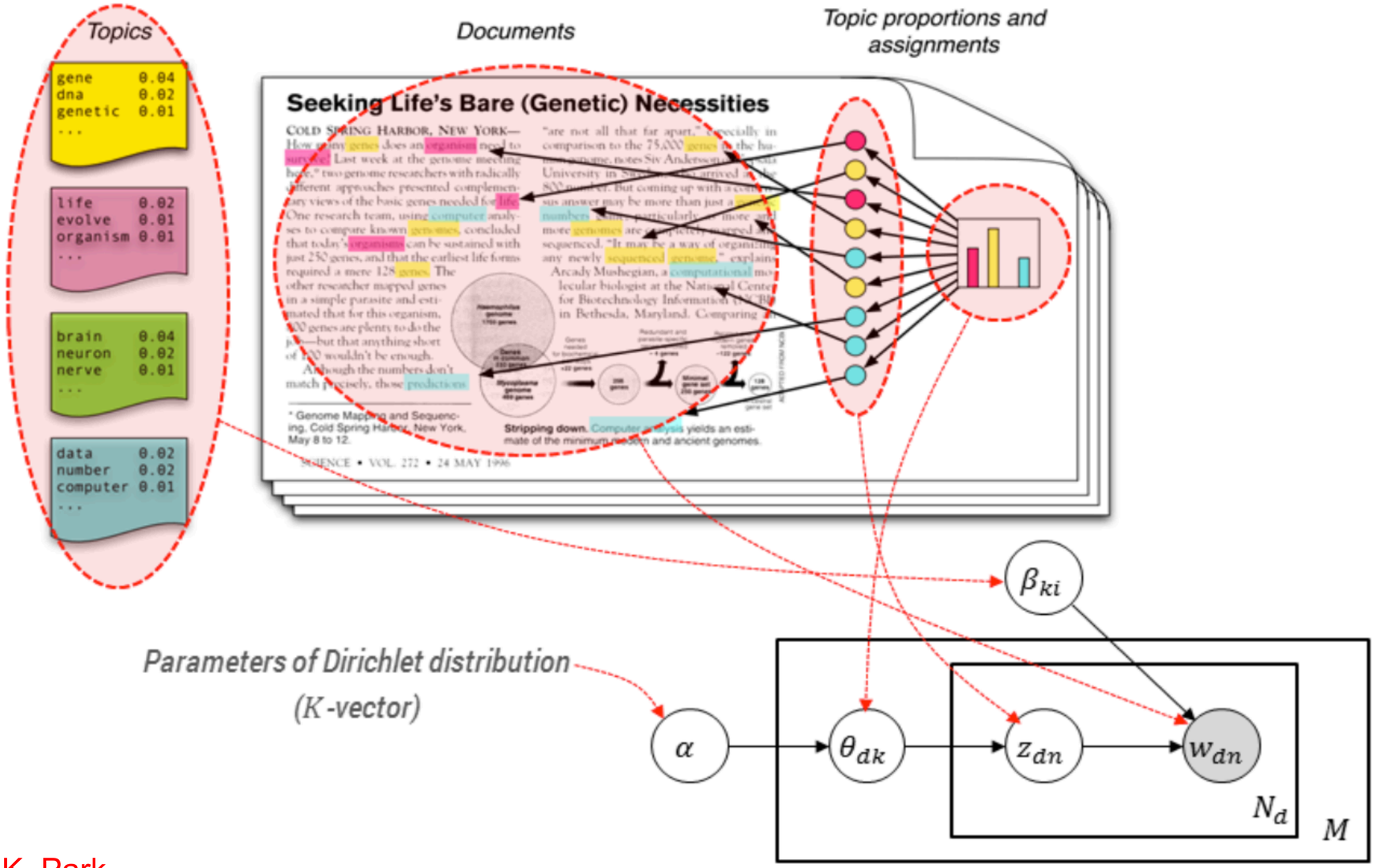


Image credit: ChangUK, Park

Latent Dirichlet Allocation

Main takeaway points

Assumptions & Terminology

Documents with similar topics will use similar groups of words

Documents are probability distributions over latent topics

Topics are probability distributions over words

Output

Topics generated do not have human readable labels.

These can be manually added but are not necessary.

One can calculate similarities between topics without these labels.

Source: Scott Sullivan, LDA algorithm description

Gensim implementation

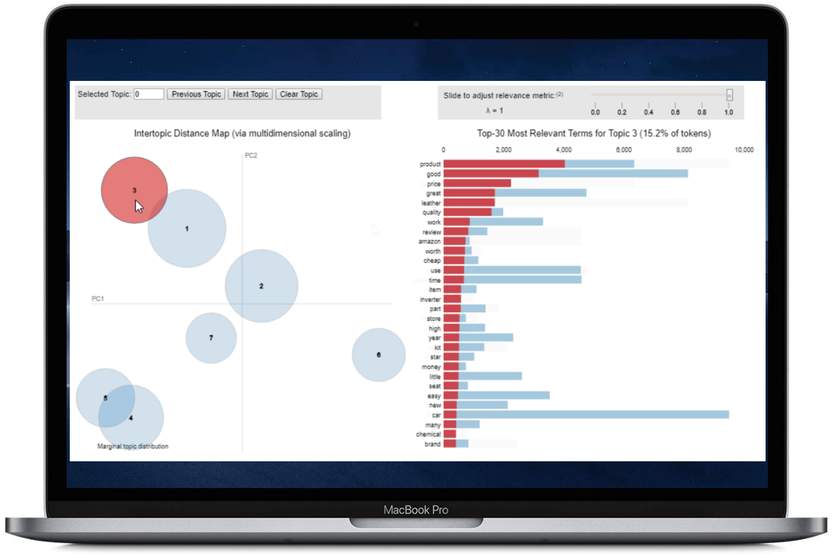
Your subtitle goes here

corpus & id2word

processed words (tokenised and lemmatised) and the mapping of word ids to words

number of topics

As many as you like, but keep in mind the length of the corpus and human interpretability.



lists of topics

These have unique identifiers but require additional labelling if one wants topic names

pyLDAvis[4]

A web-based interactive visualisation of topics estimated using LDA. Facilitates topic interpretation

Images from: www.analyticsvidhya.com, www.amazon.co.uk

[4]Sievert and Shirley, 2014

Success metrics

Create an LDA model that is generalisable & good topic coherence

Identify distinct, interpretable topics within text reviews of books

Translate those into a marketing strategy

Key findings

From model #5 out of 10

Model parameters and performance metrics

- ✧ Bigrams
- ✧ 20 topics
- ✧ 10 corpus passes
- ✧ Perplexity: -12.46
- ✧ C_v coherence score: 0.41

Highlights of labelled topics and keywords

1. High fantasy: ‘world’, ‘magic’, ‘dark’, ‘evil’, ‘fantasy’
2. Children’s books: ‘child’, ‘story’, ‘picture’, ‘illustration’
5. War: ‘war’, ‘death’, ‘battle’, ‘history’, ‘soldier’, ‘military’
6. Mystery/thriller: ‘mystery’, ‘plot’, ‘case’, ‘murder’, ‘twist’
14. Family: ‘family’, ‘mother’, ‘father’, ‘sister’, ‘brother’
15. Romance: ‘love’, ‘romantic’, ‘relationship’, ‘heart’
17. Children + young adults: ‘teen’, ‘school’, ‘boy’, ‘girl’

Interactive visualisation

Some limitations

There are a few

Book genre allocation

Involved many simplifications

Mostly focused on fiction

All non-fiction lumped into one category

Manual topic labelling

Subjective process

Labelled topics should be taken with a pinch of salt

Subject to change if the model gets updated

Corpus size

Fairly small corpus

Diversity of words could be improved

Recommendations for book marketing strategies

- Consumer text reviews of book reveal interesting insights into what they care about
- Some but not all book genres are included in the LDA topics, suggesting that genre-specific marketing materials could be worth pursuing
- Target demographics are also important, especially children and young adults

Next steps

01 Manually label missing ISBNs to increase the corpus size

02 Update the LDA model and optimise model performance

03 Use the model to predict dominant topics in reviews

04 Create a web interface where users can input reviews, then retrieve dominant topics

References

1. SPMC: Socially-aware personalized Markov chains for sparse sequential recommendation. Chenwei Cai, Ruining He, Julian McAuley. *IJCAI*, 2017.

2. Improving latent factor models via personalized feature projection for one-class recommendation. Tong Zhao, Julian McAuley, Irwin King. *Conference on Information and Knowledge Management (CIKM)*, 2015

3. Latent Dirichlet Allocation. David M. Blei, Andrew Y. Ng, Michael I. Jordan. *Journal of Machine Learning Research* 3, 993-1022, 2003.

4. LDAvis: A method for visualizing and interpreting topics. Carson Sievert, Kenneth E. Shirley. *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, 63-70, 2014

Thank you!

Questions?

Photo credit: Susan Yin