Evaluating a Session-based Recommender System using Prod2vec in a Commercial Application

Hasan Tercan[1,*], Christian Bitter[1,*], Todd Bodnar[2], Philipp Meisen[2] and Tobias Meisen[1]
[1]*Chair for Technologies and Management of Digital Transformation, University of Wuppertal, Wuppertal, Germany*
[2]*Breinify Inc., San Francisco, U.S.A.*

Keywords: Recommender System, Deep Learning, Neural Network, Embedding, Prod2vec, Word2vec.

Abstract: Recommender systems are a central component of many online stores and product websites. An essential functionality of them is to show users new products that they do not yet know they want to buy. Since the users of the website are often unknown to the system, a product recommendation must be made using the current activities within a browser session. In this paper we address this issue in a deep learning-based product-to-product recommendation problem for a commercial website with millions of user interactions. Our proposed approach is based on a prod2vec method for product embeddings, thus recommending those products that often occur together with the target product. Following the idea of word2vec methods from the NLP domain, we train an artificial neural network on user activity data extracted from historical browser sessions. As part of several real A/B tests on the website, we prove that our approach delivers successful product recommendations and outperforms the current system in use. In addition, the results show that the performance can be significantly improved by an appropriate selection of the training data and the time range of historical user interactions.

1 INTRODUCTION

Recommender systems are a central factor of success in a range of different online applications such as e-commerce, video/music streaming and social media by making it easier for users to find new items matching their personal preferences and thus boosting their engagement with the application. These systems complement classic search engines by recommending interesting items which the user may not even know to search for (Ricci et al., 2015). The goal of the recommendation is, depending on the intended use, to show items that match the user's preferences (user-to-item recommendation) or match the items with which the user is currently interacting (item-to-item recommendation). The systems are based on algorithms which evaluate past interactions of users with items, such as product purchases or web page views, and identify relationships between them. However, this is a non-trivial task, since it requires the handling of large amounts of interaction data with complex and dynamic, but also sparse relationships.

One major challenge when considering websites with openly accessible content is the presence of users which are unknown to the system. Although session cookies can be used to store information about online activities of individual browser sessions, it is hardly possible to uniquely assign activities to individual users and thus to analyze user/item relationships.

The current advances in deep learning research in various application areas show a great potential to master these challenges. Based on historical user interaction data, deep learning models can learn patterns that are essential for successful recommendations (Zhang et al., 2019). The question arises whether meaningful recommendations based on the single consideration of items within browser sessions can be delivered.

In this paper, we address this question in a real product-to-product recommendation problem for a commercial website with millions of user activities per week. The website is used by the provider to market their products and to bind new customers through campaigns. The portfolio includes approximately a thousand products of different types, with each product being described by detailed information on a dedicated web page. In addition, each page contains recommendations for other products that might be in-

*These authors contributed equally to the work.

Tercan, H., Bitter, C., Bodnar, T., Meisen, P. and Meisen, T.
Evaluating a Session-based Recommender System using Prod2vec in a Commercial Application.
DOI: 10.5220/0010400706100617
In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) - Volume 1*, pages 610-617
ISBN: 978-989-758-509-8

teresting for a user. The overarching task for this use case is the development of a recommendation engine which maximizes user engagement with the website by providing relevant product recommendations based on the currently viewed product.

The data basis consists of historical user activities which are logged and stored within browser sessions, enabling the training of a deep learning based recommendation system that learns from the data which products are often viewed together in sessions. For this purpose, we adapt an established concept from natural language processing, namely word2vec and word embeddings, to a prod2vec approach based on low-dimensional product embeddings.

For learning such embeddings, an artificial neural network is trained by supervised learning on historical activities. Thereby the network learns from the frequency distributions of pairs of products in the sessions. The embedding itself is learned in the first layer of the network. The idea of the embedding is that two products, which are often viewed together with other products in a browser session, are similar to each other in the embedding space as well. Figure 1 illustrates a trained embedding for the given use case in a two-dimensional feature space. It shows that the learned representations place similar products that are of the same category closer to one another in the embedding space as well. There are two main questions that we address:

1. *Which approach based on the trained neural network is suitable for product-to-product recommendation?*
   The network as well as the product embedding provide different possibilities of use for a recommender. While the embedding extracted from the trained network finds products that share the same context (i.e. are viewed with the same products) and thus can be used for a nearest neighbor approach, the network itself (i.e. in its entirety) finds the probability that two products occur together in a session. The question arises as to which variant, a nearest neighbor approach vs. the end-to-end network approach, is best suited for the given use case.
2. *What is the optimal time range of historical data used for the model training?*
   The user behavior on the website changes over time. Therefore the question arises on which period of user activities the model should be trained in the best possible way. In the present use case, time spans of several weeks up to two and a half years are possible.

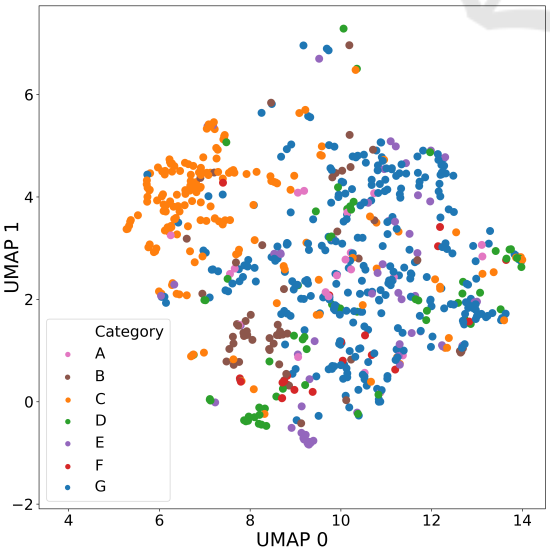

Figure 1: Trained embeddings of 1000 products, transformed into a two-dimensional space using UMAP (McInnes et al., 2018). The products are coloured according to their categories, which are anonymized in this paper.

To evaluate these questions we train and test different approaches based on prod2vec on historical data. Nevertheless, testing the models on historical data provides only limited information about their performance - the data only reflects what the user viewed given the recommendations provided by the system deployed at that time, but not the optimal products whose recommendation would have led to the maximum engagement of the user. To address this bias in the historical training data, we perform several expensive A/B tests on a live system. As part of the tests, we compare the approaches with a traditional recommendation method based on similarity calculations. It thereby becomes clear that results from real A/B tests can be quite different from the results on historical data. In addition, by performing the tests several times and successively improving the models we get answers to the above questions and obtain insights into the use of prod2vec approaches in this real-world use case.

2 RELATED WORK

2.1 Recommender Systems

Modern algorithms for recommender systems can basically be divided into content based methods, collaborative filtering methods, or hybrids of them. While content based methods calculate the similarity of items based on their properties, collaborative filtering methods consider similarities of user activities. A

very popular class of collaborative filtering methods is matrix factorization (Koren et al., 2009; Zhou et al., 2008). These methods transform a sparse user-item interaction matrix into low-dimensional latent factors which are used to approximate the preferences of a user for new articles. Other methods compute similarities between item pairs and make item-based recommendations based on the nearest neighbors (Shen and Jiamthapthaksin, 2016). In their implementation, (Shen and Jiamthapthaksin, 2016) use an efficient all-pair similarity computation method (DIMSUM) based on MapReduce that is well suited for real-world large-scale data sets (Zadeh and Carlsson, 2013; Zadeh and Goel, 2013).

2.2 Deep Learning-based Recommender Systems

Deep learning methods allow the learning of complex relationships and sequential dependencies from large amounts of data. For this reason, they are also increasingly used for recommendation systems (Zhang et al., 2019). The neural networks often extend traditional methods to a larger recommender framework, enabling both to learn latent features for collaborative filtering and representations of items (Zhang et al., 2016; Li and She, 2017; Wu et al., 2016b). (Wang et al., 2015), for example, combine a probabilistic matrix factorization method with a stacked denoising autoencoder to capture sparse relationships and also incorporate item information. Using a deep neural network on sparse interactions may cause an overgeneralization to new interactions. (Cheng et al., 2016) tackle this problem by combining a wide linear model well suited for memorizing sparse feature interactions with a deep neural network trained on low-dimensional feature embeddings that can generalize to new interactions.

Deep learning architectures are also well suited to capture sequential patterns. One common approach is to train a recurrent neural network to predict the next item in a sequence of user interactions in a session (Wu et al., 2016a; Hidasi et al., 2016; Tamhane et al., 2017; Gui and Xu, 2018) or to model long-term temporal dynamics of user behavior (Wu et al., 2017; Chen and Lin, 2019). In contrast to that, (Tang and Wang, 2018) capture short-term sequential patterns by training a convolutional neural network on sequence embeddings of previous items.

2.3 From Word2vec to Prod2vec

In the field of natural language processing, it is common to train neural language models based on low-dimensional word embeddings. These embeddings can be efficiently learned for large vocabularies by means of scalable word2vec methods like continuous bag of words (CBOW) and skip-gram (Mikolov et al., 2013a; Mikolov et al., 2013b). A very useful property of word2vec is that words that are similar to each other in the embedding space have a similar linguistic context, i.e. they are often used with the same words in a sentence. By assuming that words that have a similar context have a very similar meaning, word embeddings can be used to find related words and word groups in a corpus.

It is natural to transfer the concept of word2vec to recommender systems in terms of item2vec (or prod2vec), as proposed by (Ozsoy, 2016) for recommending new venues to visit to users or by (Grbovic et al., 2015) for e-mail advertisement. (Ozsoy, 2016) train venue embeddings based on bags of venues that are visited by users. A new recommendation is done by calculating the k-nearest venues in the embedding space using the cosine similarty. (Grbovic et al., 2015) perform a clustering approach upon the embeddings and recommend new items based on the transition probabilities between the target item and different clusters. In addition, inspired by the paragraph2vec approach (Le and Mikolov, 2014), user embeddings are incorporated in order to perform user-item recommendations. In (Vasile et al., 2016), the prod2vec approach is slightly extended to incorporate additional categorical product information to overcome product cold start problem.

Our approach follows the same intuition as the related works. Instead of the user history data we train the prod2vec model based on co-occurring products within browser sessions. In addition, we evaluate the end-to-end model against the proposed nearest neighbor approaches.

3 APPROACH

3.1 Prod2vec

We use the prod2vec approach for bigrams of products that are viewed together in browser sessions. Specifically, given the set P of products $p \in P$, we train a neural network on a set of all pairs of products (p_a, p_b) that occur in the same session. The objective is to train a D-dimensional embedding representation $e_p \in \mathbb{R}^D$ so that products that share the same co-occurring products have similar embedding vectors. We adapt the skip-gram technique for product pairs to train the underlying neural network $f(p)$. The aim of the network is, given an input product p_a, to pre-

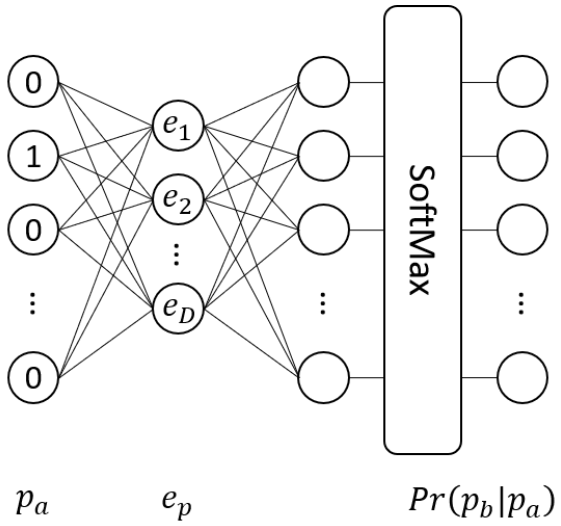

Figure 2: The adapted skip-gram architecture with an input layer for all products, an embedding layer and a softmax output layer.

dict the probabilities of all other products for being viewed after p_a. The architecture is depicted in figure 2. The input layer of the network contains the entirety of all existing products, with each product being one-hot-encoded. It is followed by a single hidden layer that represents the product embedding. The number of its neurons thus equals the dimension D of the embedding. The output layer applies the softmax activation function to train the probabilities for all other products. When training the network on product pairs (p_a, p_b), the objective is to minimize the negative log likelihood loss L over all m training examples:

$$L(f(p)) = \frac{1}{m}\sum_{i=1}^{m} -log(\hat{y}_i)$$

where $\hat{y}$ is the softmax output for all products P:

$$\hat{y} = \frac{e^{y_j}}{\sum_j^P e^{y_j}}$$

In this way, the network output learns the distribution of the products as they appear in the browser sessions for the given product p_a.

3.2 Recommender Approaches

Based on the trained neural network we derive and test three recommendation strategies. The first recommender uses the output of the network, which represents the probability of a product viewed in the same session as the input product. Thus, when making k new recommendations for a target product p_a, we take the top-k products according to the magnitude of their respective output probabilities. Since this approach uses the entire trained model, it will be referred to as the *e2e approach* in the following.

Our second recommendation strategy, the *k-nn approach*, is based on a nearest neighbor search in the product embedding space. We hereby take the trained embedding representations from the neural network. When we recommend k new products for a target product p_a, we look for the k nearest neighbors of p_a in the embedding space. We use the cosine similarity as a distance measure to calculate the product similarities.

The third and final recommendation strategy combines the intuitions behind the e2e and the k-nn approaches, hence it is subsequently referred to as the *hybrid approach*. First, we search for the one nearest neighbor p_b of the target product p_a in the embedding space. Then, we take the model outputs for p_b as a basis for recommendations. When recommending products, we prefer products that have a high probability output probability for p_b but a low probability for p_a. The motivation behind this strategy is to recommend appropriate, but also surprising products to a user.

4 USE CASE AND EXPERIMENTAL SETUP

4.1 User Interaction Data

In this paper, we investigate the described recommendation approaches for a commercial website which encounters millions of user interactions every week. Each of its product pages contains thirteen recommendations for other products that might be interesting for a user. Hereby four products are displayed directly, while the remaining nine products are provided upon the user requesting further recommendations. For analysis and advertising purposes some user activities on the website are recorded and stored. This includes the products the user views, some browser information, and a unique session ID. Thus, for each of the sessions we know how a user behaves and which products he has viewed. Unlike other recommendation use cases, there is no option to purchase products directly via the website. It is therefore important to note that the interaction data reflects user interests but not the actual purchasing behavior.

In our experiments we utilize historical data over a time span of two and a half years. From this data we extract all interactions which indicate the visit of a product page. With this step we filter out interactions irrelevant to a recommendation decision, e.g. visits

of the landing or contact page. We generate our data set by aggregating the interaction data for individual sessions and subsequently creating bigrams for each product combination in the respective session. The data set serves as the basis for training and testing the neural network-based recommenders.

4.2 Testing

In our experiments, we first train and test our recommender on historical data. We divide the data into a training set that comprises several month and a test set that contains product views of the last month, thus simulating the case of using a trained model for future recommendations. While the training minimizes the loss function on the data, we assess a recommender on the test set by calculating the hit rate (HR or top-k accuracy) on all product pairs:

$$HR_k = \frac{\#hits}{\#pairs}$$

where a hit occurs when the target product appears in the top-k recommendations for an input product. We further perform live A/B tests on the website, each test for a period of two weeks. After completion of an A/B test, we calculate the click through rates (CTR) of all tested recommenders:

$$CTR = \frac{\#clicked\_recommendations}{\#provided\_recommendations}$$

We implement our models and learning methods using PyTorch, an open source library for deep learning (Tor, 2020). For the evaluation, we first conducted initial grid search based tests on the historical data to identify the best performing topology and hyperparameters for the underlying neural network, resulting in a model with an embedding layer of 32 neurons with Rectified Linear Unit (ReLU) activation functions and the Adam optimization algorithm with a learning rate of 0.0001.

As part of the A/B tests, the models are compared with a control model that is currently in use for the website and that is based on the users' browsing histories. The recommendations provided by this model are derived from the co-occurrence of products that a user has viewed and calculated based on the DIMSUM algorithm (Zadeh and Carlsson, 2013; Zadeh and Goel, 2013). The implementation of DIMSUM is based on Apache Spark's implementation but with modifications for integrating with the rest of the existing system and memory improvements over the Spark model. This model is then run weekly and the results are stored to be later resolved when a user browses the relevant product pages. However, because this training only happens once a week, this does not account for products that are recently added to the system.

Every time a new product is added, we provide a fallback recommendation based on meta data provided with the products such as its name, type and description. Specifically, these recommendations are determined by the cosine similarity of the products calculated from a case-insensitive TF-IDF vector derived from a pre-determined set of a product's text-based attributes.

5 RESULTS

5.1 Recommender Performances

At first, we train the underlying neural network for the three recommender approaches on data of the last two and a half years, containing all product pairs that are viewed within a browser session. Figure 3 shows the test results of the approaches on the historical data. They show that the e2e-approach performs best, achieving a top-13 hit rate of 19.11%.

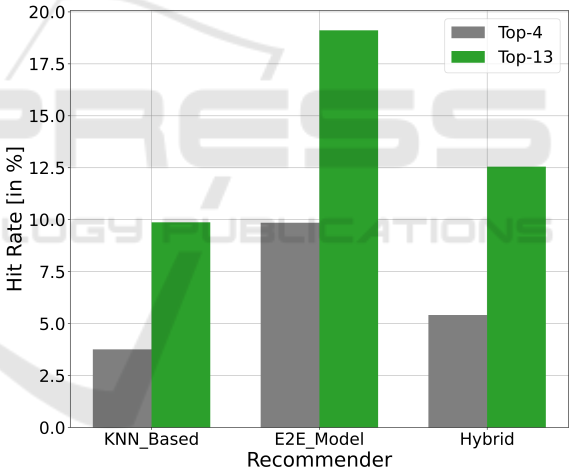

Figure 3: Calculated hit rates (top-4 and top-13 accuracies) of the three recommender approaches trained on all product pairs for the last two and a half years.

It can also be seen that the nearest neighbor based approach clearly performs the worst. As shown in figure 4, the results are also confirmed by the A/B test. The figure depicts how the CTRs of the recommendation approaches improve or deteriorate compared to the control approach. The results show that none of the mentioned approaches perform as well as the control (the e2e approach achieves about 82% of its CTR).

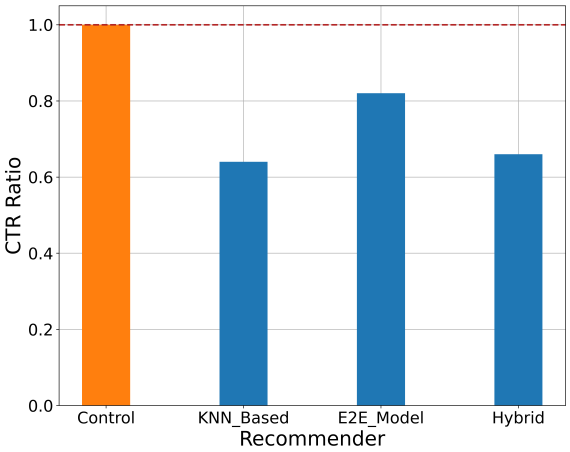

Figure 4: Evaluation results of the first A/B test for the control model and the three recommender approaches. Here, the ratios of the CTRs of the approaches to the CTR of the control model is shown.

5.2 Different Training Data Sizes

One reason for the weaker performances compared to the control model could be that the control model is derived solely from recent data, while the other recommenders are based on a much longer time span of 2.5 years. The large historical data pool might potentially include outdated data, keeping the new recommenders from picking up on current trends in user preference. To investigate this, we train the previously best performing recommender (e2e) on different periods of user activities. They range from two years to a single month. The resulting models are then evaluated and compared in a second A/B test. The results (see figure 5) show that changing the underlying time period has a significant impact on the performance of the recommendation system. It is shown that model training on more recent user activities delivers better results in general than activities that lie longer in the past. This indicates that user interest changes over time and throughout the year.

5.3 Consecutive Product Views

So far we have trained the neural network on all product pairs where the two products were viewed in the same session. However, as cookie-based sessions may last over multiple weeks, this pairing strategy also pairs up products between which many other products were viewed. Such training samples of weakly linked products may distort the temporal development of user preference. We therefore investigate the case where we train the neural network only on bigrams with product pairs that were viewed consecutively in the same session. Although this modification greatly reduces the size of the training data, the new data distribution allows the neural network to approximate the user behavior more precisely. On historical test data, the e2e-approach based on this new model achieves a top-13 hit rate of 46.9%. In the third and last A/B test, the approach is compared again to the control model. As shown in figure 6, the model achieves a 5.6% improvement in CTR over the control model.

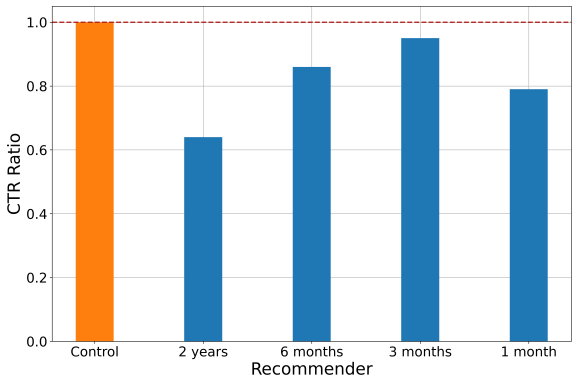

Figure 5: Evaluation results of the second A/B test for the control model and the e2e recommender approach trained on data sets with different time periods, ranging from the last two years to the single last month.

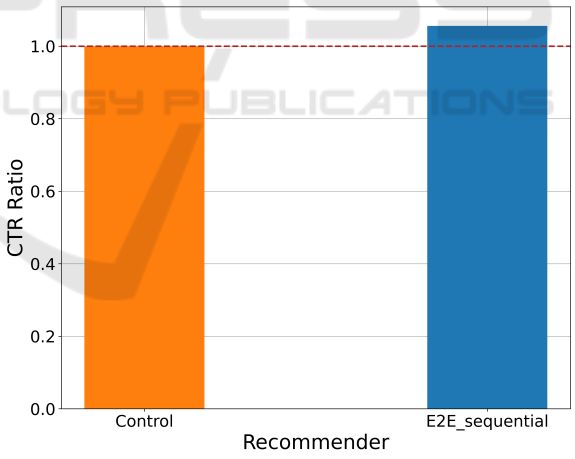

Figure 6: Evaluation results of the third A/B test for the control model and the e2e recommender approach trained on three months of user activity data containing consecutively viewed product pairs.

5.4 Further Discussion

The results of the three A/B tests show that the selection of a suitable training data basis has a considerable impact on the model performance. Figure 7 summarizes the results of the best recommendation approach, namely the E2E model, for the three tests.

Both the reduction of the training data to more recent user activities (difference from test two to one) and the focus on consecutive product views (difference from test three to two) yield significant improvements which, taken together, outperform the currently deployed recommendation system.

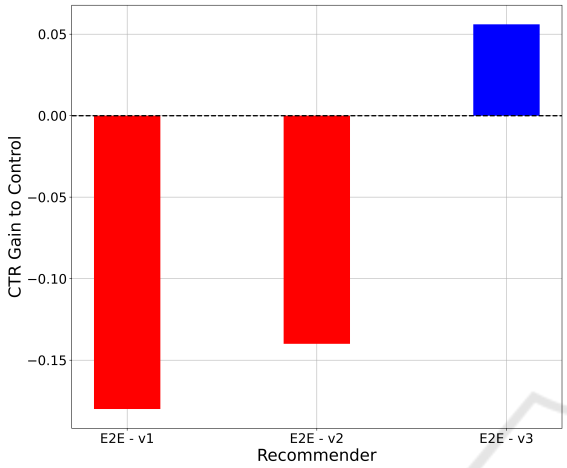

Figure 7: Performances (CTR) of the best performing model (E2E) across the three A/B-tests in relative to the CTR (gain) of the respective control model.

However, there is still potential for improvement. A crucial aspect here is that the proposed approach does not differentiate between users and user groups, so it always delivers the same product recommendations regardless of who the user is and, for example, from where he accesses the website. At least the latter is currently stored in the browser information in form of the time zone in which the user device is running. A breakdown of the CTR performance of the final model into the different time zones shows that the recommendation system performs very differently for different regions (see figure 8). Note that although the website is accessed worldwide, the majority of users are based in the USA. We can clearly see that while the recommendations for users from Eastern Standard Time (ET) perform better, they perform worst in Pacific Standard Time (PT). This indicates different user behavior in the regions, be it different recommendation click behaviors or different product preferences.

6 SUMMARY AND OUTLOOK

In this paper we addressed a real product-to-product recommender system for a commercial website by using the prod2vec approach. For this purpose we tested different recommenders based on an artificial neural network which is trained on historical user activity

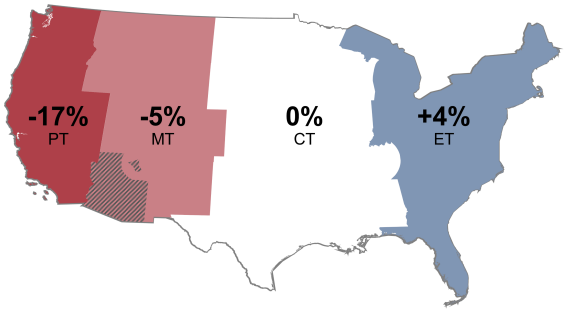

Figure 8: Performances of the E2E model in A/B test three for the individual time zones in the USA. The percentages represent the CTR of the respective time zone compared to the CTR of the Central Time Zone (CT).

data. The evaluation on the data as well as the performed A/B tests show on the one hand that an end-to-end neural network performs better than, for example, a nearest-neighbor approach, and on the other hand that the appropriate selection of the training data basis has a significant impact on delivering successful recommendations.

The best performing approach outperforms the currently used recommender system, which provides recommendations based on pre-calculated similarities of products. Yet, there are still further potential improvements to the proposed prod2vec approach. One possibility for improvement is the consideration of user and context information, such as the location or time of website access. To do this, the existing approach could be extended to incorporate additional input variables. Another possible improvement is the extension to handle sequential customer activities. Instead of making a recommendation based on the last product, the system would use the entire user history from the current browser session.

REFERENCES

(2020). Pytorch deep learning framework. Accessed: 2020-11-11.

Chen, H. and Lin, Z. (2019). A hybrid neural network and hidden markov model for time-aware recommender systems. In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, pages 204–213. INSTICC, SciTePress.

Cheng, H.-T., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., Shah, H., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., and Chai, W. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10, [Place of publication not identified]. ACM.

Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati,

N., Savla, J., Bhagwan, V., and Sharp, D. (2015). E-commerce in your inbox. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1809–1818, New York, NY. ACM.

Gui, Y. and Xu, Z. (2018). Training recurrent neural network on distributed representation space for session-based recommendation. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2016). Session-based recommendations with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196, Bejing, China. PMLR. 22–24 Jun.

Li, X. and She, J. (2017). Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 305–314, New York, NY. ACM.

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Ozsoy, M. G. (2016). From word embeddings to item recommendation. *arXiv preprint arXiv:1601.01356*.

Ricci, F., Rokach, L., and Shapira, B. (2015). Recommender systems: Introduction and challenges. In *Recommender Systems Handbook*, pages 1–34. Springer US, Boston, MA.

Shen, F. and Jiamthapthaksin, R. (2016). Dimension independent cosine similarity for collaborative filtering using mapreduce. In *2016 8th International Conference on Knowledge and Smart Technology (KST)*, pages 72–76.

Tamhane, A., Arora, S., and Warrier, D. (2017). Modeling contextual changes in user behaviour in fashion e-commerce. In *Advances in knowledge discovery and data mining*, LNCS sublibrary. SL 7, Artificial intelligence, pages 539–550, Cham, Switzerland. Springer.

Tang, J. and Wang, K. (2018). Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM 2018*, pages 565–573, New York, NY. Association for Computing Machinery.

Vasile, F., Smirnova, E., and Conneau, A. (Sept. 2016). Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 225–232, New York. Association for Computing Machinery.

Wang, H., Wang, N., and Yeung, D.-Y. (2015). Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244, New York, NY. ACM.

Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A. J., and Jing, H. (Feb. 2017). Recurrent recommender networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 495–503, New York. Association for Computing Machinery.

Wu, S., Ren, W., Yu, C., Chen, G., Zhang, D., and Zhu, J. (2016a). Personal recommendation using deep recurrent neural networks in netease. In *IEEE International Conference on Data Engineering, ICDE, 2016*, pages 1218–1229, [Piscataway, NJ]. IEEE.

Wu, Y., DuBois, C., Zheng, A. X., and Ester, M. (2016b). Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 153–162, New York, NY. ACM.

Zadeh, R. B. and Carlsson, G. (2013). Dimension independent matrix square using mapreduce. *arXiv preprint arXiv:1304.1467*.

Zadeh, R. B. and Goel, A. (2013). Dimension independent similarity computation. *Journal of Machine Learning Research*, 14(14):1605–1626.

Zhang, F., Yuan, N. J., Lian, D., Xie, X., and Ma, W.-Y. (2016). Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 353–362, New York, NY. ACM.

Zhang, S., Yao, L., Sun, A., and Tay, Y. (2019). Deep learning based recommender system: A survey and new perspective. *ACM Computing Surveys*, 52(1):1–38.

Zhou, Y., Wilkinson, D., Schreiber, R., and Pan, R. (2008). Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic aspects in information and management*, volume 5034 of *Lecture Notes in Computer Science*, pages 337–348. Springer, Berlin.