PERSONALIZED PRODUCT RECOMMENDATION ON SECOND HAND PLATFORMS

by

Ramazan Yarar

B.S., Industrial Engineering, Istanbul Technical University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering
Boğaziçi University

2022

ACKNOWLEDGEMENTS

First of all, I would like to express my deep and sincere gratitude to my research supervisor Assist. Prof. Mustafa Gökçe Baydoğan for giving me the opportunity to do research, providing invaluable guidance and continuously encouragement throughout the project

I would like to thank my friends Furkan Gürsoy and Oğuz Kaplan, who were willing and enthusiastic to assist with valuable discussions and thoughtful comments on this dissertation.A special thanks also to Kağan Fikri for encouraging me to start on this journey and motivating me throughout the process.

Last but not least, my family deserves endless gratitude. Their constant love and support keep me motivated and confident. My accomplishments and success are because they believed in me.

ABSTRACT

PERSONALIZED PRODUCT RECOMMENDATION ON SECOND HAND PLATFORMS

With the advent of online marketplaces which millions of people worldwide visit and make purchase every second, the shopping experience and competition between these platforms have been significantly changed and recommendation systems have become a more critical part of these platforms and gained popularity in the literature. One of these online marketplaces, in which the recommendation system plays a key role, is second hand platforms. In addition to general recommendation problems, these platforms have several problems which are specific to this domain such as compromising extremely unique item sets that makes the problem difficult with respect to other domains. In this study, we propose two staged model pipelines using state-of-the-art NLP techniques word2vec and paragraph2vec to address these problems with high quality personalized product recommendation in a scalable architecture. The model performance is evaluated on both offline experiments which are conducted on historical user clickstream dataset that is gathered from a popular second hand platform and A/B test on a production system. As a consequence of these experiments, the proposed model outperforms the baseline collaborative filtering-based models with respect to selected metrics, in addition, provides significant uplifts on several business metrics in the product system.

ÖZET

İKİNCİ EL PLATFORMLARDA KİŞİLEŞTİRİLMİŞ ÜRÜN ÖNERİMİ

Dünya genelinde milyonlarca insanın her saniye ziyaret ettiği ve alışveriş yaptığı online pazaryerlerinin ortaya çıkmasıyla birlikte, bu platformlar arasındaki rekabet ve alışveriş deneyimi önemli ölçüde değişmiş, öneri sistemleri bu platformların kritik bir parçası haline gelmiş ve literatürde popülerlik kazanmıştır. Tavsiye sisteminin kilit rol oynadığı bu online pazar yerlerinden biri de ikinci el platformlardır. Genel öneri sorunlarına ek olarak, bu platformların, sorunu diğer alanlara göre zorlaştıran, birbirine benzemeyen öğeler kümesinden oluşması gibi bu alana özgü çeşitli sorunları vardır. Bu çalışmada, ölçeklenebilir bir yapı içinde kişiselleştirilmiş ürün önerileriyle bu sorunları ele almak için son teknoloji doğal dil işleme tekniklerinden word2vec ve paragraf2vec kullanan iki aşamalı bir model öneriyoruz. Model performansı, hem popüler bir ikinci el platformundan toplanan geçmiş kullanıcı akışı veri kümesi üzerinde gerçekleştirilen çevrimdışı deneylerde, hem de canli sistem üzerinde yapılan A/B testinde değerlendirilir. Bu deneylerin sonucu olarak, önerilen model, seçilen metriklere göre temel işbirlikçi filtreleme tabanlı modellerden daha iyi performans gösterir ve ayrıca, ürün sistemindeki çeşitli iş metriklerinde önemli artış sağlar.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

AUC	Area under the ROC Curve
B2B	Business to Business
B2C	Business to Customer
C2C	Consumer to Consumer
CBOW	Continuous Bag of Words
CF	Collaborative Filtering
CTR	Click-Through Rate
DL	Deep Learning
LSTM	Long-Short Term Memory
MAE	Mean Absolute Error
MF	Matrix Factorization
NLP	Natural Language Programming
NN	Neural Network
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic
RSs	Recommendation Systems

1. INTRODUCTION

The last decades have seen the rapid development of an information society, and with it, the adoption of a rich ecosystem of online and offline shopping service platforms. With the advent of online marketplaces in which the buying and selling process take place electronically, people's shopping habits have changed, shifting to these platforms. They offer users a safe and easy shopping experience with many different options while obtaining marketing opportunities to sell additional products. Millions of people worldwide visit these platforms and make purchase every second. Now users are capable of meeting their demands easily from among the many options across numerous e-commerce platforms. It enhances tough competition between these platforms. They need to grasp users' preferences from various information sources and recommend the right product with personalized service to them in order to stay one step ahead of their competitors with respect to user satisfaction. As a consequence of this, recommendation system (RS) is a well-studied topic in the literature.

Recommendation systems (RSs) aim at providing the most appropriate product or service for users while they consider both their interests and the system's target. It utilizes alternative techniques to identify a user's preferences from their observed implicit feedback, such as rating, comments, or explicit feedback identified from a user's footprints in the system, such as transaction and session logs [1]. They then leverage these data to find the best recommendations according to desired target metrics or strategies. RSs are commonly used in many industries, especially in e-commerce, social media, and music/video streaming platforms.

Increasing technological developments make it easier to collect data about users' tastes and apply recommender systems on these platforms. Because RSs have a direct impact on companies' business objectives as well as user's behavior, they become a crucial part of these platforms. For instance, Youtube's published numbers show that 60% of users' clicks come from their home page recommendations; it was stated that 75% of the total watches of Netflix were attributed to their recommendation systems [2].

All of these incentives have streamlined the research in this area.

In the literature, the recommendation problem is addressed with different methods. The collaborative filtering (CF) approach is the most popular method, dominating studies in this field. It employ the notion that like-minded users have a similar product preference and utilize a user-product interaction matrix as seen in Figure 1, which each entry in this matrix represents interaction, which can rating, clicks, purchase count between selected user and product, to predict user preference.

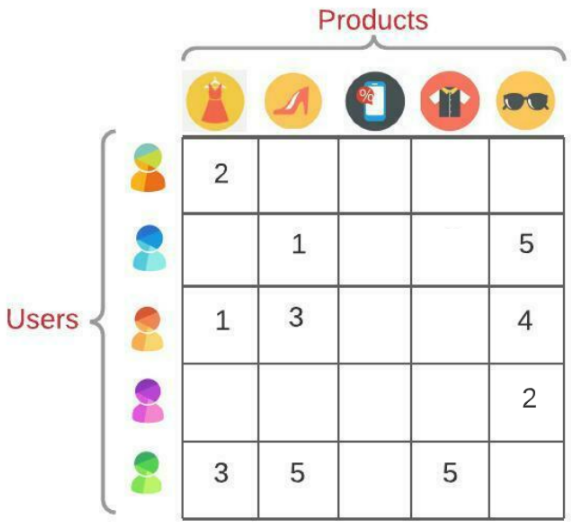

Figure 1.1. User - Product Interaction Matrix.

CF methods divided into two main types are called: memory-based and model-based CF. Memory-based CF builds an precomputed similarity database from user's interaction history on products and predicts top N similar products according to a kind of similarity metrics, such as cosine similarity and pearson correlation for the given target product. It is also divided into two main methods: user-based CF and item-based CF which are explained in detailed in the following sections. Besides this method, the model-based approaches use machine learning algorithms, which utilize both products' metadata information and users' diversified features, and is preferred to predict a user's interaction probability, rating, or desired metrics for a given product. It especially benefits from machine learning algorithms providing efficient and successful

data representation methods including matrix factorization that provides decomposing user-product matrix into low dimensional user and product matrices with their latent features, and many deep learning techniques, such as word embedding.

Even though CF techniques are very common and powerful for the recommendation problem, they bring out some common difficulties such as cold start, sparsity, popularity bias and computational cost or scalability. The cold start problem appears when users or items are new to the system. It makes it difficult to predict due to a lack of information about a new item [3]. If there is not enough data to represent users' preference on the items in the dataset, it causes a sparsity problem. The popularity bias stems from recommending the most interacted items frequently while giving less visibility to unpopular products [4] and distorts user expectation. Additionally, if the dataset consists of large numbers of users or products, especially with a few interactions between them, the models training and prediction phases require high numbers of resources, such as time and processing power, and therefore computational cost and scalability problems arise.

One of the platforms in which the RS plays a critical role is e-commerce. E-commerce is an exchange type, the buying and selling process, which takes place electronically [5]. Transactions may be conducted for goods and services, even information. The growing global digital transformation trend has shaped people's shopping habits and facilitated a world in which e-commerce has come into widespread usage. Furthermore, e-commerce's market share has been boosted in comparison with classical brick-and-mortar marketplaces. In addition to many big e-commerce companies such as Amazon, Alibaba, and eBay, new disruptive start-ups have emerged with new business types in this competitive ecosystem. These business types can be classified under three main categories that have been titled according to the buyer and seller side of the transaction: Business to Consumer (B2C), Business to Business (B2B), and Consumer to Consumer (C2C). This thesis focuses on Consumer to Consumer (C2C) online marketplaces that enable a consumer to directly sell products or services to another consumer without an established business. Taobao, eBay, Dolap, and Craigslist can be considered as examples of online marketplaces. They have become very popular as

a result of people's increasing environmental consciousness and technological trends. They provide sellers with an easy and secure selling environments without any physical or electronic setup. After signing up to those marketplaces, users are able to add their products to the platform with complementary information such as price, condition, size, age, and a photo. These products can be of numerous types: second-hand or hand-made, unique items. After that, buyers can bid, comment, or purchase products according to their preference and budget on the platform.

C2C marketplaces make money from sales commissions or listing advertisements. They should devote attention to increasing revenue while maintaining the other metrics representing the health of the business, such as active user count or click-through rate (CTR), which is ratio measuring number of clicks per number of total views in the page. To be able to raise their income, they need to benefit from their recommendation system. The main reason of it is that any given user's interaction with the platform starts with the output of the recommendation system that can be search results, homepage, or any similar item recommendation pages. So, any of these kinds of outputs directly affects users' decision and selling probability of the products. In this perspective, the recommendation system has a critical function for sustainable business, so when any improvement or exception happen on it, that can result in significant impact on the platform.

In C2C platforms, recommendation systems not only deal with e-commerce's inherent problems such as cold start, popularity bias and scalability, but also face additional problems distinctive to this area. The most prominent problem is that every product is a unique item without being in stock; they may be of different quality or age even if these items are the same product of the same brand. With a very large number of incoming new products, the proposed solutions face exponential growth in the system's overall candidate product pool with respect to classical e-commerce platforms and majority of products can get a few interactions or their visibility on listings is short-lived due to this volatile product pool, therefore, it results in an extremely sparse user-product interaction matrix which makes it difficult to obtain successful results with classical collaborative filtering approaches. Furthermore, most of the products have a

very short life cycle before being sold and get only a few interactions from the users. To illustrate, Figure 1 shows how many days it passed for a product to be purchased since it was added to the platforms in our dataset.

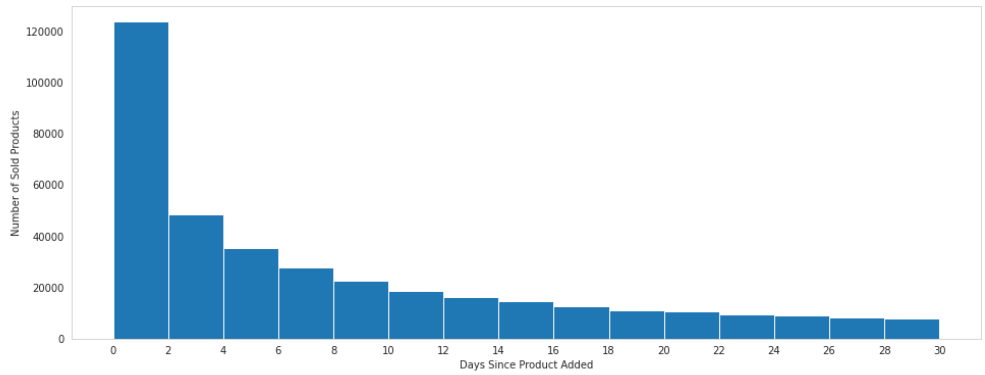

Figure 1.2. Histogram of Elapsed Days to be Purchased for a Product.

Beyond that, information quality about the product is another problem that needs to be tackled. As that is based on sellers' own claims, it can be poorly documented without structured way in a text and subjective especially with respect to product quality. For instance, there are example product listings from a second hand platform in Figure 1, most of the product properties are mentioned in a text and makes it difficult to utilize product attributes. In addition, important product attributes can be missing and not trustworthy. As a consequence of this, model based approaches using product attributes suffer from a lack of structured and trustworthy data problems and degrades model's performance. Furthermore, the proposed approach should take these problems into consideration; also it needs to be scalable because scalability is a major concern factor at real systems with millions of products and users.

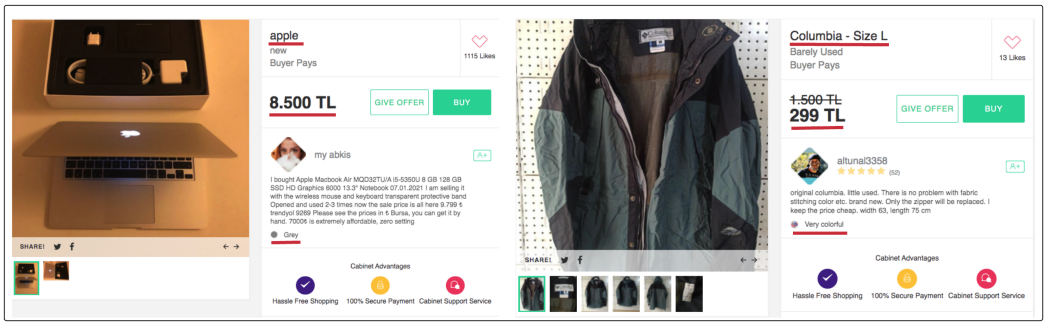

Figure 1.3. Example Screenshots of Poorly Documented Products from Dolap (Second Hand Online Marketplace).

The classical collaborative filtering and matrix factorization techniques show respectively poor performance [6], when it comes to address these problems and capture relationship between products from this extremely sparse user-product matrix. The prediction based word embedding models, which assume neighbouring/closer words in the sentence sequence are semantically related to each other and learn to predict neighbouring word for a given target word, produce state-of-the-art results in NLP tasks. These methods also have been successfully adopted to recommendation systems, especially on product representation and computing similarity between products. When the similarity estimation taken into consideration, these embedding models capture similarity between products in a superior way since their objective function is directly optimized to predict neighbouring products while traditional collaborative and matrix factorization models aim to reconstruct user - product matrix efficiently with various techniques, after that, obtain similarity value. This means that the latter approaches implicitly predict similarity between products from a reconstructed user - product matrix [7, 8].

This thesis proposes a multi staged method that provides personalized product recommendation based on users' clickstream data which is the electronic footprint left by the users during their journey in C2C second hand platforms. Our work has two phases: candidate generation and ranking candidates according to desired metrics. The candidate generation phase is necessary for identifying top N similar products for a given product. The word embedding techniques is leveraged to create embedding of

every product, which representing relationships among products. This is then used by the model to find the most similar top N items. In this way, a set of product candidates out of millions of products is obtained. In the second phase, a binary classification model has been trained to sort the candidate products according to their probability of being interacted by given user. This model uses product and user embeddings as input features, and thereby enhancing the our similarity-based first model regarding the user's preference is possible. The methodology of this thesis is experimented on a real world dataset that is supported by one of the important second hand platform in Turkey. Both are used on an offline dataset and compared the results with the traditional item-to-item CF method. Besides, the A/B test has been conducted and results have been collected in real world conditions. Our model considerably improves the key business metrics on live traffic. Both experiments demonstrate that the proposed model outperforms baseline classical approaches with efficient computation resources and training time.

The rest of this thesis is organized as follows: Section 2 gives detailed information about the structure of item-to-item CF and word embedding methods that have been used in the thesis. Then Section 3 summarizes the related literature on the recommendation system and its sub-topics. The proposed recommendation strategy is provided in Section 4. In Section 5, dataset, online and offline experiments and their results are explained. Section 6 summarizes the conclusions and future work.

2. BACKGROUND

This section presents the detailed information about item-to-item CF and word embedding methods to make the proposed approach more clear and understandable.

2.1. Collaborative Filtering

The collaborative filtering (CF) is a very common and powerful recommendation system method which frequently appears in the both literature and industry. CF exploits a set of user preferences which are obtained from different users' historical digital footprint or their explicit feedback to make predictions for the preferences of users who share common interests. CF methods are discussed in different categories and subcategories mentioned in the previous section. This thesis mainly focuses on item-to-item CF which is one of the memory-based CF approach and matrix factorization methods because many studies support the idea that item-to-item approach has shown superior performance to user-based approach, especially in extremely sparse datasets such as in e-commerce when it takes into account scalability and computation cost [9] [10].

2.1.1. Item-to-Item Collaborative Filtering

Collaborative filtering techniques aim to predict a utility value r, such as likelihood, number of interaction, a rating which is within a constant scale, or similarity value between candidate and target item. To achieve this goal, the item-to-item CF approach calculates similarity between item pairs based on an item pool including items which had interacted [9].

In item-to-item CF steps for this thesis as represented in Figure 2.1.1, it is initially built a $m \times n$ user-item matrix $\mathcal{A}$ which consists of a m dimension a set of users $u = \{u_1, u_2, \ldots, u_m\}$ and n dimension a set of items $i = \{i_1, i_2, \ldots, i_n\}$. Each element r_{ui} of the matrix shows user's preference on ith item that can be explicit feedback, such

as rating or implicit feedback which is obtained by the user's interaction history on jth item [9]. In this way, the $\mathcal{A}$ matrix's columns provide item vectors that represent users' preference on this item, after that, the similarity $\text{sim}(i, j)$ between item pairs i and j is computed based on their item vectors. The important point is that the similarity is calculated between only co-rated or co-interacted items by the same user instead of considering all item lists. Hence, it streamlines the computation process. There are two common methods for similarity measure: cosine-based similarity and correlation based similarity.

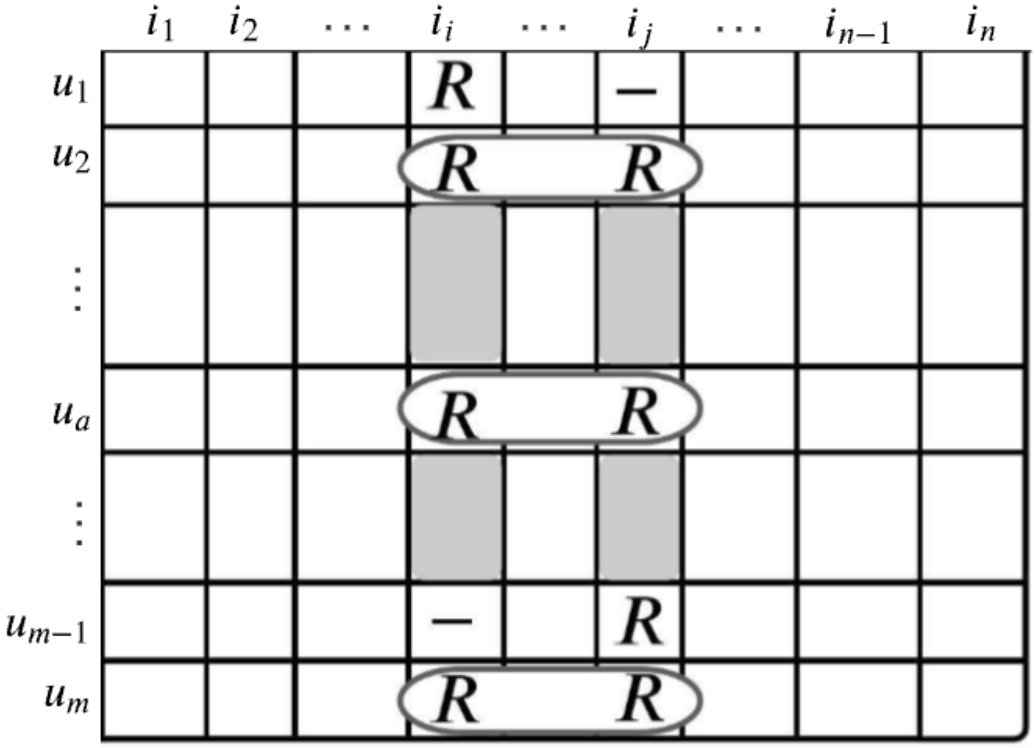

Figure 2.1. User - Item Matrix with Highlighted Co-Rated Items.

Cosine-based similarity exploits the notion that the cosine angle between item vectors $\vec{i}$ and $\vec{j}$ of dimensionality $|U|$ represent the similarity sim(i,j) of these vectors. The similarity $\text{sim}(i, j)$ is given by

$$\text{sim}(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\sum_{k=1}^{|U|} i_k \cdot j_k}{\sqrt{\sum_{k=1}^{|U|} i_k^2} \cdot \sqrt{\sum_{k=1}^{|U|} j_k^2}} \,. \tag{2.1}$$

In the correlation-based approach, Pearson-r correlation $\text{corr}_{i,j}$ is used as the similarity measurement. Likewise, it is computed between co-interacted items by the same user and is obtained by

$$\text{sim}(i,j) = \frac{\sum_{u \in U} \left(R_{u,i} - \bar{R}_i\right)\left(R_{u,j} - \bar{R}_j\right)}{\sqrt{\sum_{u \in U} \left(R_{u,i} - \bar{R}_i\right)^2} \sqrt{\sum_{u \in U} \left(R_{u,j} - \bar{R}_j\right)^2}} \tag{2.2}$$

where $R_{u,i}$ the utility value of the user u on the ith item has been defined while $\bar{R}_i$ average denotes the average of the utility values on the ith item.

The similarity between products is used to estimate the utility value of ith user on jth item with techniques such as the weighted average or regression or list K similar items to seed items in the prediction step [9].

Sparsity is one of the important problems affecting the model's performance as well as its computational cost. To overcome it, the singular value decomposition technique is used, a technique which maps the high dimensional vector space of the sparse $m \times n$ user-item matrix $\mathcal{A}$ to low dimensional dense vector spaces; $s \times n$ matrix which indicates a user-to-user relationship and $m \times s$ which represent an item-to-item relationship with latent features have been proposed in the literature [9, 11]. Reducing dimension size helps in both improving the model's results and scalability.

This process typically can be parameterized by user vectors x_u and item-factors vector y_i [9]. Taking the inner product of these vectors is used in the prediction of utility value

$$\hat{r}_{ui} = x_u^T y_i. \tag{2.3}$$

The objective is finding best vector representations and is generally modeled by the function

$$\min_{x_\star, y_\star} \sum_{r_{u,i} \text{ is known}} \left(r_{ui} - x_u^T y_i\right)^2 + \lambda \left(\|x_u\|^2 + \|y_i\|^2\right) \tag{2.4}$$

which minimizes loss with regularization parameter that penalizes overfitting. After that, the stochastic gradient descent method is widely applied for parameter estimation, especially in the explicit feedback datasets. Yu et al. extended Equation 2.4 and introduced new preference and confidence terms for implicit data [12]. The preference is represented by a binary p_{ui} variable, given by

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases} \tag{2.5}$$

where they assume that a user prefers an item i if he has interaction with the item ($r_{ui} > 0$) otherwise, he does not prefer ($r_{ui} = 0$). The utility value r_{ui} in implicit data can be purchased as frequency and click count or session duration, so they do not have a standard scale like rating. Thus, the paper proposed a confidence level parameter to identify a preference magnitude which is given

$$c_{ui} = 1 + \alpha r_{ui}. \tag{2.6}$$

With these two new parameters, the cost function is updated by

$$\min_{x_\star, y_\star} \sum_{u,i} c_{ui} \left(p_{ui} - x_u^T y_i\right)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right). \tag{2.7}$$

Unlike classical approaches, the authors utilize alternating least squares as an optimization algorithm and dramatically reduce computation time in a scale-able way.

2.1.2. Word Embedding

Word representation, which has been called embedding, has been one of the fundamental research areas in natural language processing (NLP). The main goal is representing words as a feature vector which learns their semantic relationship between each other and utilizing this vector together with mathematical methods in the solution of many NLP tasks.

Historically, vector space model, statistical language models and co-occurrence based methods have been well discussed in the word representation [13]. In classical approaches, they mostly suffer from the curse of dimensionality and scalability problems since the vector dimension generally depends on word count in the corpus. Bengio et al. introduced the neural probabilistic language model which is the first application of the neural networks in this domain to handle the curse of the dimensionality problem while enabling it to distribute representation [14]. In the paper, they agreed with the notion that the neighboring/closer words in the sentence sequence are semantically related to each other, and then modelled the problem as a next word prediction problem using a shallow feed forward neural network. Mikolov et al. made a significant contribution to this approach and introduced state-of-the-art distributed continuous word representation methods; continuous bag-of-words (CBOW) and skip-gram (SG). They dramatically reduced the model's training time and improved computational efficiency while outperforming other competitive models in various NLP tasks [13].

Assuming that traversing the words along the sentence with a fixed size sliding window; the word in the center of this window is called the target word as its left and right neighbors within the window size are called context words. The CBOW model is trained to predict the target word for given context words. Unlike CBOW, the skip-gram model aims to predict the context word based on the target word.

The feed forward neural network architecture of the prediction models can be briefly interpreted upon one elementary target-context pair before giving further details about CBOW and SG models, can be seen Figure 2.1.2.

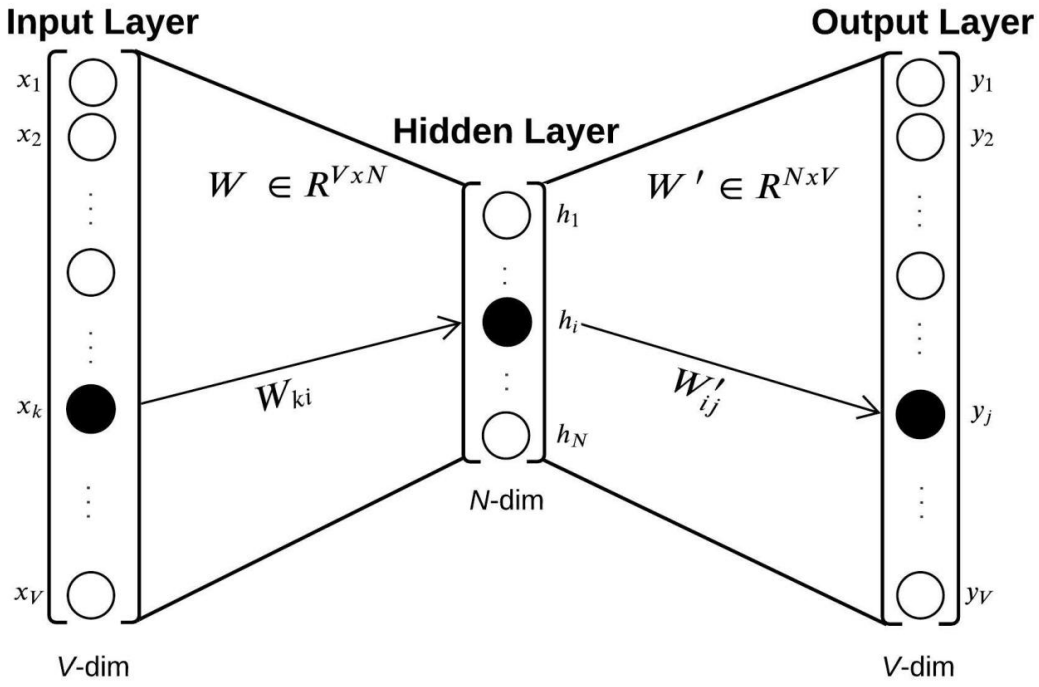

Figure 2.2. A Simple CBOW Model with One Context and One Target Word.

The NN architecture side of the study has a single hidden layer with N nodes which represent embedding size, in addition, input and output layers, their size is V which equal the vocabulary size. A given target word w_i is encoded with one-hot encoding as a binary, vector $x = \{x_1, x_2, \ldots, x_V\}$ that will be input to the model. Similarly, the context word w_j is encoded with a binary vector $y = \{y_1, y_2, \ldots, y_V\}$. The NN model aims to learn predicting correct context word w_j for given target word w_I in every iteration. So, the model has a $V \times N$ dimensional W weight matrix, which is called embedding matrix, between the input and embedding layer and its every row provides N dimensional representation vector v_w of given input word w_I. A linear activation function is used in the hidden layer's units, by the way, when the input vectors x and W matrix are multiplied; vector representation v_w is passed to the next layer and obtained by

$$h = W^T x = W^T_{(k,\cdot)} := v^T_{w_i}. \tag{2.8}$$

After that, $N \times V$ dimensional another weight matrix W' between hidden layer and

output layer is used to calculate a score u_j for each candidate word in the corpus, by

$$u_j = {v'_{w_j}}^T h \tag{2.9}$$

where v'_{w_j} denotes the j-th column in the W' matrix. softmax function is utilized to calculate the probability of the w_j's being a context word for a given target word w_I in the output layers' units. It can be formulated by

$$p\left(w_j \mid w_i\right) = y_j = \frac{\exp\left(u_j\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)} \tag{2.10}$$

where y_i represents out put of the i-th unit in the output layer. Here, Equation 2.10 can be reformulated by substituting Equation 2.9 and 2.8 and derived as

$$p\left(w_j \mid w_I\right) = \frac{\exp\left({v'_{w_j}}^T v_{w_I}\right)}{\sum_{j'=1}^{V} \exp\left({v'_{w_{j'}}}^T v_{w_I}\right)}. \tag{2.11}$$

Note that maximizing Equation 2.11 for a given input word w_I and actual output word w_O is the model objective. The loss function can be expressed as

$$\begin{aligned} E &= -\log p\left(w_O \mid w_{I,1}, \cdots, w_{I,C}\right) \\ &= -u_{j^*} + \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right) \\ &= -{v'_{w_O}}^T \cdot h + \log \sum_{j'=1}^{V} \exp\left({v'_{w_j}}^T \cdot h\right). \end{aligned} \tag{2.12}$$

Here computing probability for each word in the denominator of softmax function is computationally expensive for large corpus.

In skip-gram and CBOW, there are multiple context words for a given target word therefore a feed forward neural network architecture in a basic one-word context model mentioned above is needed to extend for multiple context word models.

The CBOW model in the proposed approach has multiple context word vectors in the input layer because the model predicts the target word which is in the middle of given context words, which can be seen in Figure 2.1.2.

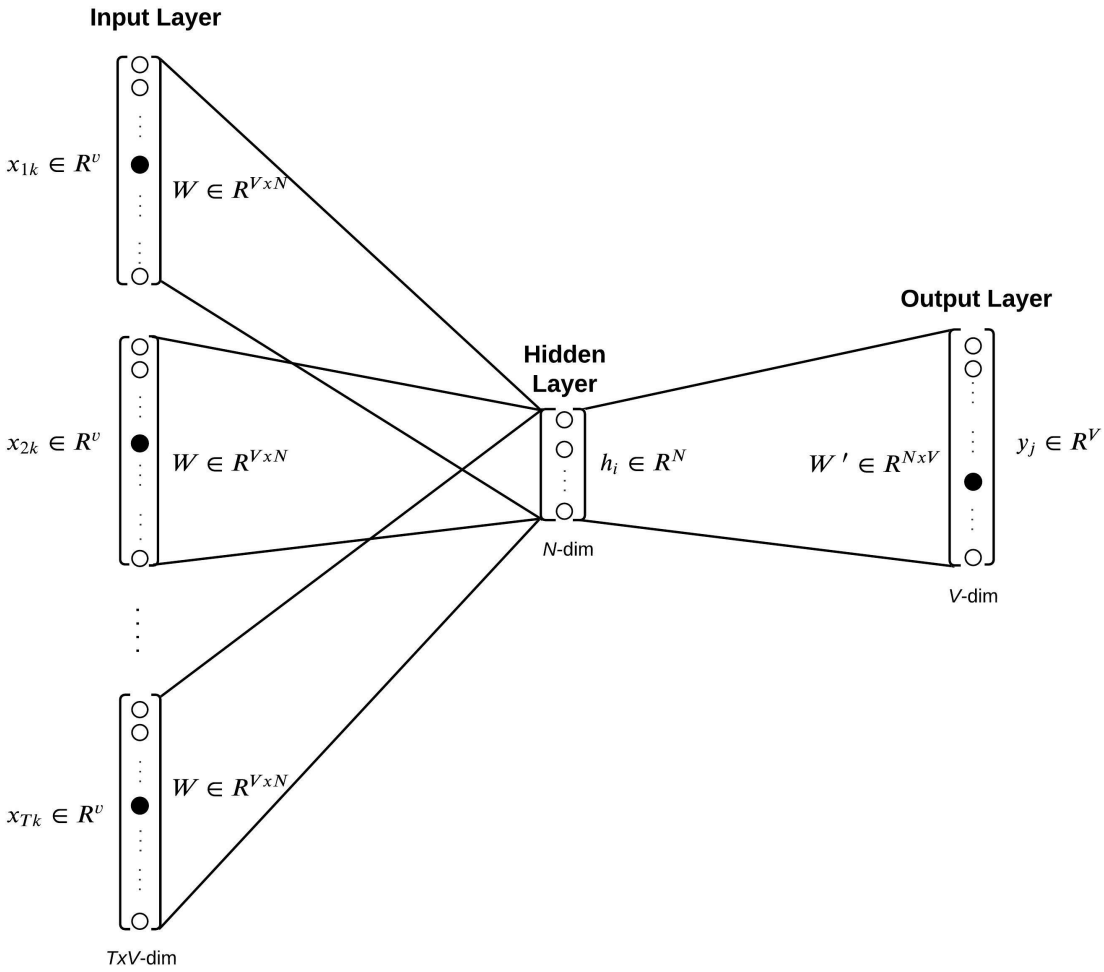

Figure 2.3. CBOW Model.

As a consequence of this, the model takes the average of these context vectors, and h as defined in Equation 2.8 needs to be updated by

$$\begin{aligned} h &= \frac{1}{T} W^T \left(x_1 + x_2 + \cdots + x_T\right) \\ &= \frac{1}{T} \left(v_{w_1} + v_{w_2} + \cdots + v_{w_T}\right)^T . \end{aligned} \tag{2.13}$$

Here T denotes the number of context words and $w_1, \cdots, w_T$ are given context words. The loss function and output layer are the same with one context model. Similarly, it can be defined as follows:

$$\begin{aligned} E &= -\log p\left(w_O \mid w_{I,1}, \cdots, w_{I,T}\right) \\ &= -u_{j^*} + \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right) \\ &= -{v'_{w_O}}^T \cdot h + \log \sum_{j'=1}^{V} \exp\left({v'_{w_j}}^T \cdot h\right) . \end{aligned} \tag{2.14}$$

Unlike CBOW, the skip-gram model's objective is predicting the context words for a given target word which is similarly in the middle of given context words. Hence, Figure 2.1.2 represents a target word on the input layer and multiple context words are on the output layer.

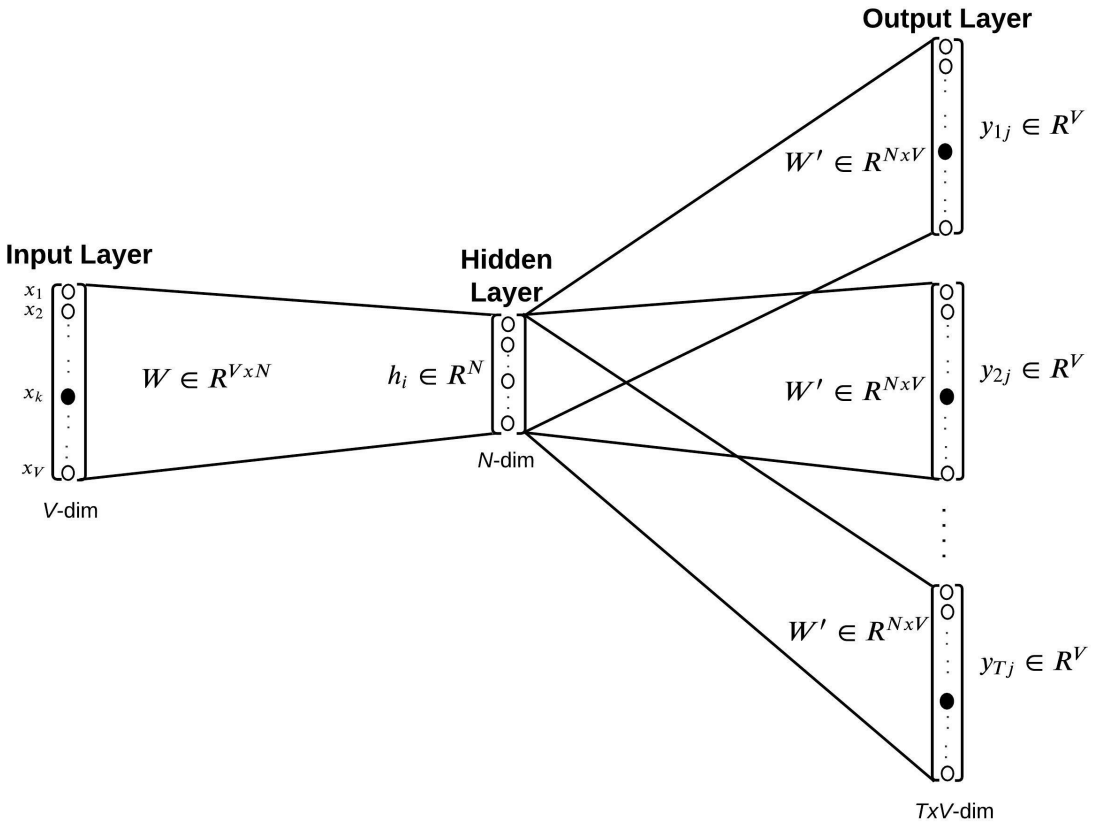

Figure 2.4. Skip-Gram Model.

Resulting from this, the output layer populates the T multinomial distribution while using the same weight matrix W', and the Equation 2.15 is reformulated as follows

$$p\left(w_{c,j} = w_{O,c} \mid w_I\right) = y_{c,j} = \frac{\exp\left(u_{c,j}\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)} \tag{2.15}$$

and then the loss function for skip-gram model can be defined as

$$\begin{aligned} E &= -\log p\left(w_{O,1}, w_{O,2}, \cdots, w_{O,C} \mid w_I\right) \\ &= -\log \prod_{c=1}^{C} \frac{\exp\left(u_{c,j_c^*}\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)} \\ &= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right). \end{aligned} \tag{2.16}$$

The lost functions are described above use the softmax function however computational cost comes to problem because it needs to calculate probability value in per training iteration for each word in the normalization part of the softmax equation. Therefore, many alternative methods such as hierarchical softmax and negative sampling are proposed to obtain efficient loss functions.

The hierarchical softmax is an approximation of the softmax function. It replaces embedding model's output layer with a multi-layer binary trees which their leaves corresponding to the words in the corpus. This enable a speeding up of the computation dramatically by decomposing probability estimation of the context word with calculated probability for each inner nodes in the tree instead of computing computationally expensive normalization part of the softmax function [15]. The binary tree has V leaf nodes that represent each product in the corpus and V-1 inner nodes. This means that there is a unique path starting from root node and traversing over the tree's inner nodes for each word, can be seen Figure 2.1.2.

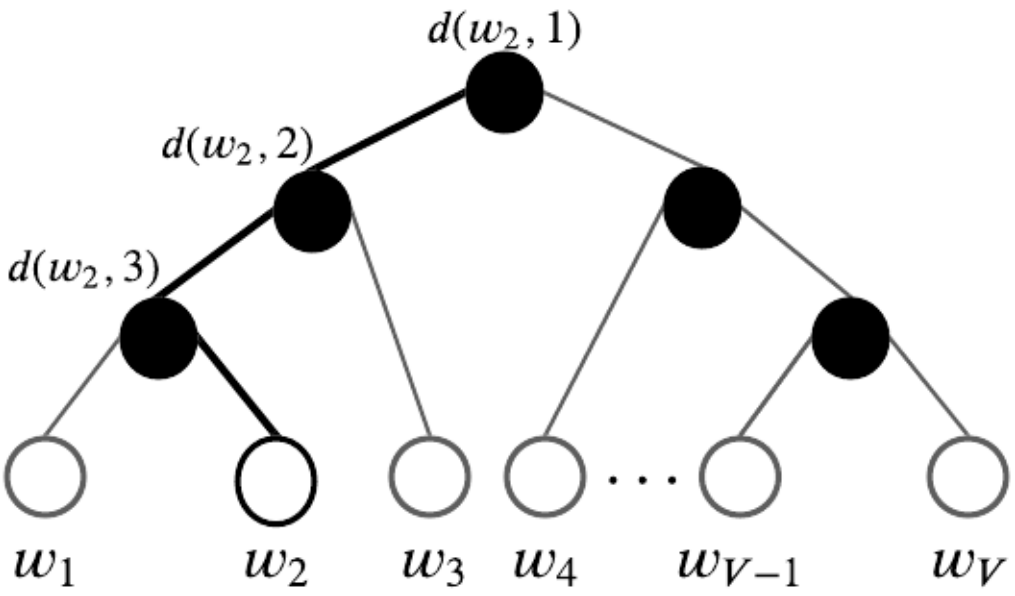

Figure 2.5. Hierarchical Softmax.

In contrast to softmax function, the hierarchical softmax uses vector representation $\mathbf{V}'_{n(w,j)}$ for each *V-1* nodes instead of using vector representation of words. In addition, a probability value is assigned for traversing left or right at each inner nodes in the tree which can be defined at an inner nodes n by

$$p(n,\ \text{left}\ ) = \sigma\left(\mathbf{v}_n'^T \cdot \mathbf{h}\right) \tag{2.17}$$

$$p(n,\ \text{right}\ ) = 1 - \sigma\left(\mathbf{v}_n'^T \cdot \mathbf{h}\right) = \sigma\left(-\mathbf{v}_n'^T \cdot \mathbf{h}\right) \tag{2.18}$$

where $\sigma(x) = 1/(1+\exp(-x))$ sigmoid function and $\mathbf{h}$ is the output value of the hidden layer, which equals Equation 2.13 in the CBOW. In this way, the probability of being a context word $w$for a given target word w_I can be expressed as

$$p\left(w \mid w_I\right) = \prod_{j=1}^{L(w)-1} \sigma\left(||n(w,j+1) = \text{ch}(n(w,j))|| \cdot {\mathbf{v}'_{n(w,j)}}^T \mathbf{h}\right) \tag{2.19}$$

where $\text{ch}(n)$ denotes the left child of unit n; $\mathbf{v}'_{n(w,j)}$ is the vector representation of the inner unit $n(w,j)$ and $||x||$ is a function is defined as

$$||x|| = \begin{cases} 1 & \text{if } x \text{ is true} \\ -1 & \text{otherwise} \end{cases} \tag{2.20}$$

and Equation 2.19 also proves that

$$\sum_{w=1}^{W} p\left(w \mid w_I\right) = 1. \tag{2.21}$$

The loss function for hierarchical softmax can be driven by

$$E = -\sum_{j=1}^{L(w)-1} \log \sigma\left(n(w,j+1) = \text{ch}(n(w,j)) \cdot {\mathbf{v}'_{n(w,j)}}^T \mathbf{h}\right) \tag{2.22}$$

Another alternative loss function to softmax function is negative sampling which is introduced by Mikolov et al. [16]. They proposed a loss function aim to identify given a target word from the k negative samples drown from a noise distribution $P_n(w)$ utilizing logistic regression. As a consequence of this, it saves us from a significant computational cost to compute and update many output vectors for each iteration in the softmax based methods since they are limited to sample count in the negative sampling. The loss function of negative sampling can be described as

$$E = -\log \sigma \left({\mathbf{v}'_{w_O}}^T \mathbf{h} \right) - \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma \left(-{\mathbf{v}'_{w_j}}^T \mathbf{h} \right) \tag{2.23}$$

where w_p is positive sample and $\mathbf{v}'_{w_p}$ is its output vector and $\mathcal{W}_{\text{neg}} = \{w_j \mid j = 1, \cdots, K\}$ negative samples drown according to $P_n(w)$. Here the value of k can be differentiated according to dataset volume and word frequency, which is recommended within the range 5-20 in the small dataset while it can be 2-5 in the large dataset.

The word embedding methods mentioned above efficiently represent relationship between words and have been shown state of the art performance in many word oriented NLP tasks. However, it comes to the document and paragraph level tasks, these models miss semantic relationship or order information of multiple words in the sentence or document representation problem is token into consideration. Mikolav and Lee proposed the Distributed Memory Model of Paragraph Vectors (PV-DM) which is motivated from the CBOW model to address this problem [16]. Figure 2.1.2 emphasizes the PV-DM architecture, it works in similar manner with CBOW. Here an additional paragraph vector p, which is one-hot encoding binary vector $p = \{p_1, p_2, \ldots, p_M\}$ compromising every paragraph in the corpus, is added to CBOW input layer with a $M \times N$ dimensional D weight matrix. After that, a paragraph vector, which is specific to word vector, is digested as an input in addition to word vector. Moreover, Equation 2.8 needs to be reformulated with D matrix, which consist of a vector mapping for each paragraph and W weight matrix by taking average or concatenating them. The remaining CBOW equations excluding Equation 2.8 can be used in the similar way in the PV-DM.

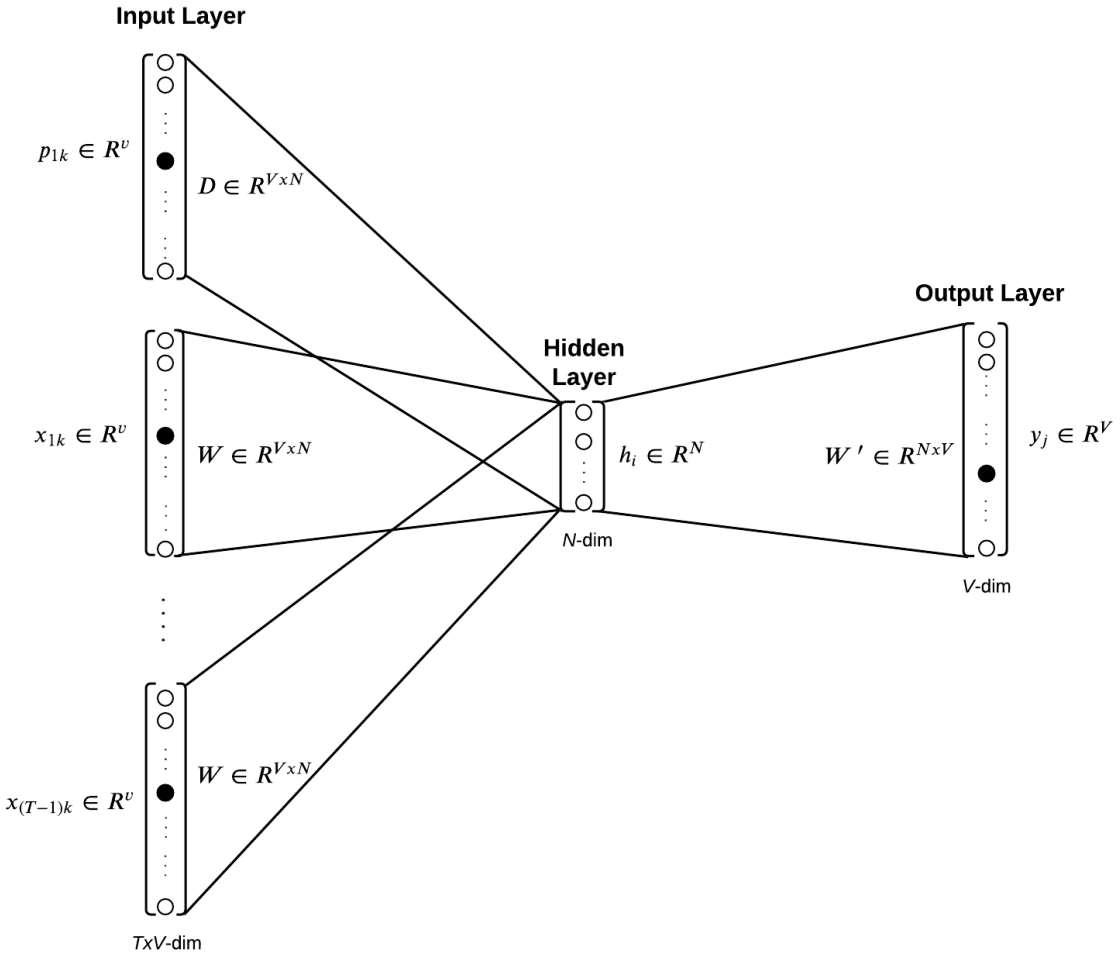

Figure 2.6. PV-DM Model.

3. LITERATURE REVIEW

This section summarizes the information about various studies and concepts that help in undertaking a research area in the recommendation system, especially on large and highly sparse datasets. In particular, these studies have focused on two main categories, both to be detailed in this section: collaborative filtering, and deep learning based approaches with their ancillary items, wherein each of them is one of the cornerstones of the recommendation system. Also, the section presents assessment criteria, implementation difficulties, and solutions. Then, it discusses studies on the recommendation system on second-hand platforms. Finally, it highlights the similarities and differences of this thesis with other studies.

Collaborative filtering is a popular method compromising the vast majority of the studies in this area [1]. It utilizes user's explicit feedback, such as rating of the product or implicit feedback [9] that are inferred from users' activity in the platform, to obtain a user-item matrix and then predict ratings or interaction possibility for a given product as a result of the calculations on this matrix. CF methods are studied under two main categories: memory-based and model-based approaches.

In memory-based approach, predictions are made on a pre-calculated similarity database and have been classified into two categories based on similarity type: user-to-user and item-to-item CFs [17]. User-to-user CF assumes that like-minded users have a similar preference, hence they initially find similar users thanks to similarity metrics that are calculated on their previous co-rate/co-purchase attempts. After that, they use the average of a certain number of similar users' previous rates in order to obtain a score for products that have not been discovered by the user. Unlike the user-to-user CF, item-based approach exploits products' co-rating/co-purchasing information to measure the similarity between products. Both of these approaches suffer common CF problems such as sparsity, cold start, and scalability.

Linden et al. applied item-to-item CF in the product recommendation problem and proposed that item-to-item CF is more scalable than the user-based approach while considered computation cost [10]. They also collected better results with respect to the user-based approach in their e-commerce dataset. Similarly, Sarwar et al. proposed different item-based CF techniques to handle sparsity and scalability problems [9].

The model-based approach is also used to solve the drawbacks of the memory-based approach. In the model-based approach, algorithms make use of the product's context information to infer a relationship between user and product, especially new products that have a few interactions. In this way, the main goal is addressing the cold start problem. However, they may face overspecialization problems due to insufficient or missing information about products [9].

The model-based approach that requires a trained model for prediction are based on favorable machine learning methods mostly using a Bayesian network, clustering, decision trees, and matrix factorization techniques [17]. To overcome sparsity problems, matrix factorization technique has become very popular in recent studies, thanks to their impressive prediction performance. The technique uses a mathematical representation to reveal the latent feature underlying the relation between the user and product by splitting a user-product matrix into user and product matrices. It applies dimension reduction techniques such as singular value decomposition, probabilistic, and non-negative MF on sparse user-product matrices [18]. That being said, they compel high computational costs for big datasets. A comprehensive study on how to implement the MF technique on implicit feedback was conducted by Hu et al. They introduce the item preference value for users similar to ratings by inferring from their historical data and then use alternating-least-squares methods to optimize proposed objective function for dimension reduction on user-item preference matrices [19]. Besides these, there exists another approach called hybrid models which combine these methods mentioned above to overcome individual drawbacks of algorithms while taking advantage of each one. Burke summarized some existence and possible hybrid recommender systems according to hybridization techniques and suggested a hybrid recommender framework for restaurant recommendation [20].

Another widely discussed topic in the recommendation system literature is how to evaluate algorithms and what evaluation metrics are. Shani and Gunawardana [21] categorized most used prediction metrics into three categories with respect to prediction type: rating accuracy, usage prediction, and accuracy of item ranking. For measuring rating accuracy, they describe Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and their normalized versions. In the usage prediction problem, the classical machine learning evaluation metrics such as precision and recall were explained together with their extended top N version. In addition, Karypis studied evaluation of the top N recommendation, specifically item-based approach, and then obtained 27% better prediction results with a training period many times faster than user-based approach [9].

Su and Khoshgoftaar (2009) have also represented a general overview of the CF approach by discussing its challenges like scalability, efficiency, sparsity, and also data manipulation, its categories that are considered as memory-based, model-based, and hybrid. Additionally, they study its evaluation metrics for the accuracy of the recommendation based on recall and precision, RoC sensitivity and mean absolute error (MAE) [17].

Recently, many studies have been centered around deep learning-based approaches in the literature [22] [23]. Among these approaches, some researchers intend to solve current challenges by extending classical models with deep learning methods, whereas some of them want to introduce new techniques with reformulation of the state-of-the art deep learning (DL) models, the success of which has been proven across many domains such as NLP and computer vision. For instance, many recent product embedding models originated from the word2vec techniques such as continuous bag-of-words (CBOW) and skip-gram (SG) that identifies semantic relationship between words according to their neighbouring words in documents [13].

Barkan and Koenigstein assumed that item baskets or co-seen items are equivalent to word order in sentences, and then applied the skip-gram technique to generate low dimensional item vectors representing an item-item relation similar to item-to-item

CF [24]. Therefore, they compared the performance of their model called item2vec and baseline SVD-based item-to-item CF model on two different dataset, and it was stated that item2vec outperformed the item-to-item CF method on their experiments.

Another example of the NLP oriented methods in the recommendation system is customer2vec technique that is based on the paragraph2vec method [16]. In addition to sequence of the together viewed or purchased item buckets, this approach considers user information as an input source in the model training; therefore low dimensional vector representation of both user and item can be taken to handle item similarity and recommendation personalization problems.

Grbovic et al. introduced a bagged-product2vec method as an extension of the product2vec, and then compared this method with product2vec, user2vec, and hybrid models using clustering techniques to find the best top N accuracy for personalization on product advertisement to Yahoo Mail Users [25]. The user purchase history that was gathered from their purchase receipts was utilized on the model training. For evaluation, they conducted an offline experiment on 29 million users' purchase dataset and online test in the production system; it resulted in a 9% CTR uplift in comparison with the existing algorithm. They also dug into proposing decay factors to take into consideration time effects in the recommendation and find proper look back dates in the training of algorithms.

Grbovic and Cheng regard the sequence of the users' interactions during their web session as similar to the role of the sentences in NLP studies [26]. With this assumption, the various product and user embedding methods were applied with classical feature sets to address personalized search ranking and similar item recommendation tasks on the Airbnb marketplace, which has an extremely unique product dataset.

Furthermore, the Meta-Prod2Vec method was proposed as an extension of the product2vec techniques by Vasile et al. [27]. In this approach, item metadata information is included as well as the sequence of products, hence it alleviates the cold start problem. Likewise, Pfadler et al. focused on extending the skip-gram method with side

information such as item and product metadata while capturing user behavior asymmetry. Many different extensions of these word embedding techniques can be seen in the literature [12, 28, 29].

The learning to rank, sometimes called machine learned ranking, is an information retrieval technique that has been applied to product recommendation or search problems. There are many academic works using it directly or in combination with hybrid approach [22]. Browman et al. have focused on handling specifically scarce information and uniqueness problems, which are also common for product recommendation on second-hand platforms, also encountered in this thesis [6]. They suggested two-staged architecture including recall and ranking steps; recall is responsible for candidate item generation according to co-view, category and title similarity information for selected item while ranking that orders candidate products with respect to calculated purchase probability. The second step leveraged the pointwise learning to rank approach, which can be modeled as a classification problem, and then the candidate items were sorted after the classifier's probability prediction. They answered the question of how to determine features, sampling techniques and model evaluation in detail in the study.

To address product uniqueness problems in second hand platforms, the neural network-based models, especially word embedding techniques, have been widely discussed in the literature. Galron et al. combined two neural network (NN) architectures called item embedding network and prediction network to estimate product similarity from implicit feedbacks that were gathered on Ebay marketplace [6]. In the item embedding network, embedding of items was generated from title, category, and aspects features by means of continuous bag of word and recurrent neural network methods. After that, they concatenated seed and candidate item embedding vectors on prediction network architecture to estimate similarity between them. In this way, they have shown promising results in comparison to classical item-item CF methods through their top N product recommendation experiments using mean recall and mean reciprocal rank metrics. Similarly, Wang et al. presented a personalized recommendation framework based on affinity score obtained by the dot product of user and item embeddings [6].

To be able to generate user representation, the multi model user embedding framework using CBOW and recurrent neural networks was proposed, and then applied to the dataset consisting of the user's previous events and some context information such as item metadata, search texts. As a result of the performed experiment in both their offline dataset and online environment, they achieved dramatic improvement on their existing system in terms of precision@n, engagement metrics. Using both item and user embedding helped in grasping and considering the user's previous preference, which may be called personalization, on the recommendation step in addition to the many advantages of the item embedding-based approach mentioned above.

In light of the aforementioned studies, this thesis proposes a new recommendation model pipeline which may be applicable to second-hand platforms. The main motivation is to address representation problem of millions of unique products and users with a few interactions in a scalable way, and afterwards, to build an model pipeline using these representations to provide personalized recommendations. In product representation, we have been inspired by the word2vec approach because it has been successfully adopted to recommendation system in several studies [24–26, 30] and has shown superior performance in comparison to classical CF methods. When the user representation is taken into consideration, Grbovic et al. [25] and Phi et al. [12] introduce the models using paragraph2vec approach to generate user-based recommendation model and observe good performance in their experiments, in the same way, the proposed model uses paragraph2vec approach to build user2vec model, which computes required user vectors. However, user vectors are utilized to provide input features to ranking step instead of using product recommendation directly. Ultimately, user and product representations are combined in a two staged model pipeline, which is widely preferred to narrow down candidates product pools in the literature, especially dealing with recommendation problem including datasets which use almost exclusively unique items, such as second hand, streaming and social media platforms [6, 22, 31]. Contrary to these studies, the study in this thesis evades highly customized and computationally expensive features to provide an efficient model pipeline which can be build on any click-stream dataset.

4. METHODOLOGY

This section introduces the proposed approach for personalized product recommendation using users' various implicit feedback as specific to second hand e-commerce platforms, which have millions of unique products with a few interactions. The main goal is providing the most similar products taking into account users' taste for a given target product in a scalable way. The proposed approach is built on three stages respectively to achieve the goal: user-product corpus, candidate generation and ranking, as represented in Figure 4. Each stage in the proposed approach will be explained in detail in the following sections.

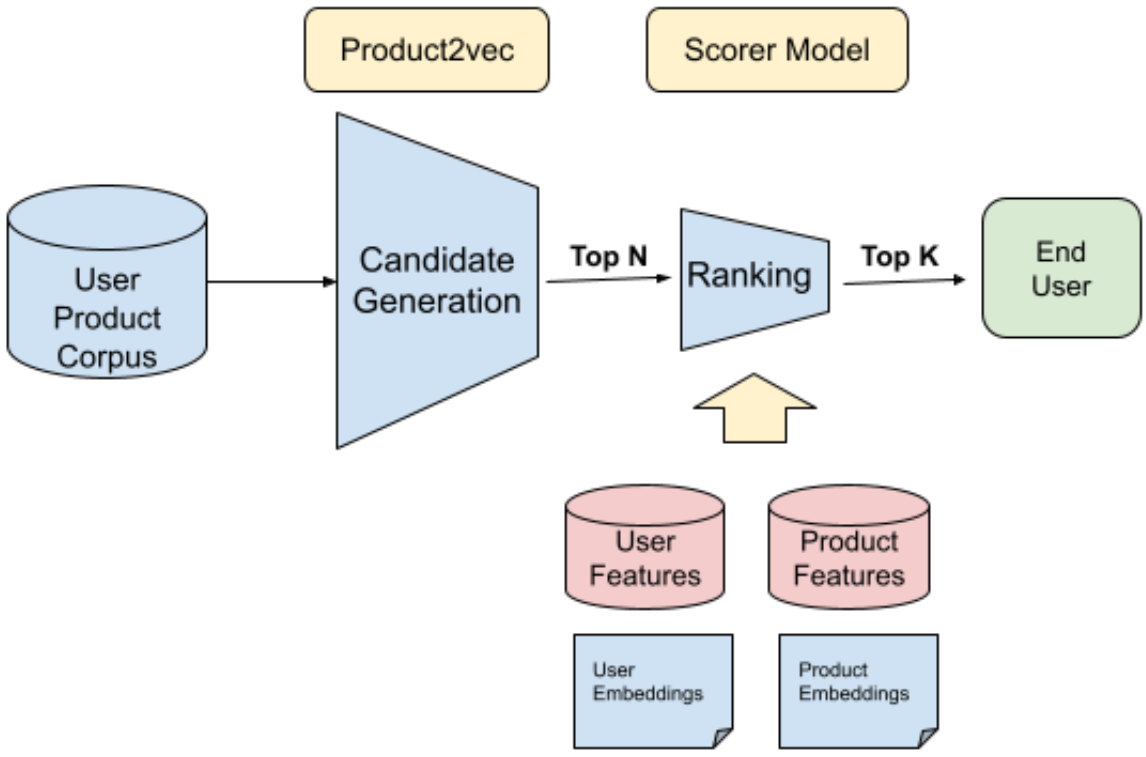

Figure 4.1. Proposed Model Pipeline.

Algorithm 1 Model Pipeline Pseudocode

function MODELPIPELINE(Users' clickstream data)

Generate user-product corpus

List<Product>topN = CANDIDATEGENERATION(*targetProduct, chosenUser, productCorpus*)

List<Product>topK = RANKING(*List<Product>topN, List<Double>cosineSim*)

return Top K similar products list

end function

function CANDIDATEGENERATION(*targetProduct, chosenUser, productCorpus*)

Calculate the cosine similarity for each product with respect to target product

Find top N similar products based on the best product embedding model

return Top N similar products and their cosine similarities

end function

function RANKING(*List<Product> topN*, *List<Double> cosineSim*)

interactionProbability = SCORERMODEL(*List<Product> topN*, *List<Double> cosineSim*)

Sort *interactionProbability* and choose top K value

return *T*op K products based on sorted *interactionProbability* ▷ K < N

end function

Figure 4.2. Model Pipeline Pseudocode.

4.1. User - Product Corpus

The dataset which is required for the proposed models need to comprise user session logs such as user's clicks, likes, purchase actions on products, which denote

user positive signals on products. Algorithm 2 represents the steps of generating user-product corpus. The key step in it is to being filtered out products which have less than c interactions, in this manner, the cold-start problem is outside the scope of our approach. As an output of the algorithm, a user-product corpus is produced using fewer products than existing.

Algorithm 2 Generate User-Product Corpus

Input: $i \leftarrow interaction$; ▷ Assume all action types are called interaction

$Map < Product, Interaction > productInteractionMap$;

Output: Processed a user-product corpus consisting of all interactions

foreach *Product p : allProducts* **do**

  Calculate *interactionCount* for p

  if $interactionCount > c$ **then**

    $productInteractionMap$.put(p, i)

  end

end

Sort $productInteractionMap$ based on $interactionTimes$ for each *user*

Remove consecutive same products from $productInteractionMap$

Combine all interactions with $productInteractionMap$

Create user-product corpus with all interactions

return *user-product corpus*

Figure 4.3. User-Product Corpus for Model Training Generation.

In addition, a validation dataset is need to be populated from the user-product corpus to obtain the best hyperparameter setup and candidate generation model. The proposed evaluation methodology is based on the assumption that products which are interacted by the same user are similar to each other, so the model performance can be measured by the prediction performance on other products in a user's product bucket for a given product in that bucket. With this notion, the steps of in Algorithm 3 are applied to create a validation dataset. Throughout these processing steps, the proposed

approach obtains k target products and m associated users who interacted with the target products and their product lists from the users. These user-product pairs are excluded from training data in the model training phase.

Algorithm 3 Validate Dataset

procedure VALIDATIONDATASET($userList, productList$)

Choose randomly m products interacting with at least k users different from each other

For each product, choose l users randomly

Take users' unique product buckets based on these users' interaction history

Combine target product with l selected users' product buckets

Create a mapping for target product and associated product list

end procedure

Figure 4.4. Validation Dataset Generation.

4.2. Candidate Generation

The candidate generation step enables to list top similar N products from the candidate product pool and their cosine similarity $\cos(\vec{i}, \vec{j})$ for a given target product. The embedding techniques are preferred in our proposed model because they have been shown state-of-the-art performance on NLP tasks, furthermore, successfully adopted from other domains [13]. Assuming that there is a set $\mathcal{S}$ of application sessions gathered from N users and each session has a product sequence $s = (p_1, \ldots, p_M) \in \mathcal{S}$ consist of user's time ordered actions on products including click, purchase, comment, like, represented as in Figure 4.5 and Table 4.1, these sequences can be accepted as a sentence and combined sessions can be supposed to document.

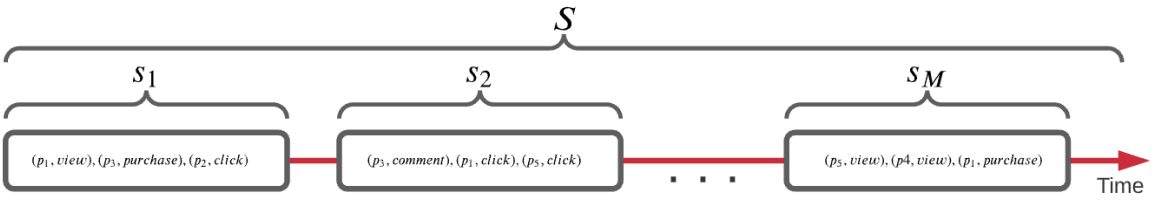

Figure 4.5. Sessions Comprised of User's Time-Ordered Actions.

Table 4.1. User - Action Sequence.

UserID	Actions based on product item $\mathbf{p}_k$
$\mathbf{U}_1$	($\mathbf{p}_1$,view), ($\mathbf{p}_1$,click), ($\mathbf{p}_5$,click), ($\mathbf{p}_1$,purchase)
$\mathbf{U}_2$	($\mathbf{p}_3$,purchase), ($\mathbf{p}_4$,view)
...	...
$\mathbf{U}_N$	($\mathbf{p}_2$,click), ($\mathbf{p}_5$,view)

With this notion, popular word embedding techniques can be utilized; skip-gram and CBOW to obtain low dimensional continuous vector representation of products. In the same way, a fixed sized sliding window moves along the product sequence and generates context and target products of these models, which can be seen in Figure 4.6. After that, these products are denoted V dimensional, which is equal to product count in the corpus, one hot encoded vector and passed to input and output layers. Subsequently, the models learn to predict context or target products with a feed forward neural network architecture during model training, therefore, embedding layers gives vector representation of products and enables to understand and assess the relationship between products by means of cosine distance.

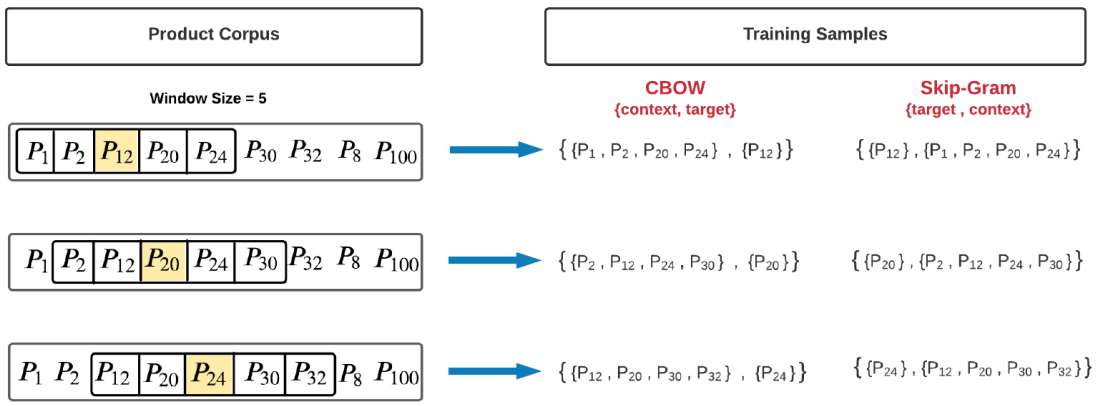

Figure 4.6. An Example Training Sample Generation with a Sliding Window.

There are many studies discussing the fact that CBOW and skip-gram can outperform each other according to corpus size and word frequency [13], therefore both approaches have been compared to obtain best performance with respect to model training time. Given that the word embedding techniques are unsupervised models, there is no direct evaluation methodology. The proposed approach uses the precision@n and diversity@n metric as evaluation metrics. Precision@n metric equals the ratio obtained by the positive recommendation count divided by total recommendation count that comprises fixed N recommendation for each target product and can be defined as

$$Precision@n = \frac{\#\ relevant\ recommendation@n}{\#\ total\ recommendation} \tag{4.1}$$

while diversity@n metric represents how many distinct products are in the total recommended products list [32] and can be expressed as

$$Diversity@n = \frac{\#\ distinct\ products\ in\ recommended\ product\ list}{\#\ total\ recommended\ products}. \tag{4.2}$$

Hyper-parameters; minimum word count, vector dimension, window size and epoch in the training of both models, as well as loss function selection, are optimized according to this evaluation metric to obtain the best performance metrics within a scalable training period. Since the given dataset consists of millions of unique products and users, negative sampling and hierarchical softmax loss functions have been used to improve our performance and training time because they decrease training time dramatically. Gensim open source NLP library [33] is used in the product2vec and user2vec models training.

4.3. Ranking

The ranking step proposes a binary classification model, which is called a scorer model. The scorer model uses a multi-layer perceptron (MLP) model, is a feed-forward neural network architecture with multiple hidden layers between the input layer and the output layer with sigmoid function [34], to estimate a user's interaction probability for the top N similar products to a given product. After that, the top N similar products are re-ranked according to this probability value and top K products are recommended, where N is grater than K.

The model architecture is built by an approach similar to a tower pattern [22], which starts from a zero hidden layer that is equal to the linear factorization with a sigmoid function, after that, incrementally increases hidden layer count and network size until observes a significant improvement in evaluation metric. In the input layer, the product and user embedding vectors are employed. Here product embedding vectors are computed by the product embedding model in the candidate generation phase, while a PV-DM paragraph embedding technique is used to obtain the user embeddings, which represent user preference and relationship between users and products.

5. EXPERIMENTS AND RESULTS

This section initially describes our dataset as used in both offline experiments and the A/B test in the production system. Then, it gives detailed information about the experiments being conducted. At the close, it interprets their results. The offline experiments run on a validation dataset which are populated from the platform's historical data, and have the popularity bias specific to the platform's recommendation system. However, the A/B test is conducted over two randomly assigned group user samples in the production system, so it is respectively robust when it comes to popularity bias.

5.1. Dataset

Our data comprises user session logs gathered from a popular online second hand platform which have various ranges of products over the course of a month. It includes user's clicks, likes, comments, bidding, and purchase actions, which denote user positive signals on products as well as products' information such as size, color, and price. In the case of 600 thousands users over 10 millions unique products; some portion of them get few interactions because 100 thousands of new products per day are added to the platform by users and getting interactions from the user takes time. Hence, the c value in Algorithm 2 is set to 5 and products which have less than 5 interactions are filtered out. This product group makes up 25% out of total product count while their transaction volume equals 82% out of total transaction volume. Furthermore, it is partitioned according to a 7 days period because some research and our insights from the data support that the product popularity starts to decrease dramatically after 5 - 7 days since its added or products mostly are sold within this period [25].

All these considered assumptions are also crucial for our A/B test in the production system. The best tuned product embedding model was trained on one of the product categories and deployed to the next product recommendation page in the production system. It was conducted A/B tests over 19 days with 1.5M sessions and

996K users. Over the course of A/B test, data manipulation steps are automatically performed to update training dataset every hour.

5.2. Experiments

In this section, the models' training pipeline and hyperparameter selection criteria are described. The proposed approach requires three separate models which need to be trained; product2vec, customer2vec and a scorer model for the proposed model pipeline. Additionally, an item-to-item CF as a benchmark model is trained on the same training and validation dataset. The model pipeline requires evaluating the models individually and as a pipeline on the same validation dataset with the same evaluation assumptions. Thus, the product2vec and customer2vec models are tuned on the following experiments, therefore the best performing model's outputs, which are embedding vectors of top N similar products, are digested from our scorer function and it calculate probability of being interacted. In the same way, the scorer model is optimized to find the best parameter setup maximizing the same evaluation metrics with embedding models. Finally, these top N product are sorted according to probability of being interacted by user and top K products are recommended.

Prior to carrying out offline experiments, the training and validation dataset should be populated. The data manipulation steps in Algorithm 2 are applied on our dataset to create the training dataset which consist of 500 thousands users and their interaction with 2.2 millions unique products over a 7 days periods. Here C value set to 5. In addition, the validation dataset is populated with respect to Algorithm 3 in the configurations; $m = 10\text{K}$, $l = 5$ and $k = 3$. In this way, 10 thousands unique products associated with *3* users and their product buckets comprise our validation dataset.

5.2.1. Product2vec Model

For the purpose of finding most similar top N products in the candidate generation step, product2vec model is proposed and the most widely used CBOW and skip-gram approach are compared during experiments to determine best model and hy-

perparameter configuration. Here many studies which explore the similar approaches to our work in the literature are reviewed, then baseline hyperparameter setup and optimization range are selected according to their findings [6, 25, 26, 35]. The further details about these parameters and their tuning range, used in the proposed approach, are expressed in Table 5.1. Moreover, the minimum word count which denotes a word's required minimum frequency in the corpus and is set to 5 in our experiments; they are already filtered out in our training data. Additionally, negative sampling and hierarchical softmax loss functions are utilized with these hyperparameters in our experiments for both CBOW and skip-gram models. It also sets the negative sampling count to 5 in the experiments using a negative sampling method, which is recommended as default value, and selecting 5-20 words for small datasets whereas 2-5 words for large datasets are advised in different studies [13].

This work also applies the best performing hyperparameters setting in the product2vec model using CBOW approach according to our validation process since PV-DM is an extended version of the CBOW approach and has similar model structure.

Table 5.1. Parameter Settings for Training Product2vec Model.

Parameter	Values	Explanation
Vector Size	[16, 32, **64**]	Dimension of product embedding
Window Size	[5, **10**, 15]	Maximum distance between context and target product
Epoch	[5, 10, **15**]	Number of iteration on entire training dataset

5.2.2. User2vec Model

Like word embedding techniques, this technique is an unsupervised machine learning technique and its performance is evaluated by various NLP tasks in the literature [16, 36]. This work applies the best performing hyperparameters setting in the product2vec model using CBOW approach according to our validation process since PV-DM is an extended version of the CBOW approach and has similar model structure.

5.2.3. Item-to-Item CF Model

As a baseline model, the item-to-item CF model is preferred in this study. Since our dataset consists of users' implicit feedback, the matrix factorization approach mentioned in the background section, which is based on the *alternative least square* method on the implicit feedback dataset, is applied in order to build the best CF model on a user-product matrix that is generated from the same training dataset with embedding models. Like this approach, the utility value r_{ui} is calculated as an user's total interaction count on ith item and confidence level parameter is obtained by Equation 2.6, where α is set to 40 which is recommended in their experiments. The open source implicit library [37] that specializes in implicit feedback datasets is utilized to implement the model and tune according to parameters represented in Table 5.2.

Table 5.2. Collaborative Filtering Parameter Settings.

Parameter	Values
Latent Factor Count	[32, 64, 128, 256]
Iteration Count	[25, 50, 100]
Regularization Factor	[0.01]

5.2.4. Scorer Model

With the best performing product2vec and user2vec models, the candidate generation step of the proposed approach can be thought of as completed. Obtained product and user embedding vectors from these models are used as inputs to scorer function. The main goal is finding the best model predicting user interaction probability for selected product-user pairs in the validation dataset.

For the purpose of building the best MLP models, several candidate hidden layer architectures using rectified linear activation unit (RELU) are tuned with several hyparameters within selected scale are shown in Table 5.3. The candidate hidden layers

which starts with a sigmoid function without any hidden layer that can be accepted as an linear classifier, and is followed by layers of incrementally increased unit size, or newly added layers. Here batch size denotes number of samples are processed in each iteration, and max epoch shows maximum epoch count during experiments. The open source Keras library [38] is used to train models and utilized early stopping function in the library, which finishes the training process when model's evaluation metric is not improved during a selected iteration counts, is called patience. Therefore, the training process can be broken up before max epoch according to early stopping criteria. Prior to model training, a separate training dataset is required to scorer model because the user-product corpus includes only users' positive feedbacks. Thus, negative samples are populated for randomly selected user sets with the products which are not interacted with these users, by the way, a training dataset, which consists of a certain percentage positive sample and negative sample, is generated. As a consequence of this, another hyperparameter positive sample ratio, which represents rate of the positive sample to training dataset, is defined. In addition, optimizer is set to Adam and binary cross entropy is used as a loss function in experiments.

Table 5.3. Scorer Function Parameter Settings.

Hidden Layers	Max Epoch	Batch Size	Positive Sample Ratio
Zero	100	[32, 64, 128]	[0.2, 0.4, 0.5]
64 RELU	100	[32, 64, 128]	[0.2, 0.4, 0.5]
128 RELU	100	[32, 64, 128]	[0.2, 0.4, 0.5]
128 RELU - 16 RELU	100	[32, 64, 128]	[0.2, 0.4, 0.5]
128 RELU - 32 RELU	100	[32, 64, 128]	[0.2, 0.4, 0.5]
128 RELU - 64 RELU	100	[32, 64, 128]	[0.2, 0.4, 0.5]
128 RELU - 64 RELU - 16 RELU	100	[32, 64, 128]	[0.2, 0.4, 0.5]

5.3. Offline Experiment Results

This section discusses our evaluation methodology and proposed models performance based on our evaluation metrics precision@5 and precision@10 over experiments. In addition, diversity@5 and diversity@10 metrics are used to measure personalization. With assumption that products which are interacted by the same user are similar to each other, offline experiments were conducted on our holdout validation dataset, which consist of *10K* unique products associated with *3* users and their product buckets. In other words, our product2vec and scorer models aim at recommending N similar products that give the best precision value for each user-product pair in the validation dataset. The experiment results are discussed for each step of the proposed model pipeline as well as overall comparison in the following sections.

5.3.1. Candidate Generation Model

For the fitting the best baseline CF models, we conduct experiments with selected parameter scale represented in Table 5.2. As Figure 5.3.1 shows, it clearly demonstrates that the best results are observed with the high latent factors in both precision@5 and precision@10 metrics while the iteration count has no significant impact on the results. Consequently, the setting with latent factor count is 256 and iteration count is set to 50 gives the best results.

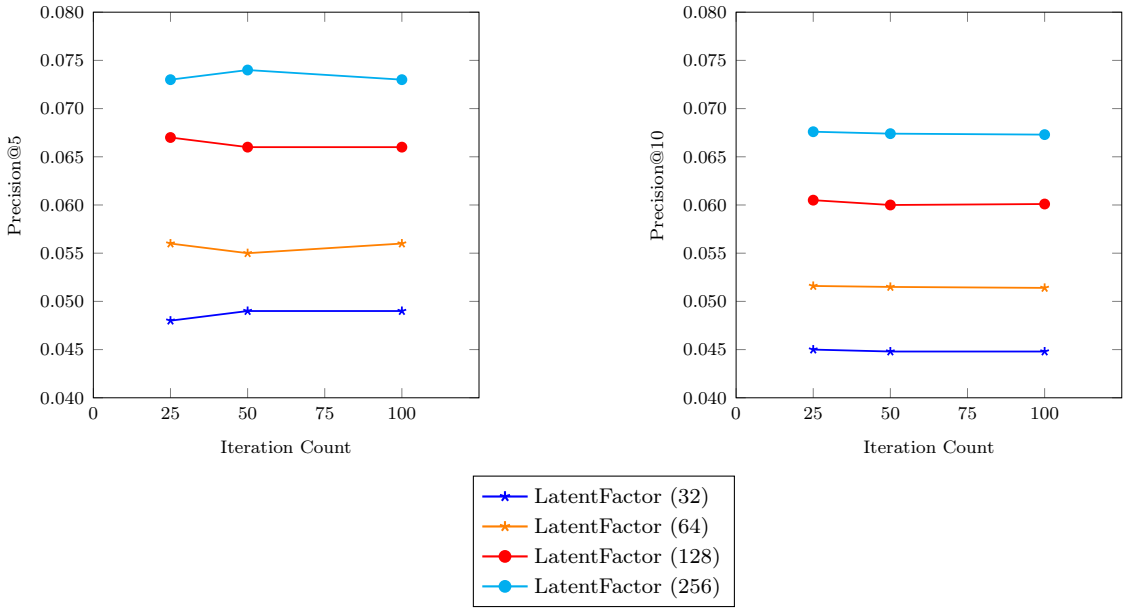

Figure 5.1. Precision@5 and Precision@10 Results for Collaborative Filtering.

In addition, the CBOW and skip-gram models are trained with common hyperparameters within selected intervals which can be seen in Table 5.1, and their performance evaluated on the same validation dataset. Figure 5.2 shows the precision@5 values on selected hyperparameters for both models with their loss function breakdown. It can be seen that hierarchical softmax provides better results than negative sampling in both CBOW and skip-gram models. This can be attributed to low product frequency in our training dataset now that it is underlined that the hierarchical softmax loss function shows good performance with infrequent words as pertaining to negative sampling in previous studies [39]. Furthermore, we see the best performance results when the dimension size is set to 64 in the both models. Note that in Figure 5.2, the skip-gram model consistently outperforms the CBOW model on all experiment trials and gives the best performing model using window size 10 and vector dimension 64 configuration, which has trained over 10 epochs with hierarchical softmax loss function. From the data in Figure 5.3, which shows the experiment results for precision@10 metric, can be seen that the same configuration gives the best results at skip-gram model for precision@10 metric as well.

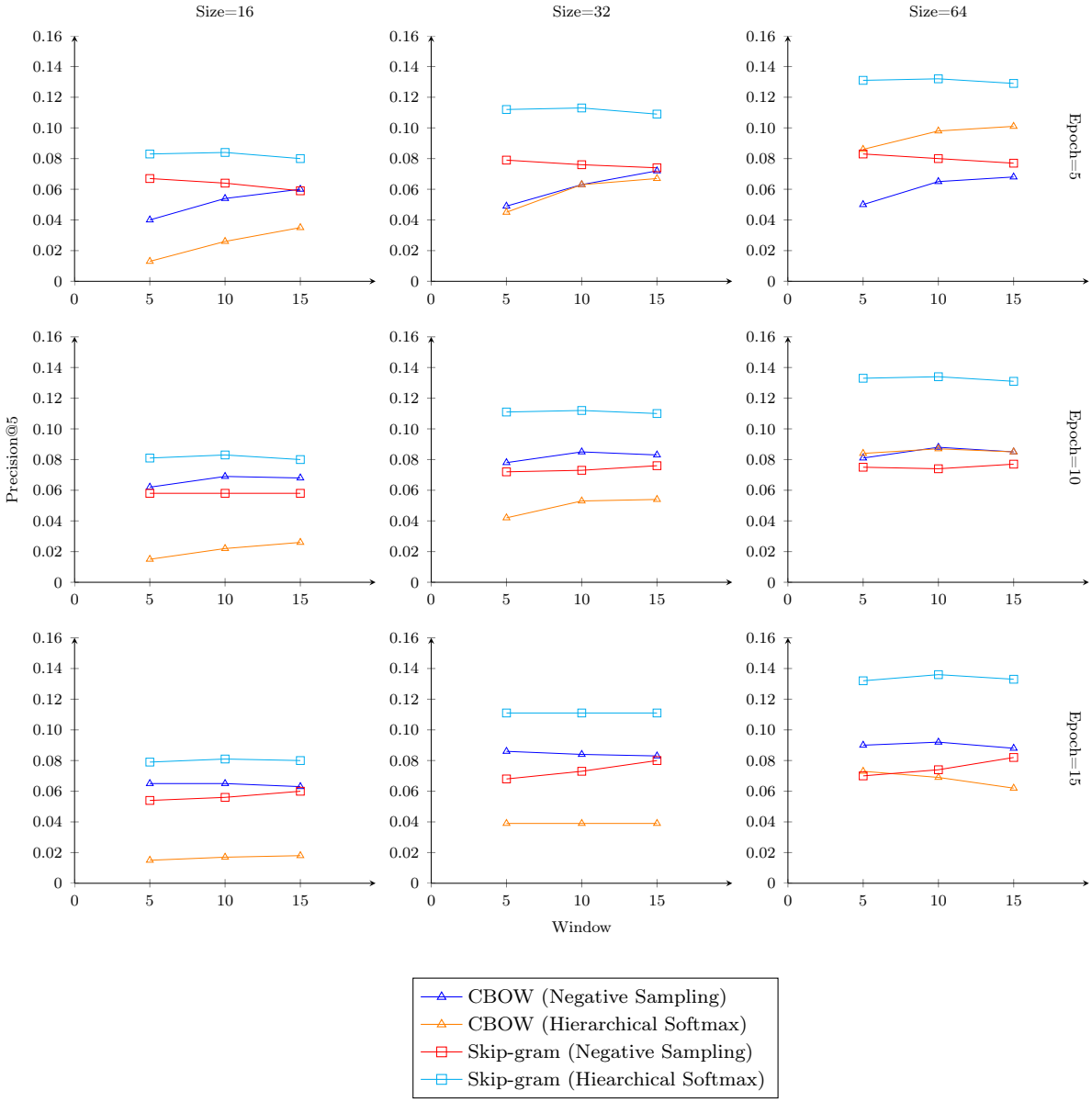

Figure 5.2. Precision@5 Results for Embedding Models.

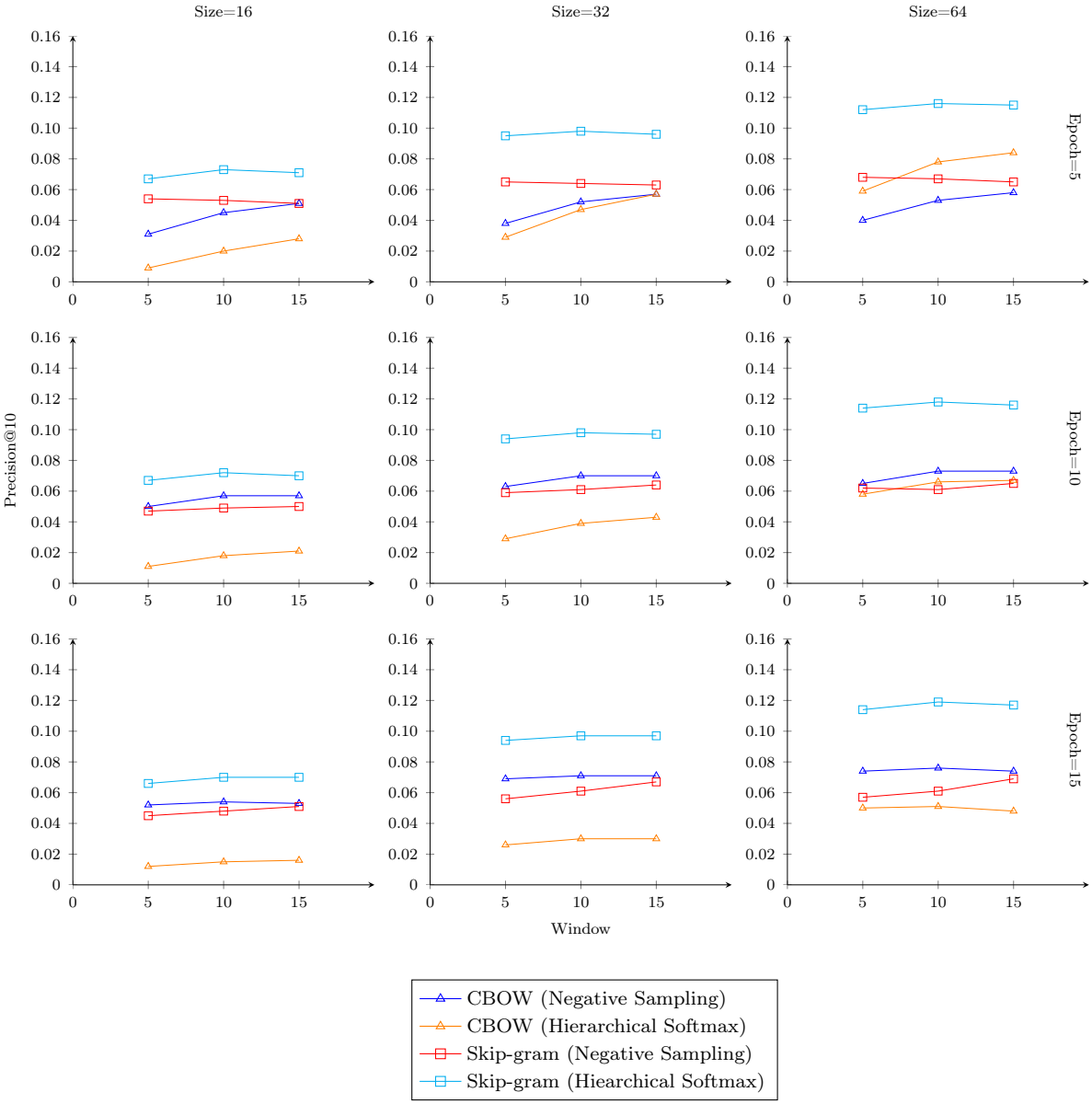

Figure 5.3. Precision@10 Results for Embedding Models.

5.3.2. Ranking Model

After obtaining the best performing candidate generation model, which is skip-gram, and parameter configuration over the experiments, the user2vec model is built with the CBOW models' best parameter setups where window size is set to 10, epoch equals to 5 and vector size is 64. Afterwards, the scorer model is tuned with hidden layer architectures and hyperparameters can be seen in Table 5.3, to predict interaction probability for a given user and product embedding pair according to AUC

metrics derived from Receiver Operating Characteristic (ROC) curve, which is classifier evaluation metric showing performance in various thresholds. The model trained on a dataset consists of 1.5M transactions whose positive sample rate is differentiated based on positive sample ratio parameter, and it is randomly split into 80% train and 20% validation datasets in experiments. Throughout the experiment process, there is a considerable improvement in all candidate MLP architectures when the positive sampling ratio is set to 0.4, hence the following results are filtered by this condition. The experiments' performance results can be seen in Figure 5.8, they support that hidden layer architecture consist of a 128 wide RELU hidden layer followed by a 64 wide RELU hidden layer and followed by 16 wide RELU hidden layer, as represented in Figure 5.4, is shown the best performance in the experiments. One of the findings from these results is that the architecture is built on only a sigmoid activation unit, which works in the same way as a linear binary classifier, does not fit a successful model and gives the worst performance result in the experiments.

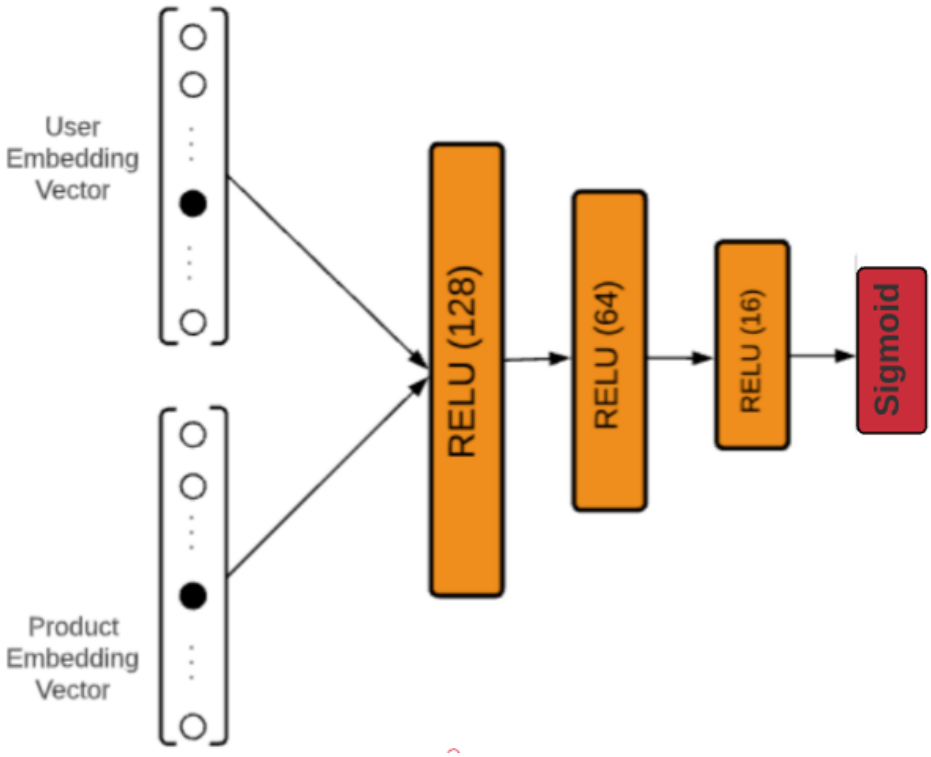

Figure 5.4. Best Performing Scorer Model Architecture.

Following these experiments, the scorer model performance is also evaluated on the same validation dataset with candidate generation models. Here the scorer model predicts interaction probability for given user and top N candidate products which are predicted from the best performing candidate generation model, where N is set to 15. Therefore, these products are sorted according to this probability and the model returns top K products, which is set to 10 in the experiments. Ultimately, the scorer model architectures' precision@5 performance with their best AUC value are shown in Figure 5.4. A significant finding from these results is a positive trend between AUC and precision values corroborating our expectation.

Table 5.4. Results for Hidden Layers Architecture.

Hidden Layers	AUC	Precision@5
Zero	0.5557	0.1234
64 RELU	0.5840	0.1215
128 RELU	0.7719	0.1221
128 RELU - 16 RELU	0.8154	0.1237
128 RELU - 32 RELU	0.8282	0.1238
128 RELU - 64 RELU	0.8401	0.1272
128 RELU - 64 RELU - 16 RELU	0.8568	0.1304

5.3.3. Model Comparison

Besides all models mentioned above, popularity-based model is also used for the comparison. It recommends the most popular N products from products which are under the same brand with a given product. All models including this one are evaluated on the same validation dataset, this allows for comparing models' performance with each other or overall model pipeline. Additionally, the diversity metrics are calculated for each best performing model in the experiments to investigate personalization and popularity bias in these models. The results, as shown in Table 5.5, indicate that scorer model and embedding models outperform the collaborative filtering-based model in both evaluation methods and the skip-gram models give the best precision values.

When it comes to diversity performance, the proposed model pipeline including scorer model shows the best diversity performance with respectively slight decrease in the precision values. As a consequence, the proposed model pipeline provides most diversified product recommendation as well as maintaining successful precision performance.

Table 5.5. Results for Best Tuning Models.

Models	Precision@5	Precision@10	Diversity@5	Diversity@10
Popularity-based	0.0676	0.0563	0.0327	0.0326
Collaborative Filtering	0.0742	0.0676	0.3181	0.3056
CBOW	0.1011	0.0844	0.3207	0.3102
Skip-Gram	0.1360	0.1192	0.3145	0.2996
Scorer	0.1304	0.1163	0.4302	0.3712

5.4. A/B Test Results

After carrying out the experiment offline, it supports that embedding models outperform the item-to-item CF model. In addition, we run a A/B test on a popular second hand platform's production system with the best performing embedding model which uses the skip-gram method and hyperparameters size=64, window=10, epoch=15 as well as hierarchical softmax loss function. Here the candidate generation step is accepted as a next product recommendation system and it is applied in the platform's similar item recommendation page which shows similar item list after customer clicks a product. A product category which takes high impression is selected for experiment. Before doing the experiment, useful operational metrics enable us to evaluate our model performance, additionally, model's impact on users' behaviour on platform in reasonable way. This means that it was examined under predetermined business metrics that are explained as follows:

- **CR** is a ratio for the number of users' clicks per number of impressions on the page

- **Trx/U** is a ratio for the number of transactions per users
- **R-Click/U** is a ratio for the number of clicks from the recommended products per users
- **R-Trx/U** is a ratio for the number of transactions from the recommended products per users

The experiment has ran during 19 days with approximately 996K distinct users at one product category: in total, 1.5M sessions. As a first step of the experiment, the current recommendation system in the platform was also observed under the business metrics mentioned above. It is accepted as a baseline and represented as A in the Table 5.6. Subsequent experiments on the production system under the best product embedding model, represented as B in the table, yields several findings.

The most striking findings of this experiment is uplifts which is +5.2 in *R-Click/U* and +3.2 in *R-Trx/U* . It clearly demonstrate that the proposed model recommends better similar item list than the current system and increase purchase rate for recommended products. Furthermore, the proposed model increase the other metrics; *Trx/U*, *CR* which shows overall performance on the platforms. That point is also important since the recommendation models can worsen the several operational metrics while improve some kind of them.

In the light of all the data gathered and represented in Table 5.6, the proposed approach outperforms in the production system by recommending more appropriate products for users, and increasing the probability of purchase. In addition, two sided Z test is designed to validate A/B test results' statistical significance with 0.95 confidence interval and p-values are calculated, can be seen in Table 5.7. These results indicate that *CR*, *Trx/U*, *R-Click/U* metrics are statistically significant because their p-value is less than 0.05 and there is no statistical evidence for *R-Trx/U* metric. This can be attributed to lack of enough sample data in transaction count for statistical significance.

Table 5.6. Comparative Daily Results for Best Product Embedding Model (B) vs Currently Used System (A).

B Vs A				
	CR (%)	**Trx/U (%)**	**R-Click/U (%)**	**R-Trx/U (%)**
Day 1	-7.1	-7.4	-0.8	-14.4
Day 2	-3.0	-3.2	-0.2	-4.6
Day 3	14.1	14.5	3.3	13.1
Day 4	10.6	11.2	17.0	12.0
Day 5	0.8	1.4	10.6	4.3
Day 6	0.3	1.3	9.1	-2.7
Day 7	3.1	3.6	1.4	0.9
Day 8	-10.7	-10.4	0.8	-13.1
Day 9	-2.4	-2.0	9.5	-1.2
Day 10	7.4	7.7	8.7	3.3
Day 11	7.1	7.5	0.8	7.0
Day 12	10.1	10.9	0.3	8.6
Day 13	4.5	4.7	8.3	4.9
Day 14	2.4	4.0	5.4	11.5
Day 15	2.7	3.8	-1.6	3.6
Day 16	2.5	2.8	4.0	1.7
Day 17	12.7	13.1	5.0	19.0
Day 18	0.5	1.6	6.8	-3.9
Day 19	6.2	6.4	6.3	9.9

Table 5.7. Cumulative Results for Best Product Embedding Model (B) vs Currently Used System (A).

	B vs A			
	CR (%)	**Trx/U (%)**	**R-Click/U (%)**	**R-Trx/U (%)**
Uplift	3.0	3.5	5.2	3.2
p-value	0.0067	0.0019	0.0000	0.5009

Table 5.8. Scorer Model Performance Results.

	AUC			*Loss*		
Hidden Layers	$BS = 32$	$BS = 64$	$BS = 128$	$BS = 32$	$BS = 64$	$BS = 128$
Zero	0.5476	0.5557	0.5555	0.6705	0.6693	0.6697
64 RELU	0.7250	0.7221	0.7155	0.5746	0.5786	0.5840
128 RELU	0.7704	0.7699	0.7719	0.5416	0.5459	0.5439
128 - 16 RELU	0.8022	0.8086	0.8154	0.5071	0.4991	0.4894
128 - 32 RELU	0.8282	0.8259	0.8262	0.4735	0.4828	0.4882
128 - 64 RELU	0.8401	0.8401	0.8377	0.4653	0.4721	0.4740
128 - 64 - 16 RELU	0.8510	**0.8568**	0.8526	0.4395	**0.4361**	0.4404

6. CONCLUSION

Recent developments in online marketplaces and environmental consciousness have drawn attention to second hand platforms. These platforms facilitate customer's shopping experience, in addition, they enable users to being both selling and buying part of the process. This easiness and cross functionality of the platforms bring up a tough competition between the platforms in their disruptive environments. When it comes to preceding your competitors, recommendation systems, especially personalized recommendation, play a critical role. However, unlike the classical e-commerce platforms, the second hand platforms consist of unique items with attributes in the different range, hence, that makes the recommendation problem to difficult and it requires exponentially increasing computational powers when the user volume is taken into consideration as well.

This work address the personalized recommendation problems in second hand platforms in an efficient and scalable way. For this purpose, we have partitioned the problem into two stages; candidate generation and ranking. With the notion that user's time ordered actions with products during sessions in the platform can be accepted as a sentence or document in NLP problems. The state of the art word2vec method is employed to product representation and computing similarity between the products in the candidate generation step, and then, it is used to recommend top N similar item list together with their vector representation from the platform candidate pool, which consist of millions of product. In the ranking step, a MLP model is utilized to predict user's interaction probability for these top N products based on product and user vector representations that denote user's preference. Afterwards, these candidate top N products are sorted according to their interaction probability and recommended top K products to end users.

In order to evaluate performance of the proposed model, we conduct offline experiments on historical clickstream dataset which is gathered from a popular second hand platform and compare the model performance with a baseline classical CF model.

The results indicate that the proposed approach consistently outperforms the baseline model in both evaluation metrics which are precision@n and diversity@n. In addition, the candidate generation step is applied in the next item recommendation page in the product system and an A/B test is conducted to compare the platform's existing recommendation system using a model based approach during 19 days over 996K users. These results support the offline experiments and we observe +5.2% uplift in R-Click/U rate and +3.2% uplift in R-Click/U rate, in addition, statistically significant increase in several operational metrics. Both offline experiment and A/B test indicate that the proposed approach provides high quality product recommendation on second hand platforms and scalable architecture can be build on any clickstream dataset without additional computationally heavy feature extraction steps since the product representation step efficiently identifies product attributes even some exclusive details such as color, size and price.

As a future work, the scorer and embedding models can be tuned with different hyper-parameters and configuration setups to improve the current performance. Particularly, the proposed methodology can be executed on products which have very few interactions since we have excluded under fixed threshold in this study. In addition, another A/B test can be performed to evaluate scorer model performance in the production system or offline experiments can be conducted on a public ecommerce dataset.

REFERENCES

1. Bobadilla, J., F. Ortega, A. Hernando and A. Gutiérrez, "Recommender Systems Survey", *Knowledge Based Systems*, Vol. 46, pp. 109–132, 2013.

2. Jannach, D. and M. Jugovac, "Measuring the Business Value of Recommender Systems", *ACM Transactions on Management Information Systems (TMIS)*, Vol. 10, pp. 1 – 23, 2019.

3. Yadav, U., N. Duhan and K. Bhatia, "Dealing with Pure New User Cold-Start Problem in Recommendation System Based on Linked Open Data and Social Network Features", *Mobile Information Systems*, Vol. 2020, pp. 1–20, 06 2020.

4. Abdollahpouri, H., M. Mansoury, R. Burke and B. Mobasher, "The Unfairness of Popularity Bias in Recommendation", *ArXiv*, Vol. abs/1907.13286, 07 2019.

5. Turban, E., J. Whiteside, D. King and J. Outland, *Introduction to Electronic Commerce and Social Commerce*, Springer, 2017.

6. Wang, T., Y. Brovman and S. Madhvanath, "Personalized Embedding-Based E-Commerce Recommendations at EBay", *ArXiv*, Vol. abs/2102.06156, 2021.

7. Goldberg, Y. and O. Levy, "Word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method", *ArXiv*, Vol. abs/1402.3722, 02 2014.

8. Marco Baroni, Georgiana Dinu, G. K., "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors", *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, Vol. 1, pp. 238–247, 06 2014.

9. Sarwar, B., G. Karypis, J. Konstan and J. Riedl, "Item-based Collaborative Filter-

ing Recommendation Algorithms", *Proceedings of ACM World Wide Web Conference*, Vol. 1, p. 285–295, 08 2001.

10. Linden, G., B. Smith and J. York, "Amazon.com Recommendations: Item-to-Item Collaborative Filtering", *IEEE Internet Computing*, Vol. 7, pp. 76–80, 2003.

11. Ekstrand, M., J. Riedl and J. Konstan, "Collaborative Filtering Recommender Systems", *Foundations and Trends in Human-Computer Interaction*, Vol. 4, pp. 175–243, 01 2011.

12. Phi, V., L. Chen and Y. Hirate, "Distributed Representation-Based Recommender Systems in E-Commerce", *DEIM Forum*, 2016.

13. Mikolov, T., K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space", *ICLR*, 2013.

14. Bengio, Y., R. Ducharme and P. Vincent, "A Neural Probabilistic Language Model", *Journal of Machine Learning Research*, Vol. 3, p. 1137–1155, 03 2003.

15. Weng, L., *Learning Word Embedding*, 2017, `https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html`, accessed in November 2021.

16. Le, Q. V. and T. Mikolov, "Distributed Representations of Sentences and Documents", *ArXiv*, Vol. abs/1405.4053, 2014.

17. Su, X. and T. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques", *Advanced Artificial Intelligence*, Vol. 2009, pp. 421425:1–421425:19, 2009.

18. Koren, Y., R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems", *Computer*, Vol. 42, No. 8, pp. 30–37, 2009.

19. Hu, Y., Y. Koren and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets", *2008 Eighth IEEE International Conference on Data Mining*, pp. 263–

272, 2008.

20. Burke, R., "Hybrid Recommender Systems: Survey and Experiments", *User Modeling and User-Adapted Interaction*, Vol. 12, pp. 331–370, 2004.

21. Shani, G. and A. Gunawardana, "Evaluating Recommendation Systems", *Recommender Systems Handbook*, Springer, Boston, 2011.

22. Covington, P., J. K. Adams and E. Sargin, "Deep Neural Networks for YouTube Recommendations", *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

23. Chen, Q., H. Zhao, W. Li, P. Huang and W. Ou, "Behavior Sequence Transformer for E-Commerce Recommendation in Alibaba", *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, 2019.

24. Barkan, O. and N. Koenigstein, "ITEM2VEC: Neural Item Embedding for Collaborative Filtering", *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2016.

25. Grbovic, M., V. Radosavljevic, N. Djuric, N. L. Bhamidipati, J. Savla, V. Bhagwan and D. Sharp, "E-commerce in Your Inbox: Product Recommendations at Scale", *ArXiv*, Vol. abs/1606.07154, 2016.

26. Grbovic, M. and H. Cheng, "Real-Time Personalization Using Embeddings for Search Ranking at Airbnb", *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

27. Vasile, F., E. Smirnova and A. Conneau, "Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation", *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

28. Pfadler, A., H. Zhao, J. Wang, L. Wang, P. Huang and D. Lee, "Billion-Scale

Recommendation with Heterogeneous Side Information at Taobao", *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 1667–1676, 2020.

29. Misztal-Radecka, J., B. Indurkhya and A. Smywinski-Pohl, "Meta-User2Vec Model for Addressing the User and Item Cold-Start Problem in Recommender Systems", *User Modeling User-Adapted Interaction*, Vol. 31, pp. 261–286, 2021.

30. Özsoy, M. G., "From Word Embeddings to Item Recommendation", *ArXiv*, Vol. abs/1601.01356, 2016.

31. Brovman, Y. M., M. Jacob, N. Srinivasan, S. Neola, D. A. Galron, R. Snyder and P. Wang, "Optimizing Similar Item Recommendations in a Semi-structured Marketplace to Maximize Conversion", *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 199–202, 09 2016.

32. Jiang, S. and J. Song, "Evaluation Metrics for Personalized Recommendation Systems", *Journal of Physics: Conference Series*, Vol. 1920, p. 012109, 05 2021.

33. Rehurek, R. and P. Sojka, "Gensim–python framework for vector space modelling", *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, Vol. 3, No. 2, 2011.

34. Nguyen, P., J. Dines and J. Krasnodebski, "A Multi-Objective Learning to Re-Rank Approach to Optimize Online Marketplaces for Multiple Stakeholders", *ArXiv*, Vol. abs/1708.00651, 08 2017.

35. Wan, M., D. Wang, J. Liu, P. N. Bennett and J. McAuley, "Representing and Recommending Shopping Baskets with Complementarity, Compatibility and Loyalty", *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018.

36. Mikolov, T., I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and Their Compositionality", *Advances in Neural*

Information Processing Systems, Vol. 26, 10 2013.

37. Frederickson, B., *Implicit*, 2017, `https://github.com/benfred/implicit`, accessed in October 2021.

38. Chollet, F. *et al.*, *Keras*, 2015, `https://github.com/fchollet/keras`, accessed in October 2021.

39. Ruder, S., *Approximating the Softmax for Learning Word Embeddings*, 2016, `https://ruder.io/word-embeddings-softmax/`, accessed in December 2021.