

BOOK RECOMMENDATION SYSTEM

Yaqeen Al Mahdi

Outline

2



Problem Description



Data Preprocessing



Data Acquisition



Modeling Experiments



Scope

Problem Description

3

goodreads

Home My Books Browse ▾ Community ▾ Search books

CURRENTLY READING

How to Argue With a Racist by Adam Rutherford

Clean Code: A Handbook of Agile Software Construction by Robert C. Martin

Psychology for Busy People by Joel Levy

See more • Add a book • General update

2021 READING CHALLENGE

2021
READING CHALLENGE

10
books completed
6 books behind schedule
10/20 (50%)
View Challenge

WANT TO READ

UPDATES

Fatima A. Alsafai started reading **Lakewood** by Megan Giddings 3h

Want to Read Rate it: ★★★★★

A startling debut about class and race, Lakewood evokes a terrifying world of medical experimentation—part *The Handmaid's Tale*, part *The Immortal Life of Henrietta Lacks*. Whe... Continue reading

Like • Comment

Colleen Dilenschneider liked this

Write a comment...

Nouf AlOtaibi made progress on **Bad Blood: Secrets and Lies in a Silicon Valley Startup** 3h

On page 174 of 339

BAD BLOOD
Secrets and Lies in a Silicon Valley Startup
John Carreyrou

Want to Read Rate it: ★★★★★

The full inside story of the breathtaking rise and shocking collapse of a multibillion-dollar startup, by the prize-winning journalist who first broke the story and pursued it... Continue reading

Customize

west elm

Romi 5-Piece Sectional 6,201 SAR

Hamilton Leather Sofa (81") 8,750 SAR

Slatted Buffet - Walnut 2,325 SAR

NEWS & INTERVIEWS

21 Fall Debut Novels to Read Now

FAULT LINES by Emily Tamkin

IN EVERY MIRROR SHE'S BLACK by Leah H. Williams

24 likes • 21 comments

RECOMMENDATIONS

Because you enjoyed Harry Potter and the Deathly Hallows (Harry Potter, #7):

Breaking Dawn (The Twilight Saga, #4) by Stephenie Meyer

Problem Description



Clean Code: A Handbook of Agile Software Craftsmanship
(Robert C. Martin Series)
by Robert C. Martin
★★★★★ 4.39 · Rating details · 17,244 ratings · 1,041 reviews

Even bad code can function. But if code isn't clean, it can bring a development organization to its knees. Every year, countless hours and significant resources are lost because of poorly written code. But it doesn't have to be that way.

Noted software expert Robert C. Martin presents a revolutionary paradigm with *Clean Code: A Handbook of Agile Software Craftsmanship*. M ...[more](#)

GET A COPY

[Amazon](#) [Online Stores](#) [Libraries](#)

Paperback, 464 pages
Published August 1st 2008 by Pearson (first published January 1st 2007)

Original Title: Clean Code: A Handbook of Agile Software Craftsmanship (Robert C. Martin Series)
ISBN: 0132350882 (ISBN13: 9780132350884)
Edition Language: English
Series: Robert C. Martin Series
Other Editions (33) 

[All Editions](#) | [Add a New Edition](#) | [Combine](#)

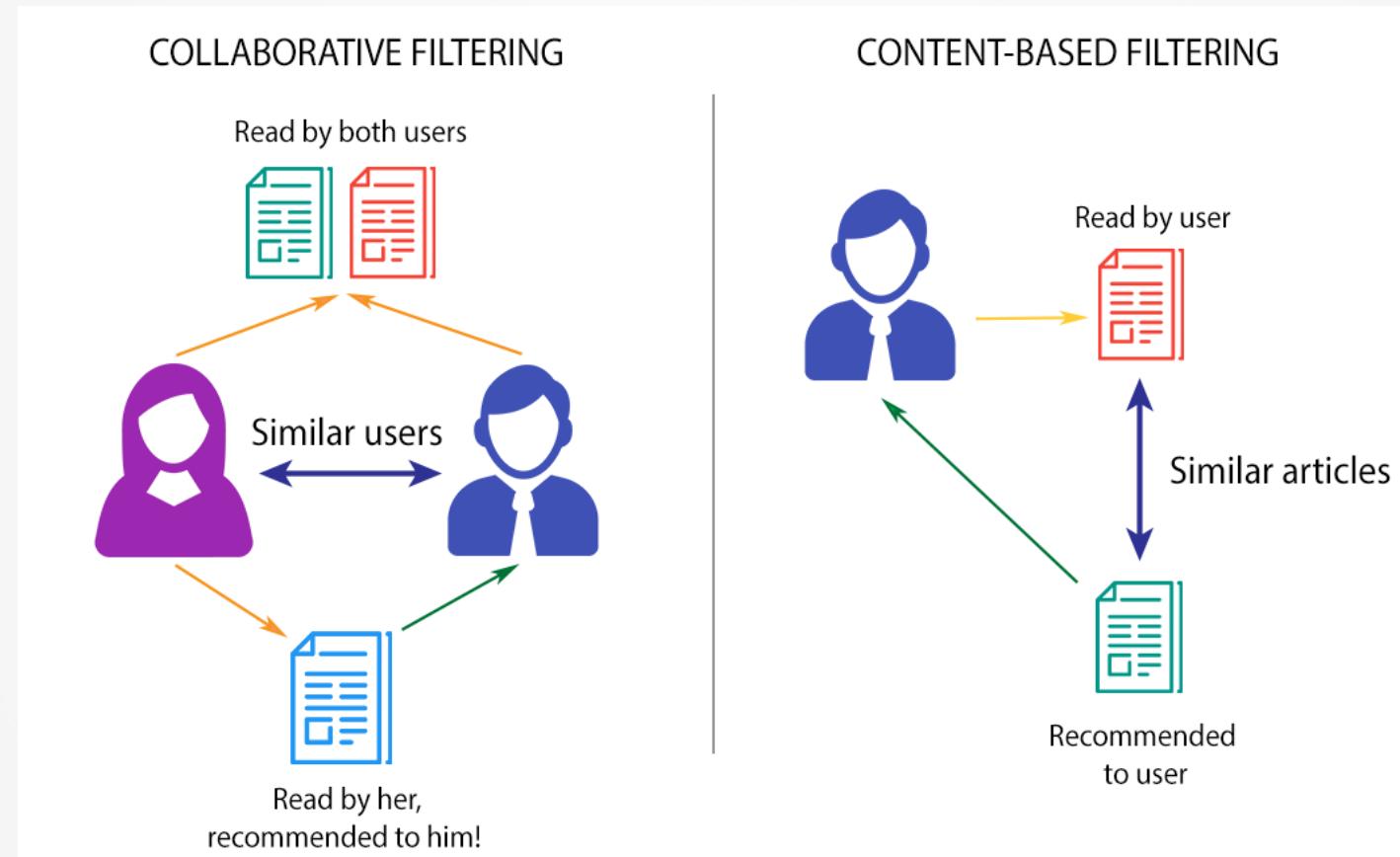
[...Less Detail](#) [Edit Details](#)

GENRES

Computer Science > Programming	1,416 users
Science > Computer Science	532 users
Science > Technology	436 users
Computer Science > Software	339 users
Computer Science > Technical	289 users
Nonfiction	283 users
Computer Science > Coding	191 users
Science > Engineering	76 users
Computer Science > Computers	65 users
Reference	45 users

[See top shelves...](#)

Problem Description



Data Acquisition

The screenshot shows the Kaggle interface for a dataset titled "Goodreads-books". The left sidebar includes links for "Create", "Home", "Competitions", "Datasets" (which is selected), "Code", "Discussions", "Courses", and "More". Under "Recently Viewed", there are links for "Clustering docu...", "Goodreads-books", "Goodreads Book ...", and "Harry Potter and ...". The main content area features a large "goodreads" logo with a count of 1355 datasets. Below the logo, the dataset title is "Goodreads-books" with the subtitle "comprehensive list of books listed in goodreads". It has a "Data" tab selected, along with "Tasks (1)", "Code (133)", "Discussion (10)", "Activity", and "Metadata". A "Download (2 MB)" button and a "New Notebook" button are also present. The dataset has a "Usability 10.0", "License CC0: Public Domain", and "Tags business, internet, literature, linguistics, demographics". The "Description" section is empty. The "Context" section contains a detailed explanation of why the dataset was created, mentioning the requirement for a clean dataset of books and the challenges of finding one. The "Data Explorer" section shows a file named "books.csv" (1.56 MB) with 12 columns. The "Summary" section indicates there is 1 file and 12 columns. The "About this file" section provides a detailed description of each column: bookID (A unique identification number for each book), title (The name under which the book was published), authors (Names of the authors of the book. Multiple authors are delimited with ~), average_rating (The average rating of the book received in total), and isbn (Another unique number to identify the book, the International Standard Book Number). Below this, there are five small histograms showing the distribution of values for each column.

Column	Description	Count
bookID	A unique identification number for each book.	10352 unique values
title	The name under which the book was published.	6643 unique values
authors	Names of the authors of the book. Multiple authors are delimited with ~.	11126 unique values
average_rating	The average rating of the book received in total.	45.6k
isbn	Another unique number to identify the book, the International Standard Book Number.	1

Data Acquisition



As of December 8th 2020, Goodreads no longer issues new developer keys for our public developer API and plans to retire the current version of these tools. You can find more information [here](#). X

API

INTRODUCTION

The Goodreads API allows developers access to Goodreads data in order to help websites or applications that deal with books be more personalized, social, and engaging. The API can be used in many ways, including:

- **Goodreads Connect:** Let members connect to their Goodreads accounts, and you'll have full access to the books in their shelves, their ratings, their reviews, and their friends – the social reading graph. Use this to personalize an ecommerce store, power recommendations, show a

[api terms](#)
[getting started](#)
[developer forums](#)
[contact us](#)

Data Acquisition

Listopia

FEATURED LISTS



Popular Kindle Notes & Highlights on Goodreads

90 books — 2 voters

New horror for your Halloween reading

21 books — 3 voters



E.J. Koh's Books to Celebrate Asian American Fiction, Non-Fiction, Memoir, Graphic Novel, and Poetry

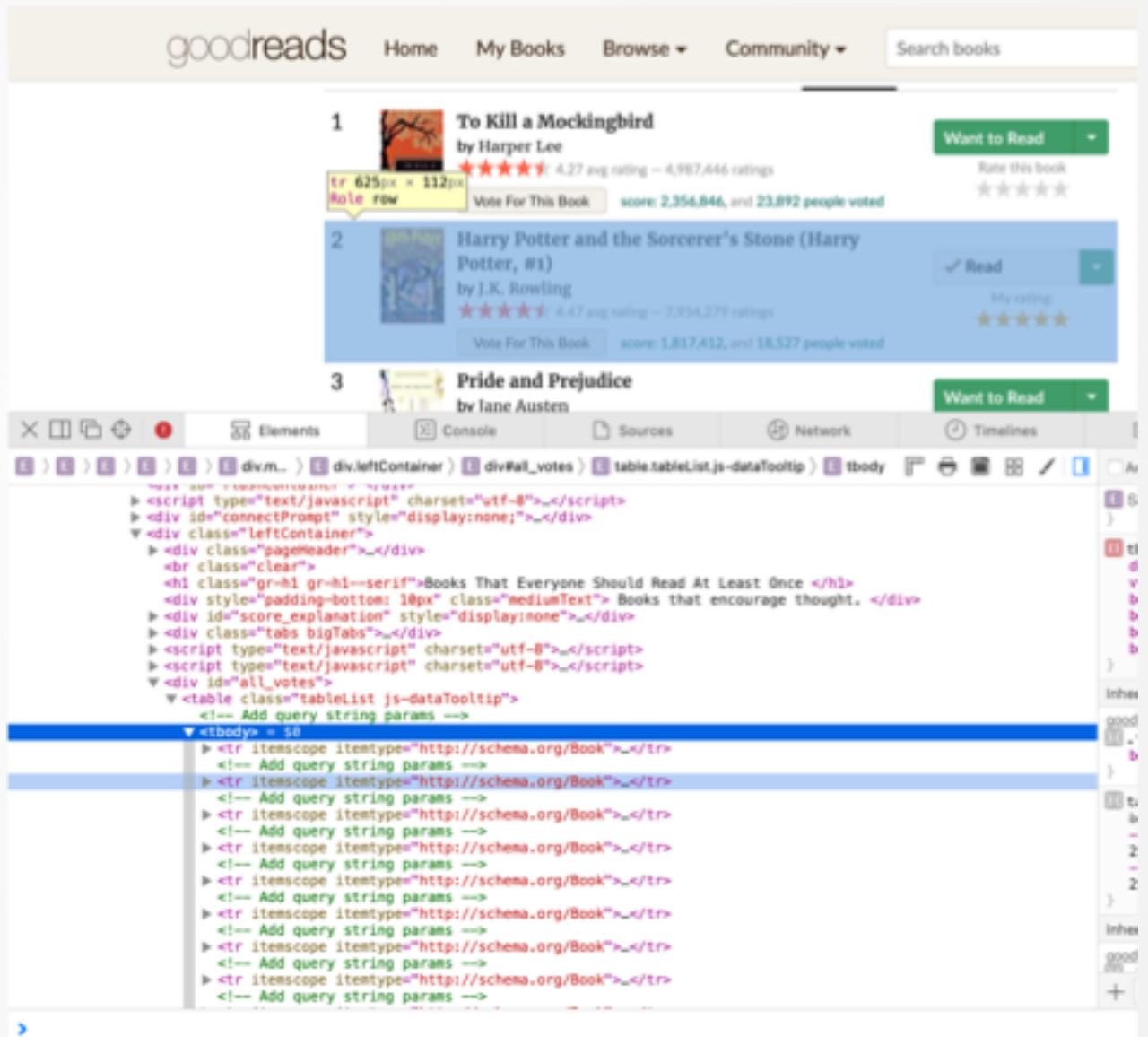
28 books — 2 voters

Food Books for Readers to Devour (nonfiction)

29 books — 7 voters

Data Acquisition

9



BeautifulSoap!

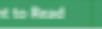
Data Acquisition

```
def get_book_list(url):
    soup = BeautifulSoup(requests.get(url).text, "lxml")
    book_list = []
    pages_count = int(soup.find("div", class_="pagination").find_all("a")[-2].text)
    for p in range(1, pages_count+1):
        soup = BeautifulSoup(requests.get("{}?page={}".format(url, p)).text, "lxml")
        book_list.extend([x["href"] for x in soup.find_all("a", class_="bookTitle", href=True)])
    return book_list
```

Listopia

Books That Everyone Should Read At Least Once

Books that encourage thought.

		All Votes	Add Books To This List
1	 To Kill a Mockingbird by Harper Lee  4.27 avg rating — 4,967,646 ratings Vote For This Book score: 2,354,940, and 23,892 people voted	Want to Read 	Rate this book 
2	 Harry Potter and the Sorcerer's Stone (Harry Potter, #1) by J.K. Rowling  4.47 avg rating — 7,954,229 ratings Vote For This Book score: 1,817,312, and 18,527 people voted	 Read 	My rating 
3	 Pride and Prejudice by Jane Austen  4.27 avg rating — 3,376,013 ratings Vote For This Book score: 1,450,066, and 14,850 people voted	Want to Read 	Rate this book 
4	 The Diary of a Young Girl by Anne Frank  4.14 avg rating — 3,072,420 ratings Vote For This Book score: 1,377,540, and 14,110 people voted	 Want to Read 	Rate this book 
5	 Animal Farm by George Orwell  3.97 avg rating — 3,078,604 ratings Vote For This Book score: 1,031,244, and 10,687 people voted	 Want to Read 	Rate this book 
6	 1984 by George Orwell  4.19 avg rating — 3,574,707 ratings Vote For This Book score: 1,030,655, and 10,539 people voted	Want to Read 	Rate this book 

← previous 1 2 3 4 5 6 7 8 9 ... 99 100 next →

Data Acquisition

```
def get_book_list(url):
    soup = BeautifulSoup(requests.get(url).text, "lxml")
    book_list = []
    pages_count = int(soup.find("div", class_="pagination").find_all("a")[-2].text)
    for p in range(1, page_count+1):
        soup = BeautifulSoup(requests.get("{}?page={}".format(url, p)).text, "lxml")
        book_list.extend([x["href"] for x in soup.find_all("a", class_="bookTitle", href=True)])
    return book_list
```

Data Acquisition

```
driver = webdriver.Safari()
dic = {"Title": [], "Author": [], "Description": [], "Publisher": [], "Year Published": [], "Genres": [], "Page Cou
for book in all_books[40000:50000]:
    driver.get("https://www.goodreads.com"+book)
    soup = BeautifulSoup(driver.page_source, "lxml")
    try:
        dic["Title"].append(soup.find("h1", id="bookTitle").text.strip())
    except:
        dic["Title"].append(None)

    try:
        dic["Author"].append([x.find("span", itemprop="name").text for x in soup.find_all("a", class_="authorName")])
    except:
        dic["Author"].append(None)

    try:
        dic["Description"].append(soup.find("div", id="descriptionContainer").text.strip())
    except:
        dic["Description"].append(None)

    try:
        publishing_info = [x.strip() for x in soup.find("div", id="details").find_all(class_="row")[1].text.split("\n")]
    except:
        pass

    try:
        dic["Publisher"].append(find_publisher(publishing_info))
    ----
```

Scope

13

Collected Features:

- Title
- Author
- Description
- Publisher
- Year published
- Genres
- Page count

```
len(set([author for authors in df["Author"] for author in authors]))
```

executed in 20ms, finished 19:37:09 2021-10-20

22425

```
genres = set()
for l in df["Genres"].values:
    genres = genres.union(set(l))
```

```
len(genres)
```

executed in 235ms, finished 10:38:35 2021-10-18

970

```
dict(genres.sum())
```

executed in 60ms, finished 18:05:22 2021-10-20

```
'Poverty': 24,
'Psychology': 1025,
'Mental Illness': 226,
'Vegan': 24,
'Adventure': 2726,
'Pre K': 4,
'Music Biography': 8,
'Wine': 3,
'Ornithology': 4,
'Eastern Africa': 44,
'Royal Air Force': 1,
'Helicopters': 1,
'Cats': 42,
'Beauty and The Beast': 23,
'Political Science': 172,
'Medievalism': 13,
'Yuri': 2,
'Chinese Literature': 16,
'Liberia': 3,
'Romanian Literature': 20,
```

Scope

```
def text_preprocessed(descriptions):

    description_processed = descriptions.apply(lambda x: x[376:-8])

    ### Remove anything that is not an alphabetical character, or a space.
    description_processed = description_processed.replace('[^a-zA-Z ]', " ", regex=True)

    ### Case unification
    description_processed = description_processed.apply(lambda x: x.lower())

    ### Stopword removal
    stopword_list = stopwords.words("english")
    description_processed = description_processed.apply(lambda x: [word for word in x.split() if word not in stopword_list])

    ### Bigrams
    phrase_model = Phrases(list(description_processed), min_count=1, threshold=1, connector_words=ENGLISH_CONNECTOR_WORDS)
    df["Description_processed"] = [" ".join(phrase_model[sentence]) for sentence in df["Description_processed"]]
    description_processed = [phrase_model[sentence] for sentence in description_processed]

    ### Stemming & Lemmatization
    description_processed = [" ".join([SnowballStemmer("english").stem(WordNetLemmatizer().lemmatize(word)) for word in sentence]) for sentence in description_processed]

    return description_processed

df["Description_processed"] = text_preprocessed(df["Description"])
```

Experiment #1: TF-IDF and Cosine Similarity

```
tf = TfidfVectorizer(analyzer="word", ngram_range=(1,3), min_df=0, stop_words='english')
tfidf_matrix = tf.fit_transform(df["Description_processed"])
cos_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
cos_sim

def get_recommendations(book, cos_sim=cos_sim):
    idx = all_books[all_books == book].index[0] #index of the book at question
    scores = pd.Series(cos_sim[idx]).sort_values(ascending=False)
    #The highest is going to be the book itself, so we exclude it
    top_10_idx = scores.iloc[1:11].index
    return scores[1:11], all_books.loc[top_10_idx]
```

Experiment #1: TF-IDF and Cosine Similarity

```
scores, recommendations = get_recommendations('Harry Potter and the Deathly Hallows')
```

executed in 17ms, finished 10:52:15 2021-10-18

```
recommendations
```

executed in 4ms, finished 10:52:15 2021-10-18

```
23806          Stories Of The Prophets
20101          The Iron Lance
22243          Mr. Psychic
6703   The Man Who Ate His Boots: The Tragic History ...
6283    The Loneliest Vampire in NYC
22139          Forever
24850          Futureland
22715          'Til the World Ends
19822  Zero to One: Notes on Startups, or How to Buil...
7055            Possessed
Name: Title, dtype: object
```

```
scores
```

executed in 4ms, finished 09:22:45 2021-10-18

```
23806  0.020658
20101  0.016718
22243  0.016594
6703   0.016182
6283   0.013223
22139  0.012986
24850  0.012597
22715  0.012443
19822  0.012252
7055   0.012045
dtype: float64
```

Experiment #2: Sentence Transformers/BERT and Cosine Similarity

The screenshot shows a Cornell University logo and the text "We the Simons". The arXiv.org URL "arXiv.org > cs > arXiv:1908.10084" is at the top, along with a search bar and "Help | Advanced Search". The paper title is "Computer Science > Computation and Language". It was submitted on 27 Aug 2019. The abstract discusses the limitations of BERT for semantic similarity search and introduces Sentence-BERT (SBERT) as a modification that uses siamese and triplet network structures to derive semantically meaningful sentence embeddings for cosine-similarity comparison, significantly reducing computation time while maintaining accuracy.

Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

Nils Reimers, Iryna Gurevych

BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) has set a new state-of-the-art performance on sentence-pair regression tasks like semantic textual similarity (STS). However, it requires that both sentences are fed into the network, which causes a massive computational overhead: Finding the most similar pair in a collection of 10,000 sentences requires about 50 million inference computations (~65 hours) with BERT. The construction of BERT makes it unsuitable for semantic similarity search as well as for unsupervised tasks like clustering.

In this publication, we present Sentence-BERT (SBERT), a modification of the pretrained BERT network that use siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. This reduces the effort for finding the most similar pair from 65 hours with BERT / RoBERTa to about 5 seconds with SBERT, while maintaining the accuracy from BERT.

We evaluate SBERT and SRoBERTa on common STS tasks and transfer learning tasks, where it outperforms other state-of-the-art sentence embeddings methods.

Comments: Published at EMNLP 2019
Subjects: Computation and Language (cs.CL)
Cite as: arXiv:1908.10084 [cs.CL]
(or arXiv:1908.10084v1 [cs.CL] for this version)

Submission history
From: Nils Reimers [view email]

Bidirectional Encoder Representations: a transformer-based machine learning technique for natural language processing pre-training developed by Google.

Experiment #2: Sentence Transformers/BERT and Cosine Similarity

```
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')

#Our sentences we like to encode
sentences = list(df["Description_processed"])

#Sentences are encoded by calling model.encode()
embeddings = model.encode(sentences)

executed in 13m 37s, finished 13:07:30 2021-10-18

books_embeddings = list(zip(df["Title"], df["Description"], embeddings))
executed in 169ms, finished 13:07:31 2021-10-18

cos_sim_embeddings = cosine_similarity(embeddings,embeddings)
executed in 10.6s, finished 13:07:41 2021-10-18

scores, recommendations = get_recommendations('Harry Potter and the Deathly Hallows', cos_sim_embeddings)
executed in 12ms, finished 13:07:59 2021-10-18

scores
executed in 6ms, finished 13:08:00 2021-10-18

19670    0.788545
16027    0.770192
1896     0.763471
20293    0.763228
11736    0.763059
9089     0.762973
2401     0.760832
9913     0.758609
12702    0.758377
21936    0.755704
dtype: float32
```

Experiment #2: Sentence Transformers/BERT and Cosine Similarity

recommendations

executed in 6ms, finished 13:08:03 2021-10-18

19670	Waterbound
16027	Keturah and Lord Death
1896	The Ignorant Maestro: How Great Leaders Inspir...
20293	Emily of New Moon
11736	Headlong
9089	The End of the Matter
2401	Kidnapped Hearts
9913	The Most Interesting Person in the Room: A bri...
12702	The Greeks and the Irrational
21936	Corbenic
Name: Title, dtype: object	

Experiment #2: Sentence Transformers/BERT and Cosine Similarity

CMU Book Summary Dataset

The CMU Book Summary Dataset supports ongoing work described in:

David Bamman and Noah Smith (2013), "New Alignment Methods for Discriminative Book Summarization," [\[ArXiv\]](#)

[booksummaries.tar.gz](#) [17M]

This dataset contains plot summaries for 16,559 books extracted from Wikipedia, along with aligned metadata from Freebase, including book author, title, and genre.

All data is released under a [Creative Commons Attribution-ShareAlike License](#). For questions or comments, please contact David Bamman (dbamman@cs.cmu.edu).

Example

The following example illustrates the data and metadata available for Don DeLillo's *White Noise*.

Book metadata

Wikipedia ID	1166383
Freebase ID	/m/04cvx9
Book title	White Noise
Book author	Don DeLillo
Publication date	1985-01-21
Genres	Novel, Postmodernism, Speculative fiction, Fiction

Plot summary

Set at a bucolic Midwestern college known only as The-College-on-the-Hill, *White Noise* follows a year in the life of Jack Gladney, a professor who has made his name by pioneering the field of Hitler Studies (though he hasn't taken German language lessons until this year). He has been married five times to four women and has a brood of children and stepchildren (Heinrich, Denise, Steffie, Wilder) with his current wife, Babette. Jack and Babette are both extremely afraid of death; they frequently wonder which of them will be the first to die. The first part of *White Noise*, called

```
df2.shape
```

executed in 5ms, finished 13:41:17 2021-10-18

(16559, 8)

```
df2["Summary_processed"] = text_preprocessed(df2["Plot Summary"])
```

executed in 1m 17.7s, finished 13:31:48 2021-10-18

```
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')
```

#Our sentences we like to encode

```
sentences = list(df2["Summary_processed"])
```

#Sentences are encoded by calling model.encode()

```
embeddings = model.encode(sentences)
```

executed in 8m 29s, finished 13:41:04 2021-10-18

```
books_embeddings = list(zip(df2["Book title"], embeddings))
```

executed in 47ms, finished 13:41:26 2021-10-18

```
cos_sim_embeddings = cosine_similarity(embeddings, embeddings)
```

executed in 3.45s, finished 13:41:33 2021-10-18

```
scores, recommendations = get_recommendations("Harry Potter and the Philosopher's Stone", cos_sim_embeddings)
```

executed in 12ms, finished 13:42:47 2021-10-18

scores

executed in 5ms, finished 13:42:58 2021-10-18

```
480      0.799137  
948      0.755065  
12207    0.750974  
15186    0.749860  
445      0.747034  
716      0.737015  
724      0.734902  
7677     0.731284  
13090    0.717387  
4841     0.716605  
dtype: float32
```

recommendations

executed in 4ms, finished 13:43:01 2021-10-18

```
480      Harry Potter and the Chamber of Secrets  
948      Tik-Tok of Oz  
12207    House of Many Ways  
15186    Necroscope V: Deadspawn  
445      Harry Potter and the Order of the Phoenix  
716      Harry Potter and the Prisoner of Azkaban  
724      Harry Potter and the Goblet of Fire  
7677     Hidden Warrior  
13090    The Great Ghost Rescue  
4841     The Witching Hour  
Name: Book title, dtype: object
```

Experiment #3: Embedding with Clustering

```
clustering_model = DBSCAN(eps=3, min_samples=2).fit(embeddings)
cluster_assignment = clustering_model.labels_

pd.DataFrame(np.unique(cluster_assignment, return_counts=True)).T
```

Experiment #3: Embedding with Clustering

```
pd.DataFrame(np.unique(cluster_assignment, return_counts=True)).T
```

executed in 19ms, finished 14:54:23 2021-10-18

	0	1
0	-1	14110
1	0	252
2	1	1715
3	2	2
4	3	2
...
186	185	3
187	186	2
188	187	2
189	188	2
190	189	2

191 rows × 2 columns

Thank You

