Customized Book Search Engine Based on Keywords Extraction of Amazon Customer Reviews

Xinyi Ye
sylva@umich.edu
School of Information

Rui Ding
ruiding@umich.edu
School of Information

Chenyun Tao
cyuntao@umich.edu
School of Information

1 Introduction

Amazon is one of the world's largest E-commerce platform. The ability to find about any product on Amazon is the main reason consumers shop there. However, the fact that customer can find almost everything on Amazon does not mean it takes less effort for them to find what they are really looking for. It is often the case that users have some specific expectations on certain characteristics of the product, but those features are not shown in the product title. As a result, it is hard for search engines to retrieve those "customized" features, which costs users bunch of time to choose the product they really want.

For example, a mom may want to buy a toy for her daughter, and she wanted the toy to be wooden (to prevent potential hurt to kids), educational and durable and suitable for girl. Amazon search engine did not perform well on recommending products for such retrieval queries like "wooden durable educational toys for girls". Thus, we want to improve the online shopping user experience by extracting useful information from millions of amazon user reviews and elaborating product title by adding some feature tags summarized from those reviews.

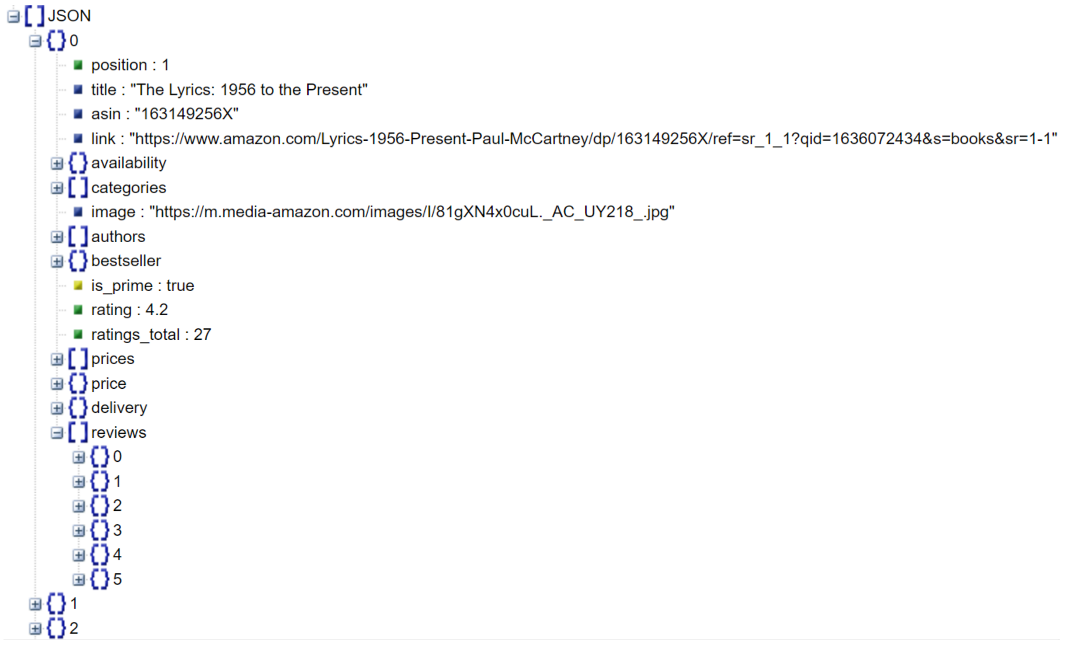
This project aims to extract useful information from Amazon reviews to elaborate the product title, such that users can easily search for products that meet their personalized requirements. The deliverable should be an improved product search engine that indicates feature summaries in product titles and at the same time provides better recommendations for customized search queries. Consider the scale of this project, we will narrow down our Amazon products to books. Besides the scale issue, we also found that titles of books on Amazon are extremely simple, only with single book names. Therefore, we find that adding review tags to books are more meaningful for personalized recommendation.

2 Data

For this project, our data comes from a professional Amazon scraping API called Rainforest API. This API provides clean, comprehensive and high-quality data especially for Amazon product data. We mainly used three APIs in this project. The first API extracted all the sub-categories under the main book categories. To be concrete in our case, it listed all 30 book genres under the "Books" main category. The Second API provided detailed information for concrete products, those information is pretty much like what users can see on the shopping website. We extracted 160 books from each subcategory. It contains everything but the reviews for that product. So finally, we used the API that could extract reviews of each product and append the reviews information to the product information mentioned in the second part. The final data are in JSON format, which contains product information of 4800 books. Later we may decide to further enlarge the dataset if necessary. One example data is shown in Figure 1 and the basic statistics for dataset is shown in table 1.

Statistics Name	Statistics Number
Number of products	4800
Number of categories	30
Number of samples from each category	160

Table 1: Basic Statistics of Dataset



```
[ ] JSON
  { } 0
    position : 1
    title : "The Lyrics: 1956 to the Present"
    asin : "163149256X"
    link : "https://www.amazon.com/Lyrics-1956-Present-Paul-McCartney/dp/163149256X/ref=sr_1_1?qid=1636072434&s=books&sr=1-1"
    { } availability
    [ ] categories
    image : "https://m.media-amazon.com/images/I/81gXN4x0cuL._AC_UY218_.jpg"
    [ ] authors
    { } bestseller
    is_prime : true
    rating : 4.2
    ratings_total : 27
    [ ] prices
    { } price
    { } delivery
    [ ] reviews
      { } 0
      { } 1
      { } 2
      { } 3
      { } 4
      { } 5
  { } 1
  { } 2
```

Figure 1: Sample Book Information in JSON Format

3 Related Work

One common keyword extraction method is based on TF-IDF. Li et al. [2007] gave the specification of keywords in Chinese news documents based on analyzing linguistic characteristics of news documents and then proposed a new keyword extraction method based on TF-IDF with multi-strategies. The approach selected candidate keywords of uni-gram, bi-gram and tri-grams, and then defines the features according to their morphological characters and context information. Moreover, the paper proposed several strategies to amend the incomplete words gotten from the word segmentation and found unknown potential keywords in news documents. Experimental results show that their proposed method can significantly outperform the baseline method.

Mihalcea and Tarau [2004] proposed a graph based keywords extraction method. The idea of TextRank is derived from the classical PageRank algorithm by Page et al. [1999]. It divides articles into several component units, and chooses important components to build graph model. The higher the number and the higher the number of votes that are cast for a vertex, the higher the importance of the vertex. In particular, the authors proposed two innovative unsupervised methods for keyword and sentence extraction, and show that the results obtained compare favorably with previously published results on established benchmark.

After the emergence of TextRank, researchers have done lots of work on improving it. A more advanced TextRank method proposed by Wen et al. [2016] is called Word2Vec Weighted TextRank. In this paper, the authors did a research on the keyword extraction method of news articles. They built a candidate keywords graph model based on the basic idea of TextRank, use Word2Vec to calculate the similarity between words as transition probability of nodes' weight, calculate the word score by iterative method and pick the top N of the candidate keywords as the final results. Experimental results show that the weighted TextRank algorithm with correlation of words can improve performance of keyword extraction generally.

Deep learning techniques, such as BERT by Devlin et al. [2018], can also be applied to keyword extraction. BERT has five keywords, which are pre-training, deep, bidirectional, transformers, and language understanding. The author proposed to use the Deep Bidirectional Transformers model initially. However, the author believes that bi-directional still cannot fully understand the semantics of the entire sentence. A better way is to use the contextual omnidirectional prediction. The core of this model is the attention mechanism. For a sentence, multiple focus points can be activated at the same time, instead of being limited to the front-to-back or the back-to-front sequential serial processing. The Transformer model after training, including its parameters, is the universal language representation model that the author expects.

Tang et al. [2019] utilized BERT model on keyword extraction of clinical data including clinical text classification and extracting information from 'free text' and so on. The challenges include dealing with noisy clinical notes which contain various abbreviations, possible typos, and unstructured sentences. The objective of their research is to investigate the attention-based deep learning models to classify the de-identified clinical progress notes extracted from a real-world EHR system. The attention-based deep learning models can be used to interpret the models and understand the critical words that drive the correct or incorrect classification of the clinical progress notes. The attention-based models in this research are capable of presenting the human interpretable text classification models. The results show that the fine-tuned BERT with the attention layer can achieve a high classification accuracy of 97.6%, which is higher than the baseline fine-tuned BERT classification model.

4 Methods

Our work includes two parts. The first one is keyword extraction, which extracts tags from the reviews. The second part is information retrieval, which ranks the documents given queries. Here we used TF-IDF, BERT and TextRank for keyword extraction. Then we used latent Dirichlet allocation (LDA) model to get a list of books that are in the same category with the query. Finally we used two different retrieval ranking methods, BERT and BM25, on titles as well as the tags to provide recommendations for users' search queries.

4.1 Keyword Extraction Methods

4.1.1 TF-IDF

For a document, we aim at extracting several most representative words for this document as its tags. We use *spacy* package in Python. We first tokenize the documents into different words. We first lemmatize these words, and remove all the stop words. Then we filter the words using their part of speech and only retain positive words. Then we use the equation 1 to calculate the score of each words in the document and select top k words with highest scores.

$$\mathrm{tfidf}(t, d, D) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \cdot \log \frac{N}{|\{d \in D : t \in d\}|} \tag{1}$$

4.1.2 BERT

For keyword extraction, we also try BERT model. We use *KeyBERT* package in Python. KeyBERT is a minimal and easy-to-use keyword extraction technique that leverages BERT embeddings to create keywords and keyphrases that are most similar to a document.

First, document embeddings are extracted with BERT to get a document-level representation. Then, word embeddings are extracted for N-gram words/phrases. Finally, we use cosine similarity to find the words/phrases that are the most similar to the document. The most similar words could then be identified as the words that best describe the entire document.

The overview of our model is listed as figure 2.

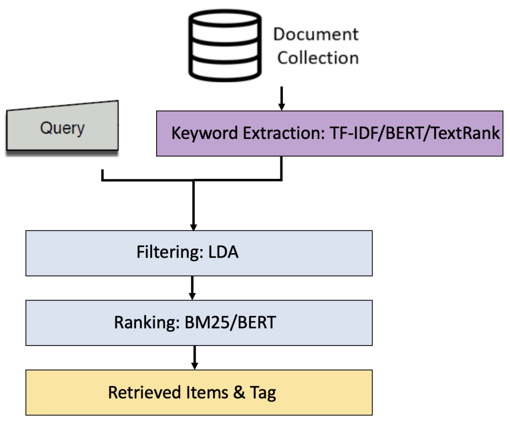


Figure 2: Overview of the model.

4.1.3 TextRank

TextRank algorithm is a graph-based ranking model for text processing which can be used in order to find keywords. The overflow of the algorithm is shown in figure 3.

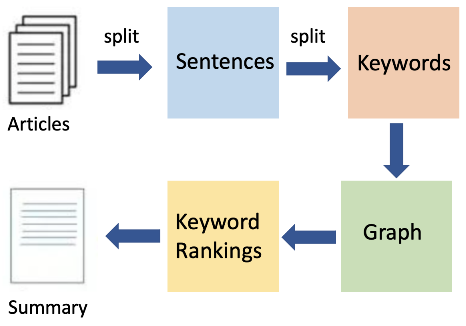


Figure 3: Structure of TextRank algorithm.

The algorithm used by TextRank for keyword extraction is as follows:

1. We split the given text T according to complete sentences, i.e., $T = [S_1, S_2, ...S_m]$
2. For each sentence $S_i \in T$, we perform word segmentation and part-of-speech tagging, and filter out stop words, and only retain words with specified parts of speech, such as nouns, verbs, adjectives, i.e., $S_i = [t_{i,1}, t_{i,2}, ..., t_{i,n}]$, where $t_{i,j}$ is the candidate keywords after retention.
3. We construct a candidate keyword graph $G = (V, E)$, where V is a node set, which is composed of candidate keywords generated in the previous step. Each node represents a candidate keyword. The graph uses a co-occurrence relationship to construct an edge between any two nodes. There is an edge between two nodes only if their corresponding keywords co-occurred in a window of length k, where k represents the window size, that is, at most k words can be co-occurred.
4. According to the equation 2, we iteratively propagate the weights of each node until they converge.

$$WS(V_i) = (1-d) + d \cdot \sum_{V_j \in \mathrm{In}(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j) \tag{2}$$

, where w_{ji} is the weight of the edge from node V_i to V_j in the graph. d is the damping coefficient, which represents the probability of pointing from a certain node to any other node in the graph, and the default value is 0.85. $\mathrm{In}(V_i)$ and $\mathrm{Out}(V_i)$ are the node set pointing to the node V_i and the node set starting from the node V_i.

5. We sort the node weights in reverse order to obtain the most important T words as candidate keywords.
6. We obtain the most important T words from the previous step and mark them in the original text. If adjacent phrases are formed, then they are combined into multi-word keywords.

4.2 Filtering

We use latent Dirichlet allocation (LDA) to generate a list of candidate books. Then we use BERT to search the most relevant documents corresponding to that query.

LDA is a generative probabilistic model that assumes each topic is a mixture over an underlying set of words, and each document is a mixture of over a set of topic probabilities. We can describe the generative process of LDA as, given the M number of documents, N number of words, and prior K number of topics, the model trains to output: ϕ_k, the distribution of words for each topic K, and θ_m, the distribution of topics for each document m.

In the Unigram text model, each word in the document is sampled independently from a separate multinomial distribution, which is the simplest case in the probabilistic text model. The probability of generating a word sequence is

$$p(\mathbf{w}) = \prod_{n=1}^{N} P(w_n) \tag{3}$$

Mixture of unigram language model introduced an implicit topic variable Z following discrete distribution. In this model, the text generation process is to choose a theme z first, and then generate N words from the conditions in the multinomial distribution $p(w|z)$ independently. The probability of generating a word sequence is

$$p(\mathbf{w}) = \sum_{z} p(z) \cdot \prod_{n=1}^{N} P(w_n|z) \tag{4}$$

As we can see from this process, the model assumes that each document can belong to only one topic, which makes the model very limited in actual text modeling, since a document often belongs to multiple topics.

PLSI text model is the most "advanced" text model before LDA. By introducing text variables, multiple topics can be combined together in a weighted form for a specific target text. The joint distribution of document and word is

$$p(d, w_n) = p(d) \cdot \sum_{z} p(w_n|z) \cdot P(z|d) \tag{5}$$

The PLSI text model does not generalize well, and it can be very weak for an "unknown" document in which most of the words do not appear in the training set document. Another serious problem is with the growth of parameters, the algorithm is easy to be overfitting for the training data.

Finally it comes our Latent Dirichlet Allocation method. The pseudo code of this algorithm is shown in 1.

Algorithm 1 Latent Dirichlet Allocation(α, β)

```
1: Sample a doc-topic distribution $\theta_m$, where $\theta_m \sim Dirichlet(\alpha)$
2: for every word $w_n$ in document $d_m$ do
3:     Sample a topic $z_n$, $z_n \sim Multinomial(\theta_m)$
4:     sample K topic-word distributions $\phi \sim Dirichlet(\beta)$
5:     Sample word $w_n$ according to $p(w_n|\phi_k)$
6: end for
```

4.3 Ranking Methods

4.3.1 BM25

For a query, our task is to give several most relevant documents. We use BM25 to calculate the similarity between each query and each document using the equation 6. For a specific query, we calculate its similarity with all the document and select k documents with the highest score.

$$S(Q,D) = \sum_{t \in Q \cap D} \ln\left(\frac{N - df(t) + 0.5}{df(t) + 0.5}\right) \cdot \frac{(k_1+1)\cdot c(t,D)}{k_1\left(1 - b + b\frac{|D|}{avdl}\right) + c(t,D)} \cdot \frac{(k_3+1)\cdot c(t,Q)}{k_3 + c(t,Q)} \quad (6)$$

4.3.2 BERT

Semantic search seeks to improve search accuracy by understanding the content of the search query. In contrast to traditional search engines which only find documents based on lexical matches, semantic search can also find synonyms.

Semantic search is to embed all entries in our corpus, whether they be sentences, paragraphs, or documents, into a vector space. At search time, the query is embedded into the same vector space and the closest embeddings from our corpus are found. These entries should have a high semantic overlap with the query.

BERT represents the two-way encoder representation of Transformer. Unlike other recent language representation models, BERT aims to pre-train deep bidirectional representations by jointly adjusting the context in all layers. Therefore, the pre-trained BERT can be fine-tuned through an additional output layer, which is suitable for the construction of state-of-the-art models for a wide range of tasks.

BERT proposes a new pre-training goal: masked language model (MLM), which randomly masks some tokens in the input of the model. The goal is to predict the original vocabulary id based only on the context of the masked word. Unlike the pre-training of the left-to-right language model, the MLM target allows the characterization to fuse the context of the left and right sides, thereby pre-training a deep two-way Transformer. The full structure of the model is shown on figure 4.

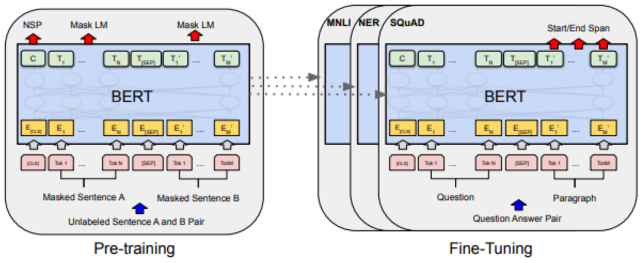


Figure 4: Structure of BERT [Devlin et al., 2018].

We use *sentence-transformers* package to do the BERT transformation. The package provides a list of pretrained models for word embedding. We use the pretrained model to do the word embedding for both corpus and the query. After getting the representing vectors for both corpus and query, we calculate the cosine similarity to find top k documents that are most relevant for this query.

4.4 Combination

Since our task includes two parts, keyword extraction and information retrieval. We try three methods for keyword extraction, i.e., TF-IDF, BERT and TextRank. We try two methods for information retrieval, i.e., BM25 and BERT. Therefore, there are six combinations in all. We try these six models separately, and record the results.

5 Evaluation and Results

As is discussed in Section 4, we have tried 6 combinations of different keyword extraction and information retrieval methods, and we have created 2 baselines to evaluate the results. One is just a random baseline that will just return 15-30 random documents for each query. Our second baseline is actually one of the 6 methods, which is the model with key word extraction method TF-IDF and IR method BM25.

We picked 30 queries, and annotated 120 results for each query retrieved by each model using the 5-point relevance scale rating, where 1 represents the most not-relevant, and 5 represents the most highly-relevant. Then we calculated NDCG at rank 10

$$\text{NDCG}_{10} = \frac{\text{DCG}_{10} = r_1 + \sum_{i=2}^{10} \frac{r_i}{\log_2(i)}}{\text{IDCG}_{10}}$$

for each query to evaluate the performance of the models. As it will be messy to show all the results here, we selected results from 20 of the queries to demonstrate in Table 5. We could see that the model with keyword extraction method TextRank and IR method BERT performs best in almost all the queries. The NDCG at rank 10 of our model fluctuates around 0.8 - 1, while the NDCG at rank 10 of the 2 baselines stays around 0.2 - 0.4, so our model is quite successful. The results will be discussed more detailedly in Section 6.

Table 2: Evaluation Results

Query	random	TF-IDF +BM25	TF-IDF +BERT	BERT +BM25	BERT +BERT	TextRank +BM25	TextRank +BERT
American dream	0.20	0.20	0.80	0.20	0.20	0.92	**0.97**
Suspense reasoning	0.20	0.20	0.57	0.20	0.33	0.42	**0.94**
Warm family	0.20	0.60	0.66	0.37	0.54	0.88	**0.89**
Artificial intelligence	0.21	0.20	0.60	0.20	0.50	0.24	**0.87**
Asian culture	0.20	0.20	0.58	0.26	0.71	0.75	**0.76**
Biography	0.20	0.31	0.35	0.36	0.20	0.45	**0.85**
Brainstorm	0.22	0.20	0.66	0.20	0.52	0.41	**0.81**
Business idea	0.20	0.59	0.69	0.57	0.55	0.42	**0.89**
Children education	0.20	0.42	0.72	0.25	**1.00**	0.61	**1.00**
Geographic	0.21	0.41	0.79	0.49	**1.00**	0.776	0.94
Historical	0.20	0.20	0.81	0.43	0.96	0.53	**1.00**
Information	0.20	0.44	0.14	0.57	0.45	0.57	**0.90**
Long story	0.20	0.20	0.66	0.20	0.40	0.77	**0.97**
Math textbook	0.20	0.20	0.60	0.36	0.65	0.34	0.69
Movie	0.20	0.20	0.35	0.20	0.24	0.71	**0.83**
Mystery story	0.20	0.54	0.69	0.63	0.70	**0.85**	0.83
Novel with vivid plot	0.20	0.20	0.78	0.20	0.20	0.90	**1.00**
Original	0.20	0.20	0.64	0.35	0.20	0.62	**0.96**
Romantic	0.21	0.30	0.81	0.20	**1.00**	0.40	0.99
Success story	0.20	0.35	0.72	0.45	**0.86**	0.68	0.85

To illustrate the results better, and to help us analyze the effects of different parts of our system on performance, we also plot barplots on NDCG of different models using the same IR method, i.e., group the models based on whether they use BM25 or BERT as the IR methods. The graphs are shown in Figure 5 and 6 respectively. From these 2 barplots, we could see that the models with TextRank as the key extraction method are the 2 best models, which indicate the large positive impact of TextRank on performance. In addition, models with IR method BERT seem to perform much better than models with IR method BM25 on average. Then we plot a barplot comparing the NDCG of the 2 best models with TextRank as the key extraction method, which is shown in Figure 7. From this plot, we could see that BERT does improve the performance as well.

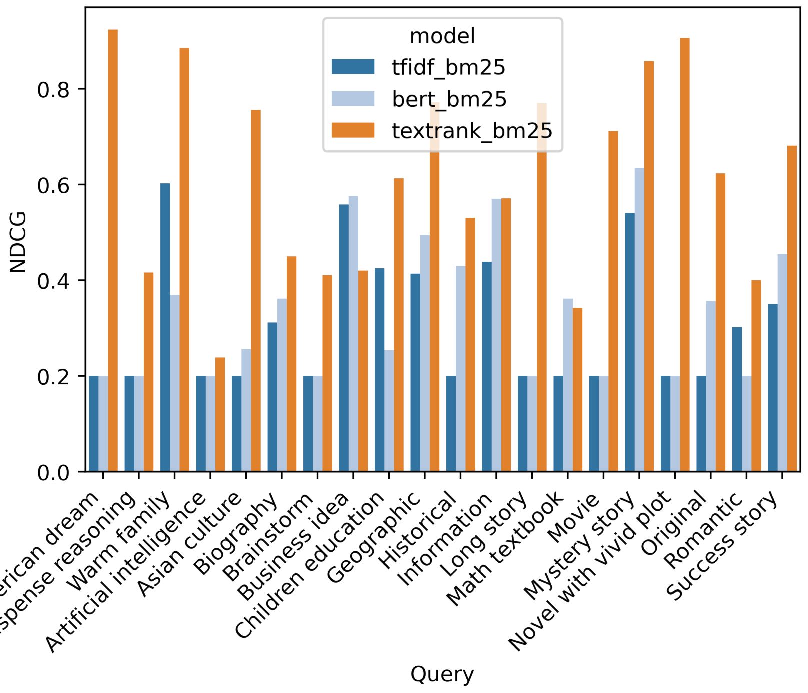


Figure 5: NDCG barplot on models with IR method BM25.

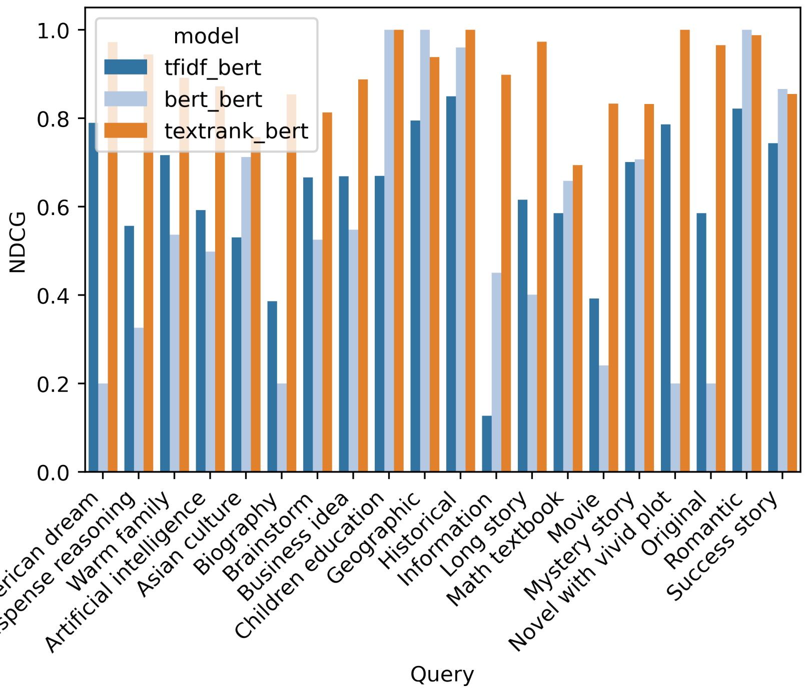


Figure 6: NDCG barplot on models with IR method BERT.

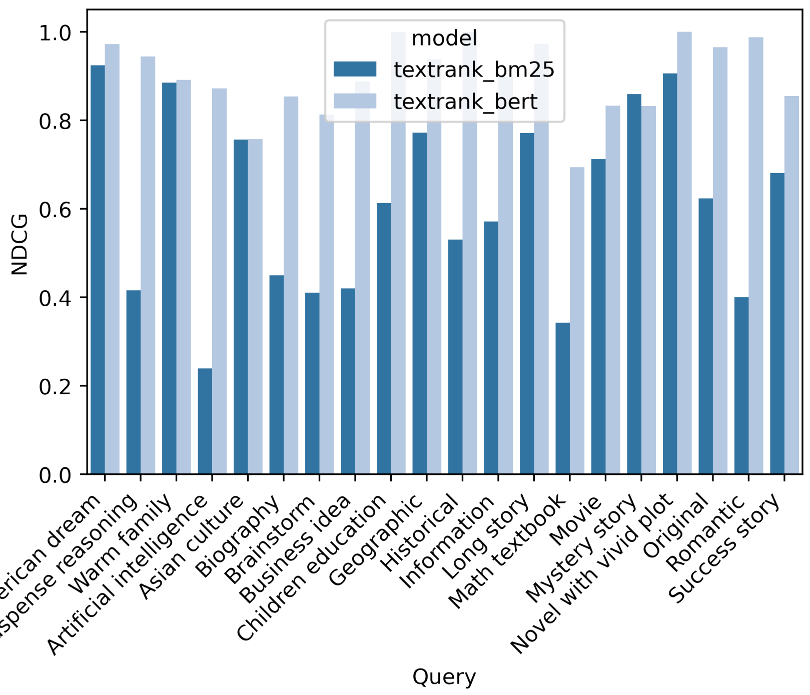


Figure 7: NDCG barplot on models with key extraction method TextRank.

6 Discussion

As is shown in Section 5, our model with keyword extraction method TextRank and IR method BERT performs best in almost all the queries. It performs much better than the 2 baselines. The NDCG at rank 10 of our model fluctuates around 0.8 - 1, while the NDCG at rank 10 of the 2 baselines stays around 0.2 - 0.5, so our model is quite successful. This large performance improvement is expected, and the results could satisfy the end-users.

It is unsurprising that the random baseline performs badly. There are 4800 books, and the probability to find a relevant book among the 15-30 randomly picked books is quite small, so the random baseline will hardly find the books meet the users' expectations.

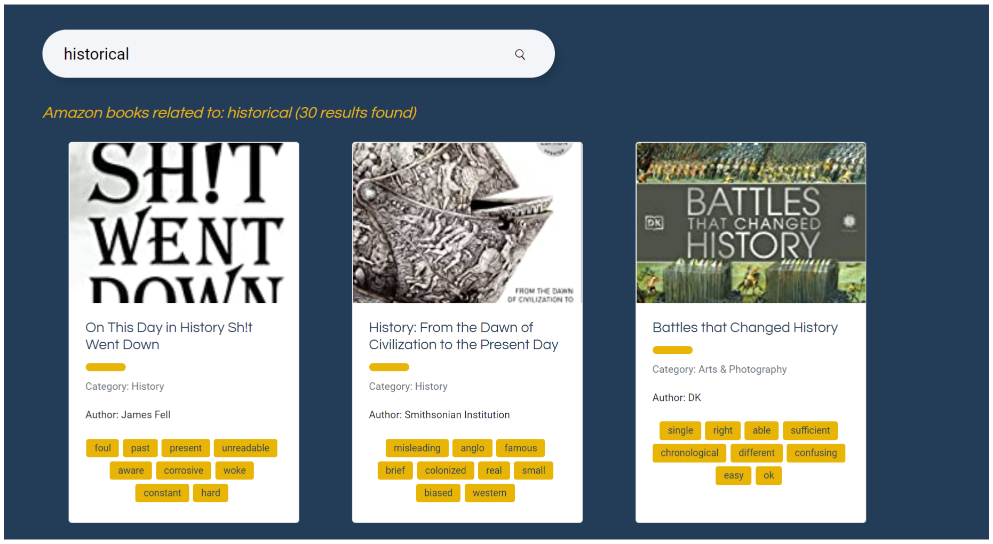
It is also unsurprising that the baseline with key word extraction method TF-IDF and IR method BM25 does not perform well. The queries that we composed are short and vague, and users might not mention the exact words in reviews frequently, so it could be difficult for the TF-IDF + BM25 baseline to extract related keywords, and retrieve the relevant books.

Finally for our model with keyword extraction method TextRank and IR method BERT, it is expected to perform well, and the results are actually even a bit better than expected. As TextRank performs really well in extracting relevant keywords, and BERT could conduct semantic search, which could be helpful to find documents related to vague and infrequent query terms, it is unsurprising that the model TextRank + BERT performs well. However, we are a bit worried that our bias in annotations might also affect the results. After searching for related work and proposing methods, we actually expect that the model with keyword extraction method TextRank and IR method BERT will have the best performance. Then since annotations are subjective, we might actually introduce bias into the relevance ratings, leading to bias in the results.

7 Conclusion

In this project, we developed a customized Amazon book search engine. The input is the user's query for books and the output is the books that are most relevant to the query. The model first extracted useful keywords from Amazon reviews with TextRank, then it used LDA to do the filtering and found the books in the same category as the query, finally it conducted the ranking of books with semantic search using BERT and showed the result in our own engine.

Model with TextRank to extract keywords and BM25 to construct the ranking performed well with NDCG at rank 10 fluctuating around 0.8 - 1. For keyword extraction, there are two reasons that TextRank performs well. The first one is that we used TextRank to construct a graph for each document and iterate the algorithm for each document until it converges, which means that we trained the model on our dataset. The second one is that TextRank can construct key phrases for each document instead of keywords. It is also due to the algorithm constructs a graph, the adjacent nodes can be connected to a phrase. We also tried BERT to exact keywords, and since we used a pretrained model, the result is not good as TextRank. For ranking, BERT outperforms BM25 since BERT can do the semantic search, while BM25 finds documents based on lexical matches.



Amazon books related to: historical (30 results found)

On This Day in History Sh!t Went Down
Category: History
Author: James Fell
foul past present unreadable aware corrosive woke constant hard

History: From the Dawn of Civilization to the Present Day
Category: History
Author: Smithsonian Institution
misleading anglo famous brief colonized real small biased western

Battles that Changed History
Category: Arts & Photography
Author: DK
single right able sufficient chronological different confusing easy ok

Figure 8: User Interface of Our Search Engine

We have deployed our search engine on here `https://tcy1999.pythonanywhere.com/`, you are welcomed to try our book search engine with some fancy queries. Notice that the search engine is deployed on free servers so it may not be stable, and it also needs 10 to 30 seconds to response the search query. One interesting future direction is to construct BERT network and train it on our own dataset to get the model for keyword extraction. We have put our codes on the github. The link is `https://github.com/yexinyinancy/SI650_project`.

8 Other Things We Tried

We also tried Deep Structured Semantic Models (DSSM) model [Huang et al., 2013]. It is not only used for text matching, but also for User-Item matching of the recommendation system. We used it for text matching. It is mainly divided into a presentation layer and a matching layer. The presentation layer can use fully connected, RNN, and Transformer networks to obtain the vector of query and doc. The matching layer generally uses cosine similarity to calculate query and 1 positive sample doc and N negative samples The similarity of doc. The full structure is shown in figure 9.

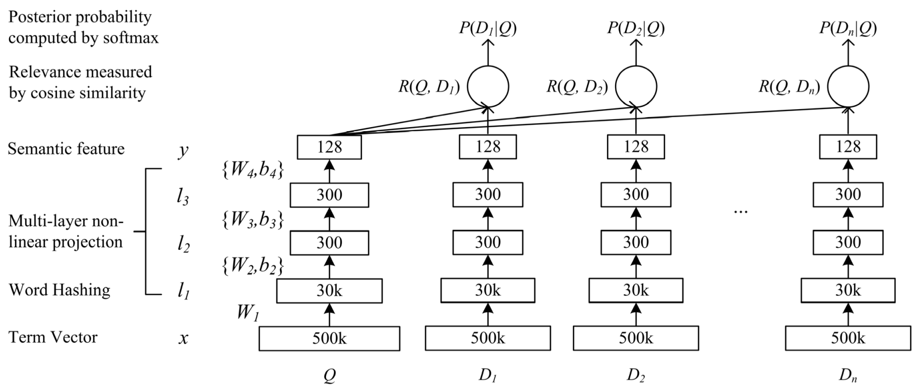


Figure 9: Structure of DSSM [Huang et al., 2013].

However, since the model does not have package, so we need to realize it by our own. Due to the time limit, we cannot successfully construct such neural network. We need to use tensorflow to construct it from the convolution and pooling layer. The whole structure is too complex for us to build it in several weeks.

9 What's Next?

Overall, the project is quite successful and meet our expectation. But there are still some points that can be further polished.

First, our dataset has some limitation. In the project, we only use the reviews of books, and do not consider other products. Besides, our dataset only contain 4800 products. Both the diversity and the amount of data are not enough. If we start the report over, we will first collect more data and data of products with wider diversity.

Second, from the results we can find that the best model is to use TextRank to extract keywords and to use BERT to do the do the ranking. For BERT used for ranking, we use a pre-trained model. If we can construct BERT model by our own and train it on our won dataset. Then we use such model to do the ranking, the result will be better. This is the idea we get from the result, and it is also the idea that we do not have time to do.

Third, we can also try some other deep learning models to do the keyword extraction as well as the ranking, such as DSSM model. Since deep learning models are so powerful, the results are anticipated.

Finally, our deployed search engine needs long time to respond to a given query! That's something related to front and back end interactions. Though it's enough for demo purpose, it needs to be polished when actually deployed in reality.

References

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.

J. Li, K. Zhang, et al. Keyword extraction based on tf/idf for chinese news document. *Wuhan University Journal of Natural Sciences*, 12(5):917–921, 2007.

R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

M. Tang, P. Gandhi, M. A. Kabir, C. Zou, J. Blakey, and X. Luo. Progress notes classification and keyword extraction using attention-based deep learning models with bert. *arXiv preprint arXiv:1910.05786*, 2019.

Y. Wen, H. Yuan, and P. Zhang. Research on keyword extraction based on word2vec weighted textrank. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 2109–2113. IEEE, 2016.