big data and cognitive computing

Article

Deep Neural Network and Boosting Based Hybrid Quality Ranking for e-Commerce Product Search

Mourad Jbene [1], Smail Tigani [1,*], Rachid Saadane [2] and Abdellah Chehri [3]

[1] Euromed Research Center, Engineering Unit, Euro-Mediterranean University, Fes 51, Morocco; morad.jbene@eidia.ueuromed.org
[2] SIRC/LAGeS-EHTP, Electrical and Telecommunications Engineering Department, EHTP, Casablanca BP 8108, Morocco; rachid.saadane@gmail.com
[3] Applied Sciences Department, University of Quebec, Chicoutimi, QC G7H 2B1, Canada; achehri@uqac.ca
* Correspondence: s.tigani@ueuromed.org or s.tigani@outlook.com; Tel.: +212-661972900

Abstract: In the age of information overload, customers are overwhelmed with the number of products available for sale. Search engines try to overcome this issue by filtering relevant items to the users' queries. Traditional search engines rely on the exact match of terms in the query and product meta-data. Recently, deep learning-based approaches grabbed more attention by outperforming traditional methods in many circumstances. In this work, we involve the power of embeddings to solve the challenging task of optimizing product search engines in e-commerce. This work proposes an e-commerce product search engine based on a similarity metric that works on top of query and product embeddings. Two pre-trained word embedding models were tested, the first representing a category of models that generate fixed embeddings and a second representing a newer category of models that generate context-aware embeddings. Furthermore, a re-ranking step was performed by incorporating a list of quality indicators that reflects the utility of the product to the customer as inputs to well-known ranking methods. To prove the reliability of the approach, the Amazon reviews dataset was used for experimentation. The results demonstrated the effectiveness of context-aware embeddings in retrieving relevant products and the quality indicators in ranking high-quality products.

Keywords: deep learning; sentiment analysis; information retrieval; Learning to Rank; e-commerce; search engines

check for updates

Citation: Jbene, M.; Tigani, S.; Saadane, R.; Chehri, A. Deep Neural Network and Boosting Based Hybrid Quality Ranking for e-Commerce Product Search. *Big Data Cogn. Comput.* **2021**, *5*, 35. https://doi.org/10.3390/bdcc5030035

Academic Editor: Min Chen

Received: 26 April 2021
Accepted: 11 August 2021
Published: 13 August 2021

1. Introduction

In the past decade, e-commerce has changed the way people buy and sell goods. As one of the most important innovations in trading, e-commerce provides "any-time, anywhere, any-device" commerce [1]. Several studies revealed that 95% of shoppers conduct research online before making any purchase [2]. Selling online has become an essential process for small, medium, and large companies. With the development of this industry, information overload makes the task of finding relevant items more difficult. Companies such as AliExpress, eBay, Amazon are among the top e-shopping platforms today; these companies compete in developing innovative solutions to both improve their sales and profits and fulfill their clients' needs. Product search engines are among the most necessary decision support tools that help customers overcome the huge number of products on such platforms. E-commerce search is considered a particular area of information retrieval (IR), and the particularity of e-commerce search functionality comes from the fact that users are not just searching for products that match their queries, but are also seeking to find good products. Recent studies showed that the utility of a product to the customer is a multidimensional modality that is affected by many attributes; for example, the popularity, price, and durability were shown to reflect the end decision of customers in online stores [3,4]. Another interesting particularity of the e-shopping search is that users' queries are usually short, not very clear, and can be specified in multiple languages and from different cultural contexts [5], posing limitations to conventional hard text-matching approaches.

Later approaches were aware of these issues and tried to overcome them by projecting products and queries into latent embedding spaces, either by learning them from scratch using appropriate datasets or by using pre-trained ones. These methods work quite well, compared to previous word-matching approaches. However, they still have the limitation of learning embeddings of words without taking into account the context in which they appear.

To overcome this issue, a new approach is proposed. This matching model benefits from word embeddings learned from huge corpora and thus is able to efficiently capture word semantics. To compare the performance of the approach using word embeddings from the BERT large [6] model that takes into account the context of words, we tested the same approach using word embeddings learned with FastText [7], which is another embedding model that was successfully applied for many natural language processing (NLP) tasks, for instance, semantic similarity and Word Translation [8,9]. Word embeddings are used as feature representations of products and queries, and a custom similarity measure is used to extract the most relevant products for a given query.

Consumer-generated data, i.e., reviews, can provide interesting features of the listed products based on the experience of customers that have already bought the item [10]. Therefore, reviews are among the most critical factors that determine the purchase behavior of customers. To refine the search result, a recurrent neural network (RNN) model is proposed. The later model extracts sentiment scores from product reviews. These scores are then used along with a set of quality indicators as input features to rank the retrieved products in descending order of utility to the user.

The rest of this paper is structured as follows. Section 2 contains related works. In Section 3, the proposed solution is described. Experimental results and discussions are presented in Section 4. Finally, some concluding remarks and future work are given in Section 5.

2. Related Works

Three lines of research are directly related to this work: product search, distributed representation of words commonly known as embeddings, and Learning to Rank.

2.1. Product Search

Product search is a fundamental function in many online platforms. In e-commerce, in particular, product search has been studied in different ways. Conventional query-item matching methods, such as the classical probabilistic retrieval model BM25 [11] and the language modeling approach query likelihood model (QL) [12], ignore the order of word sequences and are based on exact "hard" matches of tokens rather than semantic "soft" matches. Despite their simplicity, the later methods cannot satisfy users' e-shopping search behavior effectively since there is a large vocabulary gap between the product description and user queries [13].

To alleviate the vocabulary mismatch problem in product search between user queries and product attributes, newer approaches based on semantic latent space models were suggested. For example, the paper [13] proposed a product search model that consists of a latent vector space model that maps queries and products into a hidden semantic space, then retrieves products based on their similarity with the query in the latent space. The paper [14] introduced a hierarchical embedding model, which is a state-of-the-art retrieval model for personalized product search. The latent space retrieval model projects queries, users, and items into a semantic space and conducts product retrieval according to the semantic similarity between items and the composition of query and user models. Later [15] proposed an attentive embedding model that personalizes product search based on the query characteristics and user purchase histories. The paper [16] proposed a

TranSearch model that incorporates the visual preference of users as well as their textual preference for personalized product search. Ref. [17] extracts users' positive and negative feedback about product aspects from review data and builds embedding networks that encode both items and their extracted aspects to improve conversational product search results. Ref. [18] presented QUARTS, an end-to-end neural model to enhance the ranking performance in product search by addressing the problem of search engine results that do not match search query intent.

In most of the presented works, embeddings of words are learned without taking into consideration the context that they come in. In contrast, we use the BERT [6] model that was trained on a substantial corpus of data to extract context-aware word embeddings.

2.2. Word Embeddings

Dealing with text is a challenging problem. The one-hot representation of words has been widely used as the basis of NLP and IR. The critical issue with this representation is that it does not take into account any semantic relation between words and faces the data-sparsity problem.

Recently, the word embeddings approach, represented by deep learning, has attracted extensive attention and was widely used to tackle many challenging natural language processing tasks, such as text classification, question-answering, and so on. Word embeddings are dense vector representations with dimensions usually ranging from 300 to 900, depending on the size of the vocabulary that was trained on. These distributed word representations are meant to capture lexical semantics in numerical form to handle the abstract semantic concept of words.

Among the most widely-used methods to learn word embeddings, Word2Vec [19] is an efficient neural network architecture that can train distributed word representations from a large corpus. Word2Vec implements two models: CBOW and Skip-gram. The first one's training objective is to find word representations that are useful to predict the target word by its context words, and the later model's objective is to find representations that are useful to predict the context words by the target word. Another popular approach for word representation is Glove [20], which also considers the co-occurrence of words as additional global information to Word2Vec information captured from the local context window. FastText [7] is a different word embedding model that comes with the purpose of handling out-of-vocabulary (OOV) words. FastText represents each word as a bag of character n-grams; a vector representation is associated with each character n-gram, and the words are represented as the sum of these representations.

Some words can have different meanings depending on the context they come in. One of the recent works that can generate different word embeddings based on the context of words is BERT [6], as opposed to the majority of the previous methods that are based on language models that are unidirectional, which means that every token is represented based on its left or right context. BERT was designed with the objective of pretraining deep bidirectional representations from an unlabeled text by jointly conditioning on both left and right contexts of all layers. It uses a masked language model (MLM) as a pretraining objective. The MLM objective enables the representation to fuse the left and the right context, which allows pretraining a deep bidirectional transformer. The BERT model achieves state-of-the-art performance on a broad suite of sentence-level tasks such as natural language inference, and paraphrasing, as well as on token-level tasks, including named entity recognition and question answering [6]. It is worth mentioning that the success story of BERT in particular, and transformer-based models, in general, supported the creation of new variants, for instance, RoBERTa, Albert, DistilBERT, and Q-BERT [21–24], which aim to overcome some efficiency and performance challenges of BERT [25]. Moreover, more task specific models also emerged; for instance, Sentence-BERT [26] uses a Siamese network architecture to fine-tune pre-trained BERT to derive semantically meaningful sentence embeddings.

2.3. Learning to Rank for e-Commerce Search

Learning to Rank (LTR) refers to machine learning algorithms that involve learning effective ranking functions from training data. In LTR, the model is trained in a supervised manner, using a training dataset. The dataset consists of a set of query-product pairs represented by a vector of numerical features and their corresponding score labels, which show the degree of relevance of the product to its corresponding query. In the testing phase, the learned model assigns a score for each product, then products are ranked in descending order according to their scores. Liu [27] categorized the different LTR approaches based on their training objectives into *pointwise*, *pairwise*, and *listwise*. In *pointwise*, the learned model predicts a score for each query–product pair without taking into account the preference of order to the other products. In *pairwise*, the model learns to preference between each pair of products concerning individual queries; the ranking problem is thus transformed into binary classification. Then, the *listwise* approaches tries to directly optimize a ranking-metric, which is difficult because such metrics are often non-differentiable with respect to the model parameters.

The LTR problem has been widely studied in the context of web search, and several models have been proposed in the machine learning community, including neural networks, support vector machines, and boosted decision trees based methods. RankNet [28] is one popular example of a neural LTR model that was an industry choice for many years. Later listwise models, such as LambdaRank [29] and LambdaMART [30], have gained more attention, as they attempt to optimize loss functions that are directly related to IR evaluation measures, such as NDCG. Popular input features for training LTR models can be categorized into three categories [31]: query-independent (e.g., length of the title), query-dependent (e.g., TF, BM25 scores), and query-level features (e.g., number of words in query). Most of the hand-crafted features used for ranking web pages are statistical text features, while such features can have a remarkable effect in web pages ranking for product ranking; other signals that reflect the utility of the product for the user rather than just the text relevance with the query can be more important.

As compared to web ranking, there are not many works about e-commerce product ranking in the literature. Ref. [32] used an ensemble tree model to predict user clicks by hand-crafting a variety of ranking features for each item from text data and user behavior logs. In [33] the authors conducted a synthetic study of applying LTR methods to e-commerce search, revealing some of its unique challenges, such as the difficulty in defining the relevance judgment labels. Additionally, Ref. [34] proposed a LETORIF model with a ranking loss function that optimizes product search engines by maximizing the revenue. They used two types of features: text relevance features extracted from item attributes, and revenue features with the price as the main component. Recently, Ref. [35] proposed a hierarchical deep neural network for ranking products based on online reviews.

In most of the previous works, authors used features similar to those used for ranking web pages. In our work, we exclude the text-match features from the LTR features, as our matching model based on semantic embeddings could handle the text match problem between the query and the product fairly efficiently. Instead, in the LTR step, we use product quality indicators based on product attributes and user feedback.

3. Methodology

This section gives a detailed explanation of the proposed model for product search, then presents a set of quality indicators that serve as input to the ranking models.

3.1. Preprocessing Step

As a preprocessing step, the text is tokenized, and stop-words are removed as well as special characters. For FastText [7], the word tokenizer from the NLTK [36] library was used, and WordPiece tokenization [37] was used for BERT [6], as it was the same tokenization method used in the original BERT [6] paper.

3.2. Product Search Using Similarity Measure

In the proposed product search strategy, products are filtered based on their textual and semantic similarity to the query.

The title of the product is a rich element that characterizes a product; merchants tend to put most information, such as brand name, color, and size, in the title. The title of the product is also one of the strongest signals that determine the relevance of an item listing to an e-commerce query [38]. The approach of [15,39] is followed in using product titles to represent products.

First, each product is represented as a vector of the title token embeddings. The same applies for queries, as each query is represented as a vector of its token embeddings. Embeddings for both queries and product titles were extracted using BERT for the first experiment and using FastText for the second experiment. For BERT, following the authors' recommendation, word embeddings were extracted as the average of the last four layers of the BERT large model already trained on a large corpus of English Wikipedia (2500 M words).

Previous studies proposed several approaches to combine word embeddings to form a sentence embedding as mentioned by [14]. The simplest one is averaging, and an extension of this is the non-linear projection layer over the average of word embeddings. Another more complex method is to use an (RNN) with word embeddings as input and take the final network state as a latent representation of the whole sentence. The first method, i.e., taking the mean of word embeddings, does not work well as the sentence (the title of the product, or query, in our case) becomes longer, while the latter two methods require finding parameters by training on a separate set. In our work, and due to the specific nature of e-commerce search in which some words may be more important than others, for instance, the brand or the color of the product, we propose another approach that does not combine word embeddings but compares them in the following manner. Given a query, and a product title vector representations $Q = [\vec{e}_{q_1}, \ldots, \vec{e}_{q_n}]$, $P = [\vec{e}_{p_1}, \ldots, \vec{e}_{p_m}]$, where $\vec{e}_q$ and $\vec{e}_p$ are the embedding vectors of size 768 for BERT and 300 for FastText, n and m are number of tokens in the query and title, respectively. The relevance score for a product P to a query Q is computed using a similarity function presented in the following formulas:

$$S(Q,P) = \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{m}\max_{j}(s(\vec{e}_{q_i}, \vec{e}_{p_j})) \tag{1}$$

where,

$$s(\vec{e}_q, \vec{e}_p) = \frac{1+\cos\left(\vec{e}_q, \vec{e}_p\right)}{2}; \ \cos\left(\vec{e}_q, \vec{e}_p\right) = \frac{\vec{e}_q^{\,T}.\vec{e}_p}{\|\vec{e}_q\|_2\|\vec{e}_p\|_2} \tag{2}$$

For each query token embedding $\vec{e}_{q_j}$, a similarity measure is computed to all title embeddings tokens, using the s function, which is a simple modification to the cos similarity metric by adding 1 and dividing by 2, so that the similarity between the two vectors resides in the interval $[0,1]$. Then, the max operator is applied to capture the most similar tokens. The summation of all the query tokens is then divided by n to give the final relevance score.

3.3. Product Ranking Using Quality Indicators

In this step, we present a set of indicators that serve as the basis for ranking the candidate products selected previously, using the semantic relevance score to each query. The following indicators are inspired by the remarks of [40] that insist on the importance of product metadata and reviews in the users' decision.

3.3.1. Reviews Sentiment

In product search, simply returning something relevant to the user's submitted query may not lead to the purchasing behavior [41]. For example, a returned relevant product that has a bad reputation is far from being purchased by the user. Therefore, a sentiment analysis model is proposed in Figure 1. The model takes as input the embedding representations of all review tokens and passes them through a bi-direction LSTM layer [42]. The latter is composed of two LSTM layers: the first processes the input sequence as is, and the second processes a time-reversed copy of the same input sequence. This provides additional context to the network and increases its accuracy. The output of the previous layer is then averaged in a global average pooling layer before the last dense layer that has a single neuron with a sigmoid activation function $1/(1 + \exp(-x))$ to output a sentiment probability score for the input review. The sentiment score is estimated by the ratings given by users after rescaling, as it is observed that positive reviews tend to have higher user ratings and negative reviews tend to have a lower rating.

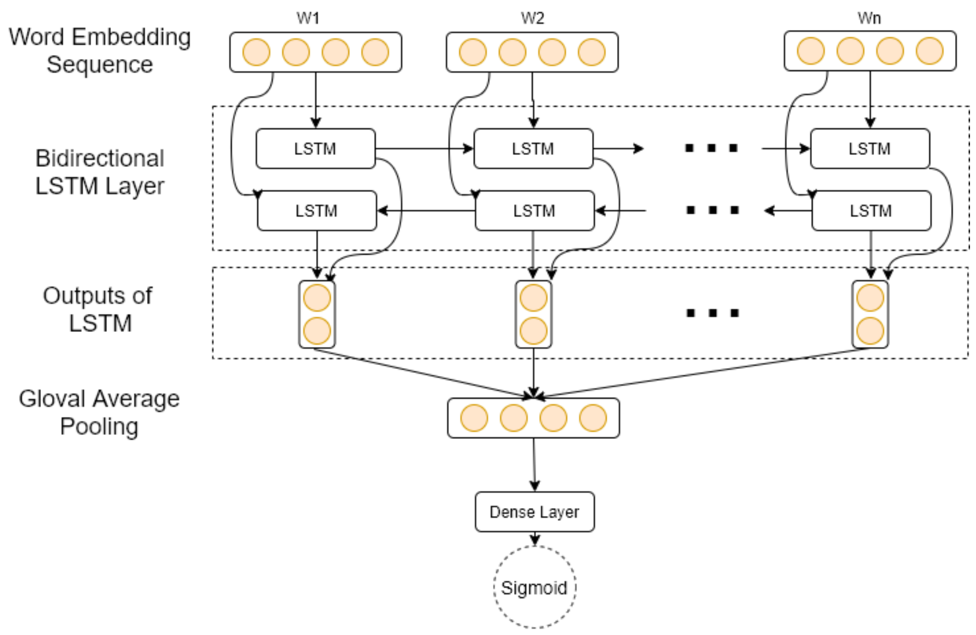

Figure 1. RNN model architecture for sentiment analysis.

3.3.2. Popularity

Popularity is usually calculated based on the number of sales of an item, compared to other items; this measure is used for Amazon.com, for example, to determine the best-seller items in each category. Because products widely purchased have greater probability of being purchased again, we calculate the number of sales of an item as the number of reviews a product has since reviews are written only by customers after purchasing an item. We obtain the popularity indicator by dividing the number of sales of a product by the maximum number of sales for the products in the corresponding query group, generated previously using the semantic match model.

3.3.3. Availability of Information

The presence of product attributes (title, description, image, price, brand, etc.) can be considered a good indicator of the quality of a product. For regular users, a product without a description or image, for example, will obtain very little attention and the customer will have a bad impression of both the item and the seller. We denote by I a one-hot vector, indicating the presence of an attribute by 1, and otherwise by 0. In addition to I, we define I_p, information for a product I as follows:

$$I_p = \frac{\sum_{x \in I}(1 - \delta_{x,0})}{|I|} \tag{3}$$

where I_p will obtain a value near 1 for products with complete list of attributes, and near 0 for products with an incomplete list. Some attributes may have a higher value for the customer. Therefore, each component of I is associated a weight value β_i.

3.3.4. Product Price

The price of the product can strongly affect the purchase decision of customers. Given C_q, the set of products selected by the semantic search model for a query q, we associate for each product $p \in C_q$ a normalized version of its price, taking $pi = \frac{price(p)}{AP(q)}$ where $AP(q) = \frac{\sum_{p \in C_q} price(p)}{|C_q|}$ as the average price of candidate products associated with query q.

3.3.5. LTR Model

The previous list of features is complemented with complementary information about the product, namely, the overall rating, brand, number of reviews.

We consider 100 products for each query retrieved, using the product search method described in Section 3.2. For each query–product pair (q, p), we assign relevance ratings. We consider the (implicit feedback) product sales rank as the ground truth, as was the case in the work of [35]. As products with higher sales have lower ranks, a transformation of the sales ranks of products is used as the ground truth relevance ratings as follows:

$$r(q,p) = \left\lfloor 4\alpha . \left| \frac{\sigma(q,p)}{\max_{p \in P_q} \sigma(q,p)} \right| \right\rfloor \tag{4}$$

where $r(q,p)$ is a discrete value in the scale $(0-4)$, $\sigma(q,p)$ is the sales rank of product $p \in P_q$ associated with query q. α is a parameter that controls the ratio of irrelevant labels. For each query, products with the highest page rank received a label of 4, while products with a small page rank received 0.

We train and compare three state-of-the-art LTR methods to test the ranking performance of the list of features; an overview of each method is described in Section 3.3.6.

3.3.6. Baseline Methods for Product Ranking

We conduct a comparison study with the following baseline methods:

- LambdaRank [29] uses a neural network to minimize the pairwise loss function, which is similar to RankNet [28]. LambdaRank simply takes the RankNet gradients of the pairwise loss function and scales them by the change in NDCG performance measure found by swapping each pair of documents.
- AdaRank [43] is a representative pairwise model. It focuses more on the difficult queries and aims to directly optimize the performance measure NDCG based on a boosting approach.
- LambdaMart [30] is a tree boosting algorithm that extends multiple additive regression trees (MART) by introducing a weighting term for each pair of data, as to how LambdaRank extends RankNet with the listwise measure.

4. Experiments

This section tackles the experimental settings by presenting the dataset, the query formulation method that we adopt to test our model, and the baseline methods for product ranking. Last, we present the evaluation metrics and discuss our experimental results.

4.1. Dataset

Traditionally, the datasets used for search problems are based on search log files, and the relevancy annotation is obtained via crowdsourcing.

With the lack of publicly available datasets of this kind for e-commerce search, we adopt *Amazon Review Data* [44] for experiments. This famous dataset has widely been used in previous studies [45–48].

The dataset includes millions of products with rich meta-data as well as user reviews. Products are divided into 29 categories, and each category contains a hierarchy of sub-categories. The original dataset is too large. For our experiment, we used the five-core version, where each user or item has at least five interactions. More specifically, we used four categories, which are *electronics*, *office products*, *toys and games*, and *appliances*.

4.1.1. Query Extraction

As reported in [49], a typical scenario of a user searching a product is to use a producer's name, a brand, or a set of terms that describe the category of the product as the query in retrieval. Based on this observation and following the paradigm of references [13,14,41], we extracted the search queries for each item with three steps. First, we extract category information for each item from product meta-data. Then, we concatenate the terms from a single hierarchy of categories to form a topic string. At last, stop words and duplicate words are removed from the topic string, and we use it as a query for the corresponding item. In this way, only the items that belong to the query are considered relevant to that query. Table 1 showcases some examples of queries extracted using this approach.

Table 1. Example queries extracted from Amazon product data.

Electronics:	Toys and ames:
- electronics home audio speakers	- toys games hand puppets
- electronics camera photo accessories	- toys game room mini table games
Office Products:	Appliances:
- products office school supplies paper	- appliances dryer parts accessories
- products office school supplies education crafts	- laundry appliances dryers washers

4.1.2. Evaluation Metrics

To evaluate the performance of the product retrieval model, we used the following retrieval metrics:

- Precision (PR): the fraction of the products retrieved that are relevant to the query.

$$PR = \frac{|Rel_p \cap Ret_p|}{|Ret_p|} \tag{5}$$

- Recall (RE): the fraction of the products relevant to the query that are successfully retrieved.

$$RE = \frac{|Rel_p \cap Ret_p|}{|Rel_p|} \tag{6}$$

- F-score (FS): the weighted harmonic mean of the precision and recall.

$$FS = \frac{2 \times PR \times RE}{PR + RE} \tag{7}$$

where Ret_p is the set of retrieved products and Rel_p is the set of relevant products.

For the ranking model, we report standard information retrieval ranking metrics:

- **Normalized discounted cumulative gain (NDCG@k)**: assesses the overall order of the ranked elements at truncation level k with a much higher emphasis on the top-ranked elements. NDCG for a query q is defined as follows:

$$NDCG@k_q = \frac{DCG@k_q}{\max DCG@k_q} \text{ ,} and \;\; DCG@k_q = \sum_{i=1}^{k} \frac{2^{l_i} - 1}{\log(1+i)} \tag{8}$$

  where $\max DCG@k_q$ is the ideal value of $DCG@k_q$, and l_i is the label of the i-th listed product.

- **Expected reciprocal rank (ERR@k)** [50]: a cascade-based metric that is commonly used for graded relevance.

$$ERR@k = \sum_{r=1}^{n} \frac{1}{r} \prod_{i=1}^{r-1} (1 - R_i) R_r \tag{9}$$

where n is the number of items in the ranked list; r is the position of the document; and R is a mapping from relevance degree to relevance probability.

4.2. Results and Discussions

The performance of the product search model is reported in Table 2 in terms of precision, recall, and the F1-score at a cut-off $K = 100$ results. The model was tested on four datasets, using two types of word embedding representations: FastText and BERT.

Table 2. Comparison of product search model using FastText and BERT embeddings on the Amazon product search datasets. The best performance is highlighted in bold.

Dataset	Embeddings	Precision@K	Recall@K	F1-Score@K
Appliances	Bert	**0.27**	**0.25**	**0.26**
	FastText	0.22	0.12	0.16
Electronics	Bert	**0.13**	**0.22**	**0.15**
	FastText	0.12	0.12	0.12
Office Products	Bert	**0.12**	**0.22**	**0.16**
	FastText	0.13	0.09	0.11
Toys and Games	Bert	**0.18**	**0.17**	**0.17**
	FastText	0.16	0.09	0.12

It can be seen from Table 2 that the retrieval performance of our product search model differs from a dataset to another. The performance is best in terms of Precision@K for the *appliances* dataset with 0.27, and worst for *office products* with 0.13, using the model with BERT embeddings. In terms of Recall@K and F1-Score@K, the *appliances* dataset obtained the best results using both the model with BERT and with FastText embeddings. On the other hand, *toys and games* achieved the worst Recall@K, and *office products* the worst F1-Score@K.

Comparing the results of the model using the different kinds of embeddings, the version with BERT embeddings generates results higher than FastText ones for all the datasets, which can be explained by the difference in dataset size on which the two models were pre-trained on, and most importantly, the capability of BERT in capturing and generating context-dependent word embeddings.

To analyze the performance of the sentiment analysis model in detecting the sentiment in reviews, we divided the reviews into two sets 80% for training the model and 20% for validation. The model was trained for 80 epochs, with a batch size of 512 and the *Adam* optimizer with a learning rate of 0.01, and the optimized AUC (Area Under the Curve) metric, which is commonly used to measure the quality of classification models. These parameter settings were obtained based on a grid-search approach for different values of the parameters. Following, we present plots of the loss and ROC (Receiver Operating Characteristic) curve for both the training and validation sets in Figures 2 and 3.

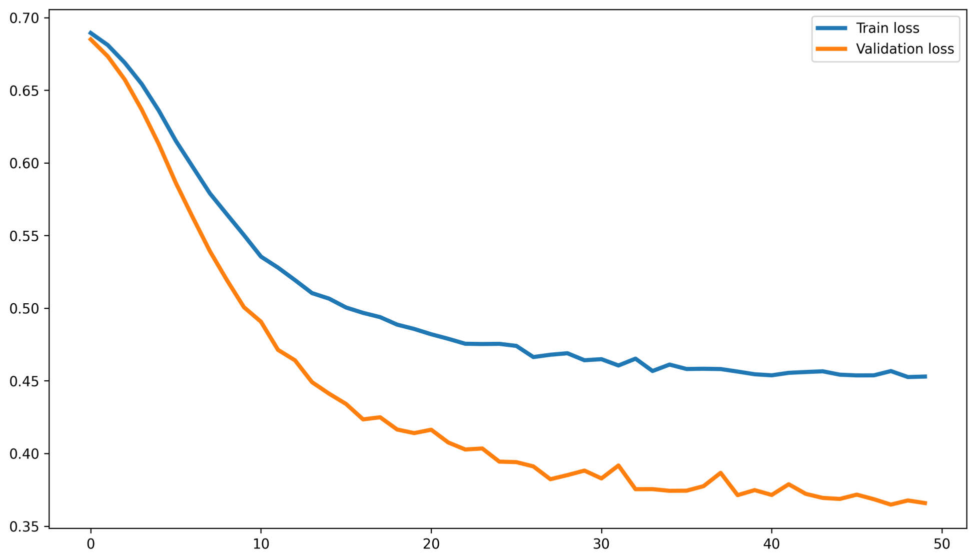

Figure 2. Sentiment analysis model performance—training and validation loss plot.

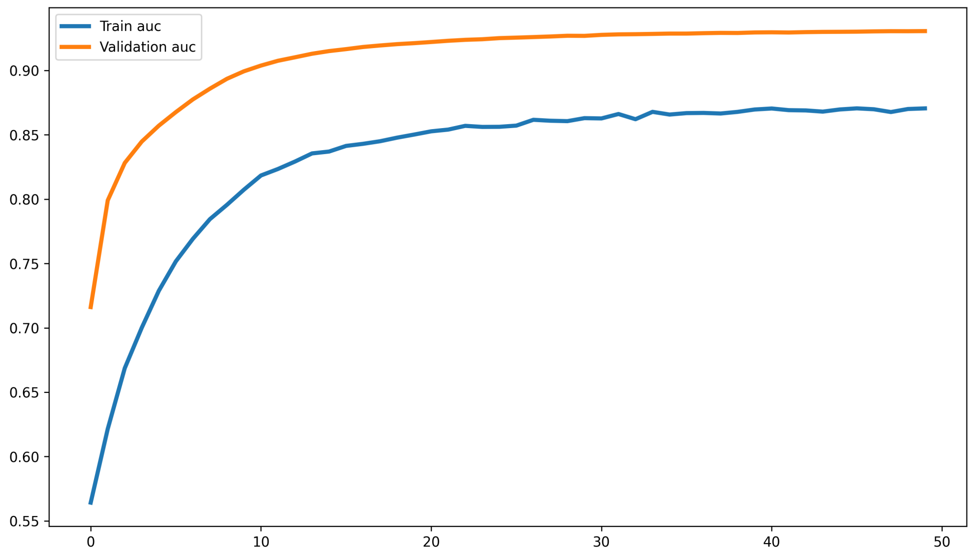

Figure 3. Sentiment Analysis Model Performance—Training and Validation AUC Plot.

In Figures 2 and 3, the loss and AUC metrics are depicted for both training and validation sets during the training phase of the sentiment analysis model. The loss decreases significantly during the first 20 epochs, while the accuracy increases remarkably during the first 10 epochs. The best model performance is achieved and saved at around 50 epochs, using a callback function. Later, the trained model is used to predict the sentiment polarity scores for each customer review. Then, averaging the sentiment scores for the reviews, we obtain the sentiment score of each product.

Table 3 presents a summary of results for applying different state-of-the-art LTR algorithms on four Amazon datasets. Comparing the models, LambdaMART achieves the best validation performance in terms of both NDCG@10 and ERR@10 for all the datasets, followed by AdaRank and RankNet. This observation is consistent with benchmark studies of [33]. From the datasets side, the *electronics* dataset achieves the highest validation NDCG@10 for LambdaRank and AdaRank algorithms, while *office products* obtains the highest ERR@10 for LambdaMART. On the other hand, *toys and games* obtains the lower performance for AdaRank and LambdaMART, while the worst ERR@10 and NDCG@10 for LambdaRank are recorded for the textitoffice products dataset.

Table 3. Performance comparisons of three LTR methods on each Amazon dataset.

Methods Categories	**LambdaRank**		**AdaRank**		**LambdaMART**	
	ERR@10	**NDCG@10**	**ERR@10**	**NDCG@10**	**ERR@10**	**NDCG@10**
Appliances	0.476	0.480	0.565	0.560	0.612	0.600
Electronics	**0.487**	**0.490**	**0.571**	**0.570**	0.627	**0.650**
Office Products	0.407	0.410	0.479	0.482	**0.631**	0.644
Toys and Games	0.477	0.489	0.508	0.492	0.603	0.554

5. Conclusions and Future Work

This paper attempted to solve the problem of product search. The process was broken down into two parts: (1) selecting candidate products for each user query, using a similarity measure function on top of the product and query embeddings; and (2) ranking candidate products, using different state-of-the-art LTR models, with quality indicators as input. A deep neural network model was used to extract sentiment scores from reviews, while the other quality indicators were calculated using custom formulas. The experiments show that the similarity function was able to retrieve a good subset of relevant items to the queries despite the significant similarity between products of the same dataset and the generality of the extracted queries. Furthermore, the quality features show good performance in predicting the sales rank of the product, and LambdaMART was the best performing LTR model. An important issue that could affect the performance of our approach is fake reviews of product. Therefore, a promising path for feature work will be to incorporate other user-related features that could reveal the credibility of user interactions, which should enhance the ranking phase and potentially increase user satisfaction.

Author Contributions: Conceptualization, S.T. and M.J.; methodology, M.J. and S.T.; software, M.J.; validation, S.T., M.J., R.S., A.C.; formal analysis, S.T., M.J.; investigation, M.J.; resources, M.J.; data curation, M.J.; writing—original draft preparation, M.J.; writing—review and editing, S.T., R.S., M.J., A.C.; visualization, M.J.; supervision, S.T.; project administration, S.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Euro-Med University of Fez.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: We would like to thank the anonymous referees for their valuable comments and helpful suggestions. Special thanks to those who improved the language quality and made this paper more readable.

Conflicts of Interest: We declare that we have no conflict of interest and no financial or other interest with any entity to disclose.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of open access journals
TLA	Three letter acronym
LD	Linear dichroism

References

1. Kaabi, S.; Jallouli, R. Overview of E-commerce Technologies, Data Analysis Capabilities and Marketing Knowledge. In *Digital Economy, Emerging Technologies and Business Innovation*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 183–193.
2. Turban, E.; King, D.; Lee, J.K.; Liang, T.P.; Turban, D.C. E-Commerce: Mechanisms, Platforms, and Tools. In *Electronic Commerce: A Managerial and Social Networks Perspective*; Springer Texts in Business and Economics; Springer International Publishing: Cham, Switzerland, 2015; pp. 51–99. [CrossRef]
3. Moraes, F.; Yang, J.; Zhang, R.; Murdock, V. The role of attributes in product quality comparisons. In Proceedings of the 2020 Conference on Human Information Interaction and Retrieval, Vancouver, BC, Canada, 14–18 March 2020; pp. 253–262. [CrossRef]
4. Carmel, D.; Haramaty, E.; Lazerson, A.; Lewin-Eytan, L.; Maarek, Y. Why do people buy seemingly irrelevant items in voice product search? On the relation between product relevance and customer satisfaction in ecommerce. In Proceedings of the 13th International Conference on Web Search and Data Mining, Houston, TX, USA, 3–7 February 2020; pp. 79–87. [CrossRef]
5. Ahuja, A.; Rao, N.; Katariya, S.; Subbian, K.; Reddy, C.K. Language-agnostic representation learning for product search on e-commerce platforms. In Proceedings of the 13th International Conference on Web Search and Data Mining, Houston, TX, USA, 3–7 February 2020; pp. 7–15. [CrossRef]
6. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186. [CrossRef]
7. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [CrossRef]
8. Nguyen, H.T.; Duong, P.H.; Cambria, E. Learning short-text semantic similarity with word embeddings and external knowledge sources. *Knowl. Based Syst.* **2019**, *182*, 104842. [CrossRef]
9. Lample, G.; Conneau, A.; Ranzato, M.; Denoyer, L.; Jégou, H. Word translation without parallel data. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
10. Fan, Z.P.; Li, G.M.; Liu, Y. Processes and methods of information fusion for ranking products based on online reviews: An overview. *Inf. Fusion* **2020**, *60*, 87–97. [CrossRef]
11. Robertson, S.E.; Walker, S.; Jones, S.; Hancock-Beaulieu, M.; Gatford, M. *Okapi at TREC-3*; 1994. Available online: https://www.researchgate.net/publication/221037764_Okapi_at_TREC-3 (accessed on 12 August 2021).
12. Ponte, J.M.; Croft, W.B. A Language Modeling Approach to Information Retrieval. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, 24–28 August 1998. [CrossRef]
13. Van Gysel, C.; de Rijke, M.; Kanoulas, E. Learning Latent Vector Spaces for Product Search. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, Indianapolis, IN, USA, 24–28 October 2016; pp. 165–174. [CrossRef]
14. Ai, Q.; Zhang, Y.; Bi, K.; Chen, X.; Croft, W.B. Learning a Hierarchical Embedding Model for Personalized Product Search. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 645–654. [CrossRef]
15. Ai, Q.; Hill, D.N.; Vishwanathan, S.V.N.; Croft, W.B. A Zero Attention Model for Personalized Product Search. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 379–388. [CrossRef]
16. Guo, Y.; Cheng, Z.; Nie, L.; Xu, X.S.; Kankanhalli, M. Multi-modal Preference Modeling for Product Search. In Proceedings of the 26th ACM International Conference on Multimedia, Seoul, Korea, 22–26 October 2018; pp. 1865–1873. [CrossRef]
17. Bi, K.; Ai, Q.; Zhang, Y.; Croft, W.B. Conversational Product Search Based on Negative Feedback. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 359–368. [CrossRef]
18. Nguyen, T.; Rao, N.; Subbian, K. Learning Robust Models for e-Commerce Product Search. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 6861–6869. [CrossRef]
19. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; Volume 2, pp. 3111–3119.
20. Pennington, J.; Socher, R.; Manning, C. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543. [CrossRef]
21. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
22. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv* **2020**, arXiv:1909.11942.
23. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.

24. Shen, S.; Dong, Z.; Ye, J.; Ma, L.; Yao, Z.; Gholami, A.; Mahoney, M.W.; Keutzer, K. Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020.
25. Rogers, A.; Kovaleva, O.; Rumshisky, A. A Primer in BERTology: What We Know About How BERT Works. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 842–866. [CrossRef]
26. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv* **2019**, arXiv:1908.10084.
27. Liu, T.Y. Learning to Rank for Information Retrieval. *Found. Trends Inf. Retr.* **2009**, *3*, 225–331. [CrossRef]
28. Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; Hullender, G. Learning to rank using gradient descent. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 89–96. [CrossRef]
29. Burges, C.J.C.; Ragno, R.; Le, Q.V. Learning to rank with nonsmooth cost functions. In Proceedings of the 19th International Conference on Neural Information Processing Systems, Doha, QA, Canada, 12–15 November 2006; pp. 193–200.
30. Burges, C.J. *From RankNet to LambdaRank to LambdaMART: An Overview*; Technical Report MSR-TR-2010-82. Available online: https://www.microsoft.com/en-us/research/uploads/prod/2016/02/MSR-TR-2010-82.pdf (accessed on 12 August 2021).
31. Mitra, B.; Craswell, N. Neural Models for Information Retrieval. *arXiv* **2017**, arXiv:1705.01509.
32. Wu, C.; Yan, M.; Si, L. Ensemble Methods for Personalized E-Commerce Search Challenge at CIKM Cup 2016. *arXiv* **2017**, arXiv:1708.04479.
33. Karmaker Santu, S.K.; Sondhi, P.; Zhai, C. On Application of Learning to Rank for E-Commerce Search. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 475–484. [CrossRef]
34. Wu, L.; Hu, D.; Hong, L.; Liu, H. Turning Clicks into Purchases: Revenue Optimization for Product Search in E-Commerce. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 365–374. [CrossRef]
35. Lee, H.C.; Rim, H.C.; Lee, D.G. Learning to rank products based on online product reviews using a hierarchical deep neural network. *Electron. Commer. Res. Appl.* **2019**, *36*, 100874. [CrossRef]
36. Bird, S.; Loper, E. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*; Association for Computational Linguistics: Barcelona, Spain, 2004; pp. 214–217.
37. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv* **2016**, arXiv:1609.08144.
38. Bell, A.; Senthil Kumar, P.; Miranda, D. The Title Says It All: A Title Term Weighting Strategy For eCommerce Ranking. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 2233–2241. [CrossRef]
39. Bi, K.; Teo, C.H.; Dattatreya, Y.; Mohan, V.; Croft, W.B. A Study of Context Dependencies in Multi-page Product Search. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 2333–2336. [CrossRef]
40. Lawani, A.; Reed, M.R.; Mark, T.; Zheng, Y. Reviews and price on online platforms: Evidence from sentiment analysis of Airbnb reviews in Boston. *Reg. Sci. Urban Econ.* **2019**, *75*, 22–34. [CrossRef]
41. Guo, Y.; Cheng, Z.; Nie, L.; Wang, Y.; Ma, J.; Kankanhalli, M. Attentive Long Short-Term Preference Modeling for Personalized Product Search. *ACM Trans. Inf. Syst. (TOIS)* **2019**, *37*, 19:1–19:27. [CrossRef]
42. Schuster, M.; Paliwal, K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]
43. Xu, J.; Li, H. AdaRank: A boosting algorithm for information retrieval. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, 23–27 July 2007; pp. 391–398. [CrossRef]
44. Ni, J.; Li, J.; McAuley, J. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 188–197. [CrossRef]
45. Zheng, L.; Noroozi, V.; Yu, P.S. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, Cambridge, UK, 6–10 February 2017; pp. 425–434. [CrossRef]
46. Ling, G.; Lyu, M.R.; King, I. Ratings meet reviews, a combined approach to recommend. In Proceedings of the 8th ACM Conference on Recommender Systems, Foster City, Silicon Valley, CA, USA, 6–10 October 2014; pp. 105–112. [CrossRef]
47. Seo, S.; Huang, J.; Yang, H.; Liu, Y. Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 297–305. [CrossRef]
48. Catherine, R.; Cohen, W. TransNets: Learning to Transform for Recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 288–296. [CrossRef]
49. Rowley, J. Product search in e-shopping:a review and research propositions. *J. Consum. Mark.* **2000**, *17*, 20–35. [CrossRef]
50. Chapelle, O.; Metlzer, D.; Zhang, Y.; Grinspan, P. Expected reciprocal rank for graded relevance. In Proceedings of the 18th ACM Conference on Information and Knowledge Management, Hong Kong, China, 2–6 November 2009; pp. 621–630. [CrossRef]