Should we Embed? A Study on the Online Performance of Utilizing Embeddings for Real-Time Job Recommendations

Markus Reiter-Haas *
Moshbit GmbH
Graz, Austria
markus.reiter-haas@moshbit.com

Emanuel Lacic *
Know-Center GmbH
Graz, Austria
elacic@know-center.at

Tomislav Duricic
Graz University of Technology
Graz, Austria
tduricic@know-center.at

Valentin Slawicek
Moshbit GmbH
Graz, Austria
valentin.slawicek@moshbit.com

Elisabeth Lex
Graz University of Technology
Graz, Austria
elisabeth.lex@tugraz.at

ABSTRACT

In this work, we present the findings of an online study, where we explore the impact of utilizing embeddings to recommend job postings under real-time constraints. On the Austrian job platform Studo Jobs, we evaluate two popular recommendation scenarios: (i) providing similar jobs and, (ii) personalizing the job postings that are shown on the homepage. Our results show that for recommending similar jobs, we achieve the best online performance in terms of Click-Through Rate when we employ embeddings based on the most recent interaction. To personalize the job postings shown on a user's homepage, however, combining embeddings based on the frequency and recency with which a user interacts with job postings results in the best online performance.

KEYWORDS

Job Recommendations; Online Evaluation; Real-time; Item Embeddings; Frequency; Recency; BLL Equation;

1 INTRODUCTION

Job recommender systems have become an integral part of both academia and industry for a few decades now [23], which is also illustrated by the fact that XING[1] has organized two recent RecSys Challenges [1, 2]. In the past, research on job recommendations has mainly employed various Collaborative- and Content-Based Filtering approaches or their hybrid combinations [3, 26] to improve the recommendation accuracy. Recently, learning latent item representations (i.e., embeddings) for recommender systems has become a popular technique and has shown state-of-the-art performance in the job domain. For example, the authors of [25] use Doc2Vec [17] to create job embeddings based on job-related content features. To test their approach, they conduct an offline evaluation, where they manually score the quality of similar jobs from a small subset of 100 randomly selected jobs. Other works [5, 16, 22] define the task at hand as an item-to-item recommendation problem and evaluate embedding approaches also in offline studies.

However, whether a user indeed accepts a recommendation can only be measured with either user studies or online evaluations. User studies are to date rarely used as they require active participation of users over a period of time [6] and online evaluations are expensive to set up, as they need a fully functional system with a significant userbase [7]. As a consequence, related work that reports on online studies of job recommender systems is scarce. For example, recent work [12] explored in an online study how to increase the engagement toward underserved jobs. Besides, in the case of the RecSys 2017 Challenge, the top-25 participating teams were allowed to publish their solutions once per day to be rolled out on the XING platform [2]. In line with recent research [8, 11], recommendation approaches used in an online evaluation usually need to consider real-time constraints [9, 24], such as having response times, which are below 100-200 milliseconds or immediately considering data updates in the next recommendation request.

The present work. In this work, we contribute to the sparse line of research on evaluating job embeddings under real-time constraints in an online setting. For this, we learn job embeddings using the popular Doc2Vec approach. We obtain fixed-length vectors from the job description text and investigate their impact on the online performance of recommending job postings in real-time. Similarly, as in the offline setting of [16], we further represent a user's browsing behavior by combining the extracted embeddings using a model from human memory theory, that integrates factors of frequency and recency of job posting interactions. To measure the real impact of such an approach, we perform several A/B tests on the Studo Jobs platform. That is, we compare against two popular recommendation use-cases that we tackle under the real-time constraint: (i) providing content-based recommendations for similar job postings to the one currently viewed by the user and, (ii) personalizing the homepage with job postings using collaborative filtering. Our findings suggest that in situations when we recommend similar job postings, using embeddings based on the most recent interaction tends to improve the online performance. In contrast, combining embeddings based on the frequency and recency with which a user interacts with job postings improves the online performance when we personalize the job postings on the homepage.

[*] Both authors contributed equally to this work.
[1] https://www.xing.com

RecSys '19, , September 16, 2019, Copenhagen, Denmark

https://doi.org/N/A

	Test	User Count	Reco Count	Days	Approach	CTR	↗	Runtime (ms)	↘
Similar Jobs	Impact of embeddings	8,576	31,968	32	CBF	0.0194	18.04%	51	23.53%
					LAST	**0.0229***		**39****	
	Influence of frequency and recency	4,715	18,464	15	LAST	**0.0249****	75.35%	**67****	28.72%
					BLL	0.0142		94	
	Merit of recency	3,375	11,992	15	$\text{BLL}_{d=0.6}$	**0.0174***	35.94%	97	2.06%
					$\text{BLL}_{d=0.4}$	0.0128		**95**	
Homepage	Influence of frequency and recency	9,620	26,334	25	BLL	**0.0671***	15.69%	**114****	13.64%
					CF	0.0580		132	
	Combining frequency and recency	9,313	24,907	19	HYB_{BLL}	**0.0471****	33.05%	172	38.37%
					CF	0.0354		**106****	

Table 1: We report the mean Click-Through Rate (CTR) and the mean Runtime of the approaches utilized in the corresponding A/B tests. The increase ↗ in accuracy and decrease ↘ in runtime is reported for the best performing approach. Moreover, we use the $*$ symbol to indicate it the results are significantly better with a p-value < 0.05 and the $**$ for results with a p-value < 0.0005.

2 RECOMMENDATION STUDY

Our study is carried out in the Studo Jobs[2] platform. We tackle two distinguished recommendation scenarios, which the platform supports. First, we recommend similar jobs. Second, we personalize the ranked list of all possible job postings in the system to improve engagement on the homepage of Studo.

As shown in an offline study in [16], learning embeddings on the textual description of job postings can improve both the accuracy and diversity of content-based recommendations. In the present work, we learn embeddings of job postings by utilizing Doc2Vec [17], a variation of the widely popular Word2Vec [19] approach. In order to investigate the online performance of the extracted job embeddings[3] under real-time constraints, we employ two variants for performing content-based recommendation of job postings, which are described in this section. For evaluation, we measure both Click Through Rate (CTR) and runtime.

Utilizing the most recent job interaction (LAST). A natural way of using embeddings is to apply them in a content-based manner (e.g., [20]). That is, given a reference vector representation, the task is to find the top-k similar vectors (i.e., job postings) using the Cosine similarity. As in [16], to obtain this reference vector, we use the embedding of the last (i.e., most recent) job posting with which the user has interacted. With this recommendation strategy, we can study the online performance of the recommender when we recommend jobs that are similar to the one the user is currently viewing.

Integrating interaction frequency and recency (BLL). One issue of the previously mentioned LAST recommendation strategy is that it solely focuses on the factors of interaction recency. However, related work has shown that past interaction frequency and recency are crucial factors for personalization [13, 16]. In this respect, the cognitive architecture ACT-R defines the Base-Level Learning (BLL) equation, which integrates these two factors to model the information access in human memory. Thus, to simultaneously account for both frequency and recency factors of job posting interactions, we use the BLL equation to model a user's browsing behavior:

$$BLL_{u,j} = \ln\left(\sum_{i=1}^{n}(TS_{ref} - TS_{j,i})^{-d}\right) \qquad (1)$$

where $BLL_{u,j}$ is the BLL value for a given user u and a given job j, and n states the number of times u has interacted with j in the past. Moreover, $TS_{j,i}$ is the timestamp (in seconds) of when u has interacted with j for the i-th time and TS_{ref} is a reference timestamp such as the time when the job recommendations are requested. The parameter d is used to set the time-dependent decay of item exposure in human memory and unless stated otherwise, we set it to its default value of 0.5 (i.e., according to Anderson et al. [4]).

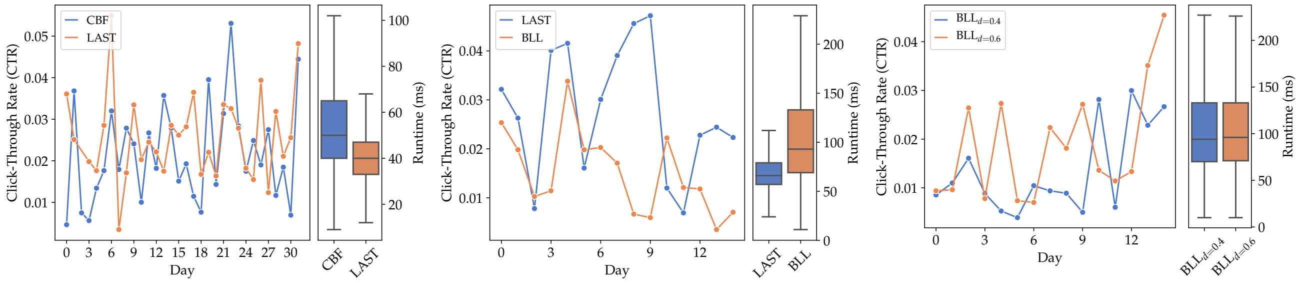
In this work, we use Anderson's model of human memory theory to create a reference vector representation that can be used in a content-based manner (i.e., to find similar job postings). For that, we first normalize the BLL values using a softmax function and then multiply them with the vector representations assigned to the individual job postings from a user's browsing history. This way, we form a weighted sum of embeddings based on how frequently and recently the user has interacted with the particular job postings. As previous offline experiments have shown [16], utilizing embeddings in such a way results in a recommendation performance with lower accuracy but higher diversity when compared to the previously mentioned LAST approach.

Adapting for real-time job recommendations. In practice, response times of recommendations need to be below 100-200 milliseconds [9, 24]. To adapt the LAST and BLL approaches for an online setting (i.e., to provide recommendations in real-time), we further propose to store the learned job embeddings in the form of payloads in Apache Lucene[4]. Payloads are a general purpose array of bytes that are associated with a Lucene token at a particular position. Each job posting is thus annotated with multiple positions of the latent vector dimensions. The latter positional information can be used for fast retrieval and calculation of vector similarities (i.e., utilizing the Cosine similarity) at runtime. For the online study on the Studo Jobs

[2] https://studo.co/jobs

[3] We obtain the embeddings using a Doc2Vec model that we train with a window size of 20, a learning rate of 0.025 and 10 negative samples.

[4] An example of how to implement this functionality in Elastic Search, a search engine that is built on top of Apache Lucene can be found at the following link: https://github.com/lior-k/fast-elasticsearch-vector-scoring.



(a) Impact of embeddings

(b) Influence of frequency and recency

(c) Merit of recency

Figure 1: Analysis of incorporating job embeddings to recommend similar jobs. The reported results show a daily CTR and the distribution of the measured runtime performance.

platform, we use Apache Solr[5] to store, retrieve and calculate the similarity of job embeddings in real-time.

Experimental setup. In our experiments, we measure the Click-Through Rate (CTR) and the runtime performance. To obtain the CTR, we compute the percentage of recommended job postings with which the users have interacted. For the runtime analysis, we measure the time it takes to generate each recommendation in milliseconds. We compare two approaches at a time (i.e., conduct an A/B test) to avoid being subject to periodical changes and other anomalies. For this reason, we divide our userbase into two equal groups and assign them to one of the two approaches that we evaluate. We further perform a chi-squared test on the measured recommendation outcome (i.e., a user either did or did not engage with a recommendation) and a t-test on the runtime performance to determine if the differences in the reported results are statistically significant. Concerning real-time constraints, all recommendation approaches in the Studo Jobs platform calculate new recommendations for every request and filter out those job postings that the user has already interacted with in the current session.

3 SIMILAR JOBS

When users view a particular job posting in the Studo Jobs platform, recommendations with similar, alternative jobs are shown to them. The location of the shown recommendations depends on the layout of the device used. On the desktop, the recommendations appear in the sidebar, while on a mobile device they will appear under the job posting description. Furthermore, this type of recommendations only suggests a short list of 3 alternative job postings.

Baseline: Content-Based Filtering (CBF). A popular method in many systems for recommending similar items (i.e., jobs) is Content-Based Filtering [3, 21]. This method analyses item metadata to identify other items that could be of interest for a specific user. In Studo, this is done using TF-IDF on the description text of the job posting with which the user currently interacts. Besides being a typical pick for recommending similar items, another reason for using CBF is that it can easily be adapted for an online setting, where recommendations need to be served in real-time[6].

Impact of embeddings. The initial aim of this work is to investigate if utilizing embeddings, which we learn from the textual content of job postings, can outperform traditional content-based recommendations when used in a similar item scenario. For this, we first did a preliminary A/B test of the LAST approach to evaluate the impact of the embedding size. We found that in the case of Studo, having an embedding size larger than 100 did not contribute to a higher CTR, but did increase the overall runtime performance. Table 1 reports on all A/B tests, for which, we use 100 as the dimension size of job embeddings.

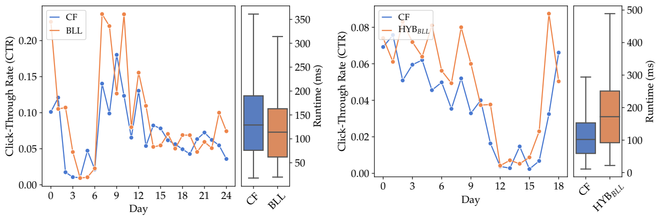
In Figure 1a, we report the performance of the LAST approach when compared to the CBF baseline. As seen, the CTR varies over the 32-day testing period, but overall, using the job embedding from the currently viewed job posting leads to a significant increase of the CTR by 18.04%. Moreover, utilizing embeddings in such a way resulted in a 23.53% lower runtime (i.e., as reported in Table 1), which is a desirable effect when providing recommendations in real-time.

Influence of frequency and recency. Building upon the insights on the impact of using embeddings, we evaluate the model of human memory theory during a shorter, 15-day period. That is, we investigate if we can further enhance the recommendation performance by using the proposed BLL equation from Section 2 to create the reference job vector. Interestingly enough, Figure 1b clearly shows that modeling a user's browsing behavior in this manner did not result in better performance than the LAST approach in terms of both CTR and runtime. As seen in Table 1, we get the highest relative difference in CTR which did not justify the increased computational overhead, that resulted in higher runtime performance.

Merit of recency. We hypothesize that the BLL approach did not exhibit a better performance due to the specific recommendation scenario where it was applied in (i.e., showing similar jobs to the currently viewed one). This suggests that factors of recency are especially influential in this setting and as such, we perform an additional experiment to confirm this effect.

[5] A search engine that, similar as Elastic Search, is built on top of Apache Lucene: https://lucene.apache.org/solr/.

[6] As shown in [15], we leverage the built-in functionality of the Apache Solr search engine to recommend jobs with the most similar textual content.



(a) Influence of frequency and recency **(b) Combining frequency and recency**

Figure 2: Online performance of job embeddings when used to personalize the homepage.

In our previous experiment, we set the time parameter d from the BLL equation to have the default value of 0.5. However, this parameter changes the rate at which things will be "forgotten." Thus, it controls the decay of the impact of consumed items at an exponential rate. The question is therefore on whether a shorter memory (i.e., higher time decay) or a long memory is better for the setting of recommending similar job postings. For this experiment, the exponents $d = 0.6$ (shorter memory) and $d = 0.4$ (longer memory) were compared. As seen in Figure 1c and Table 1, favoring shorter memory (i.e., recency) when calculating the BLL equation resulted in a significantly better CTR. Indeed, this confirms the described effect where users expect recommendations which are similar to the more recent browsing behavior.

4 HOMEPAGE

As in many other systems, personalization in the Studo Jobs platform starts already on the homepage. The homepage consists of a list of 25 job postings from which the first 5 are the calculated recommendations. The advantage in this setting is the seamless integration of recommendations with the list of available job postings. Moreover, users not only first visit the homepage when they access the Studo Jobs portal, but also often come back to it after they stop exploring a given job posting. Such behavior results in the homepage being responsible for more than 80% of all recommendations that the user has interacted with and thus suggests to be a better fit for applying the model of human memory theory to represent the user's browsing behavior.

Baseline: Collaborative Filtering (CF). To this day, one of the most explored and utilized techniques for personalizing a system in real-time is Collaborative Filtering [10, 15, 18]. The Studo Jobs portal uses the User-Based Collaborative Filtering approach to personalize the job postings on the homepage. In that setting, a target user will get those job postings recommended that have been previously interacted with by similar users (i.e., the neighbors). As shown by [14], to provide recommendations in real-time, the inverted-index structure available in the Apache Solr search engine is used to find the k-nearest neighbors using the Cosine similarity metric.

Influence of frequency and recency. As previously stated, we hypothesize that by incorporating the factors of frequency and recency from a user's browsing behavior, we can further enhance the online performance of recommendations on the homepage. For this, we use the BLL equation on the extracted embeddings from the user's interaction history to create a reference vector representation and recommend the top-k similar job postings. To account for cold-start users, we utilize the most popular job postings as a fallback[7].

As seen in Figure 2a and the second part of Table 1, using the BLL equation on embeddings from the user's job history manages to significantly outperform the CF baseline for both, the CTR and the runtime performance. Such results suggest that the scenario of personalizing the homepage is indeed a setting where the user expects the recommendations to consider both, factors of frequency and recency of her browsing behavior.

Combining frequency and recency. Instead of replacing the CF baseline with the BLL approach, we further explore the efficacy of a hybrid combination which uses these two approaches in a round-robin fashion. We assume that for the homepage, where the user interacts most with the provided recommendations, it would make sense to allow picking from multiple sources of relevant job postings since such a recommendation strategy has often lead to the best performance in offline evaluation settings (e.g., [16, 26]).

As seen in Figure 2b and the last row of Table 1, the hybrid combination of the BLL approach and the CF baseline also performs significantly better than the CF baseline concerning CTR. The relative improvement of 33.05% for the CTR in this A/B test is much better than in the case when we just used the BLL approach on its own. This, however, comes with a trade-off, namely, with a significant increase in the runtime.

5 CONCLUSION

In this work, we contributed to the sparse line of research on evaluating job embeddings under real-time constraints in an online setting. We performed a variety of A/B tests on the Studo Jobs platform and ran evaluations concerning CTR and runtime for two different recommendation scenarios, namely, recommending similar jobs and personalizing the job postings that are shown on the homepage.

We found that for the case of recommending similar jobs, using embeddings based on the most recent interaction provides the best online performance. In contrast, combining embeddings based on the frequency and recency with which a user interacts with job postings significantly improves the online performance when we personalize the jobs on the homepage.

Limitations and Future Work. While Doc2Vec is a popular choice for learning item embeddings, other deep learning methods such as, e.g., Autoencoders or Convolutional Neural Networks might also perform well for this task. Furthermore, we did not explore the impact of using additional user or job-related metadata on the quality of learned embeddings. We also did not study the effects of the time-dependent decay parameter d from the model of human memory theory to a greater extent for personalizing the jobs shown on the homepage. As such, we aim to tackle these points as future work. Finally, the data we used for this study is proprietary and owned by Moshbit, the owner of Studo. Currently, we cannot release this data to the research community due to Moshbit's terms of service.

[7] Such a fallback strategy is used for every recommendation approach on the homepage of the Studo Jobs portal (including the CF baseline).

REFERENCES

[1] F. Abel, A. Benczúr, D. Kohlsdorf, M. Larson, and R. Pálovics. Recsys challenge 2016: Job recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 425–426. ACM, 2016.

[2] F. Abel, Y. Deldjoo, M. Elahi, and D. Kohlsdorf. Recsys challenge 2017: Offline and online evaluation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 372–373. ACM, 2017.

[3] S. T. Al-Otaibi and M. Ykhlef. A survey of job recommender systems. *International Journal of Physical Sciences*, 7(29):5127–5142, 2012.

[4] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychological review*, 111(4):1036, 2004.

[5] O. Barkan and N. Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016.

[6] J. Beel, M. Genzmehr, S. Langer, A. Nürnberger, and B. Gipp. A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. In *Proceedings of the international workshop on reproducibility and replication in recommender systems evaluation*, pages 7–14. ACM, 2013.

[7] P. G. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1-2):67–119, 2014.

[8] B. Chandramouli, J. J. Levandoski, A. Eldawy, and M. F. Mokbel. Streamrec: a real-time recommender system. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 1243–1246. ACM, 2011.

[9] C. Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich, and J. Leskovec. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1775–1784. International World Wide Web Conferences Steering Committee, 2018.

[10] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

[11] Y. Huang, B. Cui, W. Zhang, J. Jiang, and Y. Xu. Tencentrec: Real-time stream recommendation in practice. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 227–238. ACM, 2015.

[12] K. Kenthapadi, B. Le, and G. Venkataraman. Personalized job recommendation system at linkedin: Practical challenges and lessons learned. In *Proc. of ACM RecSys'17*.

[13] D. Kowald, S. C. Pujari, and E. Lex. Temporal effects on hashtag reuse in twitter: A cognitive-inspired hashtag recommendation approach. In *Proc. of WWW'17*.

[14] E. Lacic, D. Kowald, and E. Lex. Neighborhood troubles: On the value of user pre-filtering to speed up and enhance recommendations. *arXiv preprint arXiv:1808.06417*, 2018.

[15] E. Lacic, D. Kowald, D. Parra, M. Kahr, and C. Trattner. Towards a scalable social recommender engine for online marketplaces: The case of apache solr. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 817–822. ACM, 2014.

[16] E. Lacic, D. Kowald, M. Reiter-Haas, V. Slawicek, and E. Lex. Beyond accuracy optimization: On the value of item embeddings for student job recommendations. *arXiv preprint arXiv:1711.07762*, 2017.

[17] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proc. of ICML'14*.

[18] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, (1):76–80, 2003.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[20] C. Musto, G. Semeraro, M. de Gemmis, and P. Lops. Learning word embeddings from wikipedia for content-based recommender systems. In *European Conference on Information Retrieval*, pages 729–734. Springer, 2016.

[21] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[22] V.-T. Phi, L. Chen, and Y. Hirate. Distributed representation based recommender systems in e-commerce. In *DEIM Forum*, 2016.

[23] R. Rafter, K. Bradley, and B. Smyth. Personalised retrieval for online recruitment services. In *The BCS/IRSG 22nd Annual Colloquium on Information Retrieval (IRSG 2000), Cambridge, UK, 5-7 April, 2000*, 2000.

[24] A. Said, J. Lin, A. Bellogín, and A. de Vries. A month in the life of a production news recommender system. In *Proceedings of the 2013 workshop on Living labs for information retrieval evaluation*, pages 7–10. ACM, 2013.

[25] J. Yuan, W. Shalaby, M. Korayem, D. Lin, K. AlJadda, and J. Luo. Solving cold-start problem in large-scale recommendation engines: A deep learning approach. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1901–1910. IEEE, 2016.

[26] C. Zhang and X. Cheng. An ensemble method for job recommender systems. In *Proceedings of the Recommender Systems Challenge*, page 2. ACM, 2016.