Enhanced Content-Based Similarity Detection for Book Recommendation

by

Peilin Sun

Partner Company:	Rakuten Kobo Inc.
Academic Supervisor:	Dr. Murat A. Erdogdu
Industrial Supervisor:	Chris Sjostrom

A report submitted in conformity with the requirements
for the degree of Master of Science in Applied Computing

Department of Computer Science
University of Toronto

Enhanced Content-Based Similarity Detection for Book Recommendation

Peilin Sun
Supervisor(s): Dr. Murat A. Erdogdu, Chris Sjostrom
Partner Company: Rakuten Kobo Inc.

Department of Computer Science, University of Toronto
December, 2020

Abstract

At Kobo, item recommendation is the main tool for users to discover e-books and audiobooks in the company database. Based on users' purchase history and reviews, books are recommended through a process known as collaborative filtering. However, this method does not give ideal results for books with little purchase history, thus, an advanced book recommender system is needed. This is the main focus of this project. We developed a recommender system based on Kobo's existing content-based similarity detection algorithm, which can also work on books with no purchase history.

With book descriptions as the training data, two approaches have been implemented, Latent Dirichlet Allocation (LDA) and doc2vec, to extract the embedding vectors of books. Recommendations were generated by comparing the similarity scores between all book pairs. Considering the possibilities of production, as well as low time complexity and high parallelization potential, the doc2vec model was chosen as the final model. By comparing the recommendation results with the original model and purchase-based similarities, the new approach worked significantly better than the original one, with almost doubled mean average precision scores. For further improvement, future work includes considering name entity recognition in text preprocessing and leveraging more book content. Finally, an A/B test was performed on production, with statistically significant result in terms of click conversion rate.

Acknowledgments

This work was supported by Mitacs through the Mitacs Accelerate program.

Contents

Chapter 1

Background Information

Founded in 2009, Kobo is a Canadian company that sells e-books, audiobooks, e-reader, and now it's part of Japan's e-commerce platform Rakuten. As an online bookstore, item and user recommendations are an important feature of the Kobo platform. As a major path to content discovery and personalized user experience, recommendations account for a significant percentage of Kobo's total sales. Currently, Kobo's item recommendations are driven by a process known as collaborative filtering (CF). In this method, a similarity score between a pair of books is calculated by taking the intersection of users who bought both books and dividing it by the union of all users who bought either book. The book pairs with higher ratios of common users are deemed more similar. Item recommendations are also used to generate user recommendations. The recommender system suggests items to users that are highly related to those that are already in user's library.

Recommendations generated through the collaborative filtering process works very well for the most part. However, this process has a major shortcoming. The recommender cannot generate recommendations for items with little or no purchase history, i.e. cold-start problem. Kobo has developed a simple content-based similarity detection that analyzes the books' texts. However, this does not provide meaningful recommendations in certain cases, especially for bestsellers and fiction books.

As an intern in the Big Data team, my goal was to develop new approaches that can address the shortcomings described above, and design a new content-based book recommendation system using NLP techniques.

Chapter 2

Related Works

The main challenge in computing which books are similar to each other is that books, when represented by numbers, can have very high dimensions. Traditional methods including bag-of-words and TF-IDF, which could be extremely high dimensional, and must be mapped to lower dimensions without losing important textual information. Strategies like Latent Semantic Indexing (LSI) project numeric representations derived through methods like bag-of-words to a lower-dimensional space by applying singular value decomposition [1]. Probabilistic Latent Semantic Indexing (pLSI) further advances this projection strategy by assuming that each word in a document is sampled from a mixture model that essentially represents a topic [1]. However, one drawback to pLSI is that the topics are derived from the training set and cannot adequately represent new information [1]. Addressing this drawback, Blei et. al developed Latent Dirichlet Allocation (LDA) which treats topics as hidden random variables, rather than parameters to be inferred from the training set [1]. Doing so allows LDA to represent documents as a composition of different topics even if the model had not seen that topic before [1].

Other than classic techniques mentioned above, researchers keep developing innovative methodologies in document embedding and deep recommendation systems. In 2014, Le et al. [5] put forward an approach that generates vector representation for documents, or even books, based on word2vec. In this framework, every paragraph is mapped to a unique vector in a Paragraph Matrix. Dai et al. [2] showed that Paragraph Vectors gave promising results in document similarity tasks and outperformed LDA's best result in analyzing arXiv papers. Moreover, Wu et al. [8] proposed a new approach called Word Mover's Embedding (WME) that utilized word2vec, Word Mover's Distance (WMD) [3] and D2KE [7], and gave great performance compared to other state-of-the-art approaches in textual similarity tasks.

Chapter 3

Research Goals and Outcomes

The main goal of this project is to improve Kobo's text-based similarity analysis using state-of-the-art approaches to provide better recommendations for all titles regardless of popularity and genre, as well as for all users who want recommendations that better reflect their purchases.

As mentioned in the background, to provide meaningful book recommendations, two models were designed for book embeddings extraction: LDA and doc2vec. Once the books were reduced to low-dimensional vectors, a distance measurement was used to calculate which books were closest, and hence most similar, to each other. The closest books to an item became its recommendations. By comparing the result with users' actual purchase history, the new model worked significantly better than the original approach, with doubled mean average precision at K (MAP@K) scores.

Chapter 4

Methodology

The training data were the raw book descriptions shown on Kobo's product pages. After text preprocessing, we need to extract the document embeddings of each book. Two approaches had been implemented, Latent Dirichlet Allocation (LDA) and doc2vec. LDA is Kobo's original model and it generates the topics distributions of each book. In this project, we built a new LDA model from scratch due to the lack of legacy code. Doc2vec is a neural network model that extracts document embeddings based on the popular word embeddings model, word2vec. After we got the embedding vectors, recommendations were generated by comparing the similarity scores between all book pairs. Finally, we performed an A/B test on production, evaluating the performance in terms of click conversions.

4.1 Data

The source data were the raw descriptions of 2.5M books, which were all English ebooks in Canadian market. For balanced training data, we selected books by stratified random sampling with similar numbers of books in each top-level categories, and their sub-categories, including fiction, nonfiction, romance, kids, mystery/suspense, sci-fi, business/finance, and biography. After text preprocessing, only books with more than 10 tokens were included in the training set.

After filtering and preprocessing, 2.4M books were kept with 1.9M books in the training set and 440K books in the test set.

4.2 Preprocessing

All raw text of a book will be preprocessed by spaCy to a list of individual tokens in unicode. Text preprocessing methods include:

- Parse raw HTML
  Since the raw text stored in the database is in HTML form, we used an HTML parser to remove all tags and extract text.

- ASCII transliterations of unicode text
  Convert special unicode text to more readable ASCII characters, e.g. "\u2019" to " ' ", "\u2043" to "-".

- Replace all non-alphanumeric characters with space
  In this case, abbreviations like "A.M.", "A.S.A.P", and hyphenated words like "cost-free", "high-tech" would not be identified. However, without this step, the vocabulary list we generated from test data would be filled with very unusual words or words with random symbols and punctuations, which were not usable in topic modeling. This is an effective compromise in reducing the size of the dictionary.

- Tokenization

- Lemmatization
  An effective lemmatizer is essential in converting words into their basic forms, thus reducing the dictionary size and training a more robust topic model.

- Removing stopwords
  The stopwords in this model were from the default list in spaCy package, and Buckley & Salton Stopword List [1]. For LDA model, common English names were also excluded to reduce noise and better catch the main idea of the document. The common names list comes from CMU AI Repository [2].

Since all documents were independent, we implemented parallel computing with multiple cores to speed up the preprocessing. Processed documents were stored as lists of ordered tokens in mini-batch groups for the following training process.

4.3 Models

4.3.1 Latent Dirichlet Allocation

In 2003, Blei et al. proposed the Latent Dirichlet Allocation (LDA), a generative probabilistic model for discrete data like text corpora [1]. In this project, it was used to infer the topic distribution of books given book description data.

[1] https://www.lextek.com/manuals/onix/stopwords2.html
[2] https://www.cs.cmu.edu/Groups/AI/util/areas/nlp/corpora/names/0.html

For statisticians, a document can be seen as an array of words $\mathbf{w} = (w_1, w_2, \ldots, w_N)$, and each word was generated from a specific distribution. Suppose we have a vocabulary indexed by $\{1, \ldots, V\}$ including all words in the corpus, it assumes each document $\mathbf{w}$ consists of k different topics, and each topic z can be described by words in the vocabulary, with a different probability. Based on Bayesian statistics, LDA also assumes the number of words in each documents $N \sim \text{Poisson}(\xi)$, the probability of topics in each documents θ follows a Dirichlet distribution $Dir(\alpha)$, and the same with topic-word probability $\varphi \sim Dir(\beta)$. By Bayes rules, we aim to find the topic distribution of each document, and the word distribution of each topic based on the observation of the corpus D, i.e., the posterior distribution of θ, $\mathbf{z}$, and φ:

$$P(\theta, \mathbf{z}, \varphi | \alpha, \beta, \xi, D) = \frac{P(\theta, \mathbf{z}, \varphi, D | \alpha, \beta, \xi)}{P(D | \alpha, \beta, \xi)}$$

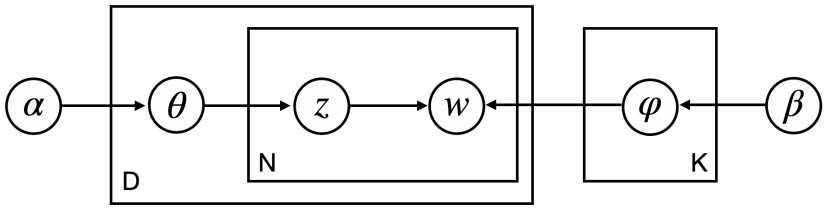

Figure 4.1: A graphical model of LDA

The graphical model of LDA is shown in Figure 4.1. However, it is unlikely to compute the above distribution analytically, so we can use some approximate inference algorithms like variational inference and Gibbs sampling.

The implementation of LDA was by python's machine learning package scikit-learn. Considering the large corpus size, it was impractical to load the full dataset into memory and train simultaneously. To learn incrementally from a mini-batch of instances, we utilized an online version of LDA, and the data were only loaded to the memory when being used. The raw input was ordered lists of tokens, which are first transformed into bag of words with hashed words and corresponding counts. The complete bag-of-words vector of the full corpus was used as the dictionary of the LDA model, in which all words that occur less than 30 times are discarded.

With topic size $K = 100$, and hyperparameters $\alpha = 1$, $\beta = 0.1$, the top words in some topics is shown in Table 4.1:

Topics	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
1	understand	explore	science	study	knowledge	research
2	write	short	collection	funny	laugh	humor
3	include	fiction	feature	issue	contain	note
4	search	escape	desperate	trap	disappear	release
5	offer	figure	subject	reveal	individual	engage
6	history	event	detail	modern	account	unique
7	look	question	answer	rise	walk	fill
8	tell	mean	believe	choose	reason	speak
9	start	share	realize	course	demand	involve
10	welcome	odd	crazy	hang	length	newly

Table 4.1: Top words in 10 topics of the LDA model

We also used this table in evaluation since the interpretability of topics is an important indicator of its performance.

After obtaining the topic distribution vector of books, a similarity score was calculated for all book pairs using Hellinger distance:

$$H(\mathbf{P}, \mathbf{Q}) = \frac{1}{\sqrt{2}} \|\sqrt{\mathbf{P}} - \sqrt{\mathbf{Q}}\|_2$$

where $\mathbf{P} = (p_1, \ldots, p_k)$ and $\mathbf{Q} = (q_1, \ldots, q_k)$ are the topic distribution vectors of the book pair. We converted this distance to a similarity score $S = 1 - H$, where $S = 1$ indicating the two books have the same topic distribution, and $S = 0$ means their topics are completely different. For each book, the scores with all other books were calculated and 96 books with the highest similarity score were chosen as content-based recommendations.

4.3.2 Doc2vec

Apart from topic modeling, a more advanced technique to convert documents into vector representation would be doc2vec, which is based on the word2vec approach. As a two-layer neural network that transforms individual words into vector representations, word2vec considers context information, detects similarities, and gives relatively good results [6]. As mentioned in related works, Paragraph Vectors gave a promising result in document similarity tasks and outperformed LDA's best result in analyzing arXiv papers.

There are two approaches in training using doc2vec: PV-DM (Distributed Memory) and PV-DBOW (Distributed Bag of Words). DM is a revised version of the

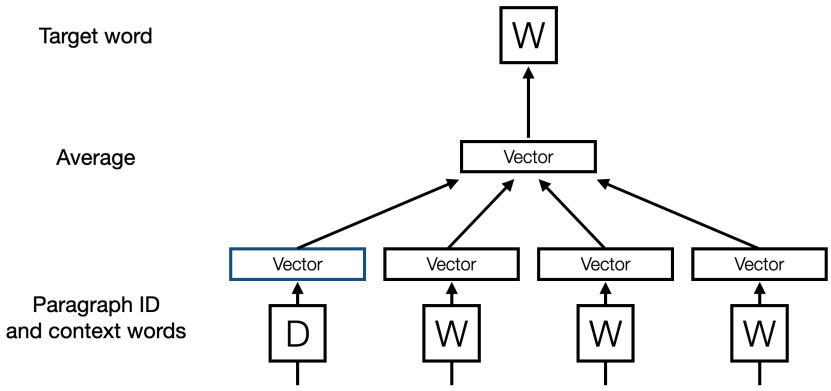

Figure 4.2: A graphical model of doc2vec PV-DM model

CBOW model in word2vec, as shown in Figure 4.2. In each training, a fixed number of words were selected from the sliding window, with the center word as the target word, and the others were context words. The word vectors of the context words and the paragraph vector of the document were the input layer. Then took the weighted average as the hidden layer, and predicted the target word with a softmax classifier. When started training, word vectors and paragraph vectors were randomly generated, and by training the same document multiple times, the paragraph vector could summarize the content of the document precisely. Instead of predicting the target word, we only input the paragraph vector to the DBOW model and predicted a random word in the sliding window.

As the vocabulary size is extremely large, it was computationally expensive to train all words with the softmax classifier. Mikilov et al. [6] suggested some optimization algorithms including hierarchical softmax and negative sampling. In this project, we trained both PV-DM and PV-DBOW models from scratch with Kobo's book description data and used negative sampling considering computational efficiency. For implementation, we used gensim package with Python, and similar to the LDA model, it was online training with mini-batches and 10 epochs. All tokens in the corpus were used as the dictionary of the doc2vec model, in which words that occur less than 30 times were discarded.

The model setting in this project is shown as follows:

- Negative sample = 5
- Number of epochs = 10

- Sub-sampling threshold = 10^{-5}
- Window size = 10
- Initial learning rate $\alpha = 0.025$
- Minimum learning rate $\alpha_{min} = 0.0001$

With document vectors, the similarity scores of all book pairs were calculated using cosine similarity, which is the cosine value of the angle between two vectors:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

where $\mathbf{A} = (a_1, \ldots, a_k)$ and $\mathbf{B} = (b_1, \ldots, b_k)$ are the document vectors. As $\cos(\theta) \in [-1, 1]$, this was normalized to $[0, 1]$.

4.4 Outcome

4.4.1 Evaluation

As an unsupervised learning process, a challenge in this project was to determine a vector representation is "good" for recommendation purposes. Some simple ideas include manually checking the interpretability of topics in the LDA model, or self-similarity in doc2vec. For LDA, perplexity, which is the exponent of the cross-entropy loss, reveals how the model was close to the actual topic distribution. This was used in tuning hyperparameters. After we found the best model settings of the two models, respectively, a challenge in the evaluation was to compare the performance between the LDA and doc2vec models.

To quantitatively measure the quality of the models, other related works tended to evaluate the performance of the subsequent supervised learning model. In this project, a novel idea would be building a bridge that connects document vectors to Kobo's recommender system. Since traditional collaborative filtering (CF) approach summarizes users' ratings, and purchase history, it provides good item-item recommendations that users may be interested in. Suppose purchase-based similarity was the "best" result, and the actual similarity, then we could compute the mean average precision at K (MAP@K) to compare the performance of content-based models. The MAP@K score was derived from the average precision @ K (AP@K), which measures the mean precision of relevant results among the top K documents. The formula of MAP@K is as follows:

$$MAP@K = \frac{1}{N}\sum_{i=1}^{N} AP@K_i$$

Table 4.2 shows the MAP@K scores, which quantitatively measure the quality of the models.

K	baseline	LDA	PV-DM	PV-DBOW
20	0.13776	0.20808	0.11829	0.23753*
50	0.12793	0.19664	0.11817	0.22307*
80	0.12731	0.19899	0.11898	0.22415*

Table 4.2: MAP@K scores of LDA and doc2vec models with purchase-based similarity. The baseline is Kobo's original model, and PV-DM and PV-DBOW are the doc2vec model we implemented.

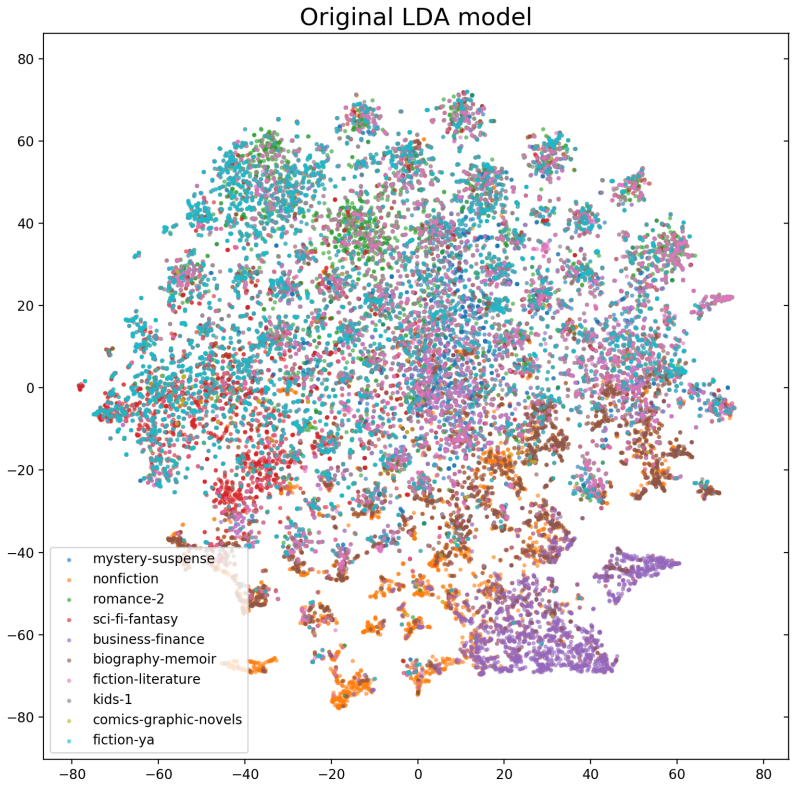

Figure 4.3: A visual representation of sample topic distribution vectors of the original LDA model

The baseline result came from Kobo's original LDA model. We notice that the PV-DBOW model of doc2vec works significantly better than the baseline, followed by the new LDA, and the PV-DM model did not provide ideal results. Although Mikilov et al. suggested that PV-DM alone usually worked well for most tasks [5], some researchers, like Lau et al. found that DBOW is superior to DM in some cases

[4]. This is the same in this project, partly because PV-DBOW typically works better with short documents and low-frequency words.

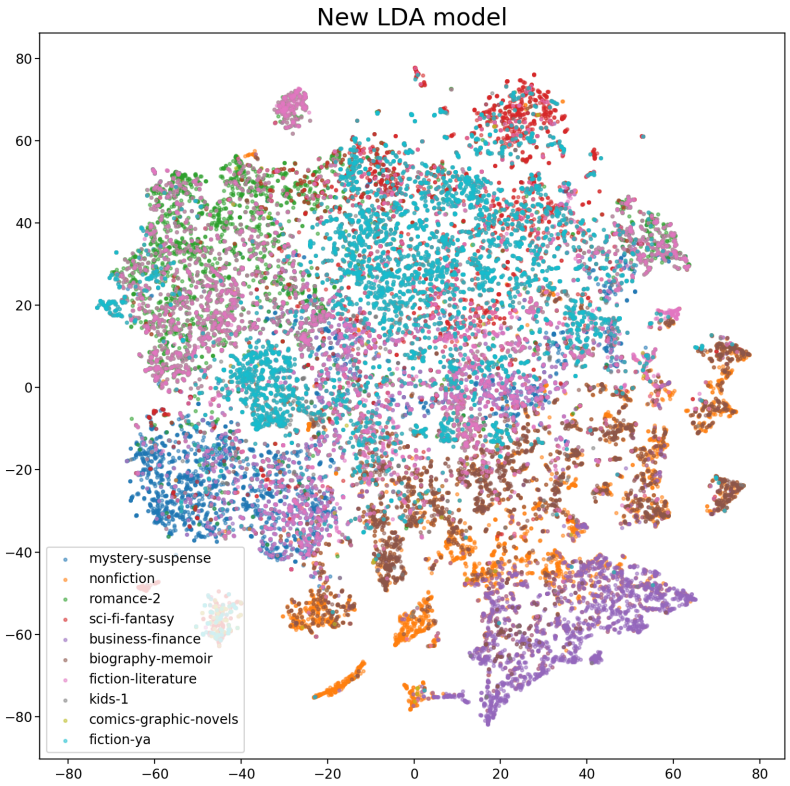

Figure 4.4: A visual representation of sample topic distribution vectors of the new LDA model

Another useful way in evaluation was to utilize book genres. As each book has pre-assigned labels with multiple levels of categories, we could check if books in the same category were similar according to content-based models. The 100-dimensional document vectors were reduced to a 2-dimensional space by t-SNE. Figure 4.3, 4.4, and 4.5 demonstrates a small portion of test data with books in the same category sharing the same color.

From Figure 4.3, it can be seen clearly that points with different categories are overlapped. For the new model as shown in 4.4, we notice clustered points with only a few colors, especially for nonfiction, business/finance, and biography books. Although the clusters of the LDA model look more separated from each other, in Figure 4.5, the single-colored pattern is more clear for PV-DBOW model. This could be visual evidence of the improvement based on the original model.

As the MAP@K scores gave a quantitative analysis of the model performance, it

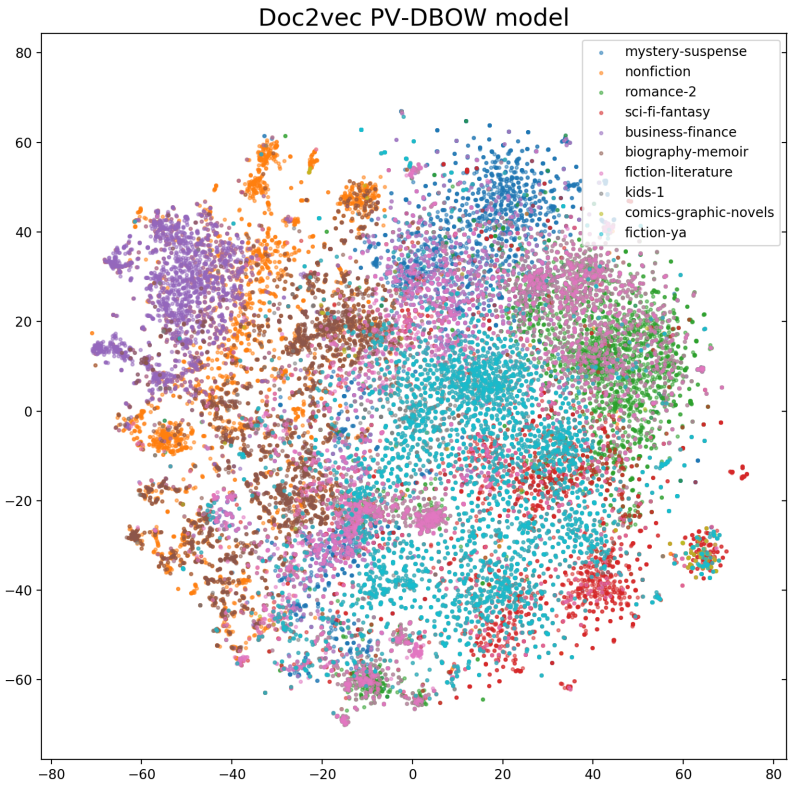

Figure 4.5: A visual representation of sample document vectors of PV-DBOW model

was given high priority in our evaluation. Then, considering t-SNE plots, as well as low time complexity and high parallelization potential, the PV-DBOW model was chosen as the final model, with a 76% increase in MAP@K score.

4.4.2 A/B Testing

Based on book traffic data (numbers of views and clicks) in September 2020, we randomly selected about 30,000 books and split them into two groups (A and B) with roughly equal traffic. The recommendations of group A books were only based on the original LDA model, while in group B, we used PV-DBOW model to find similar books. As the click conversion rate of selected books are about 2%, and the purchase conversion rate is 0.2%, we would need about 1.4 months for click conversion, and 15 months for purchase conversion if we would like to observe a 10% relative detectable change. Considering the timeline, we decided to test changes in click conversion rate.

Table 4.3 shows the result of the two-month A/B testing. Since group B's observed conversion rate (3.24%) was 19.27% higher than group A's conversion rate (2.72%). We could be 95% confident that this result was a consequence of the changes we made

	Visitors	**Conversions**	**Conversion Rate**
Group A	62898	1710	2.72%
Group B	72811	2361	3.24%*

Table 4.3: A/B testing result

and not a result of random chance. Therefore, we can conclude that the recommendations of our new content-based model appealed to more user traffics in terms of click conversion.

Chapter 5

Impact

As we chose the PV-DBOW model as our final model, the A/B testing result is promising. With this enhanced content-based algorithm, the recommendations of new arrivals are more accurate and appeal to customers with more clicks and potential purchases. As we replaced Kobo's original content-based model developed 7 years ago, our work is crucial to the growing need for our recommendations to reflect more of the books' content. This new recommender system will have a direct impact on millions of Kobo users, and Kobo will benefit from an improved customer experience and a potential increase in book sales.

From a technical perspective, in this project we implemented a neural network based approach of document embeddings, which provided insight into future improvements, since the core idea was not limited to book topics any more. We utilized users' purchase information as a gold standard to evaluate the model quantitatively. As this method applied to any similarity detection algorithms, it would also apply to future improvements if more advanced techniques are available.

Chapter 6

Conclusions and Future Research Plans

In this project, two NLP algorithms were applied for an enhanced book similarity detection: LDA and doc2vec. By comparing with purchase-based information, the PV-DBOW model of doc2vec was chosen as the final model in production, with a 76% increase in MAP@K scores, and 19.27% lift in click conversion rate. Since the doc2vec model had a simpler structure and leveraged word order information, it generated more accurate book recommendations with significantly higher click conversion rate.

With only book descriptions as the training data, the model might not capture the whole idea of the book, and the large vocabulary introduced noise into the model. A possible future improvement would be using name entity recognition (NER) in text preprocessing to replace people's names or other proper nouns that are unique in a book into uniform tags. Other approaches worth trying include leveraging more book content in the training process or using state-of-the-art NLP techniques.

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. *Latent dirichlet allocation*. 2003.

[2] Andrew M Dai, Christopher Olah, and Quoc V Le. *Document embedding with paragraph vectors*. 2015.

[3] Matt Kusner et al. *From word embeddings to document distances*. 2015.

[4] Jey Han Lau and Timothy Baldwin. *An empirical evaluation of doc2vec with practical insights into document embedding generation*. 2016.

[5] Quoc Le and Tomas Mikolov. *Distributed representations of sentences and documents*. 2014.

[6] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26 (2013), pp. 3111–3119.

[7] Lingfei Wu et al. *D2ke: From distance to kernel and embedding*. 2018.

[8] Lingfei Wu et al. *Word mover's embedding: From word2vec to document embedding*. 2018.