XWalk: Random Walk Based Candidate Retrieval for Product Search

Jon Eskreis-Winkler[1], Yubin Kim[1] and Andrew Stanton[1]

[1]*Etsy, Inc., Brooklyn, NY, USA*

Abstract

In e-commerce, head queries account for the vast majority of gross merchandise sales and improvements to head queries are highly impactful to the business. While most supervised approaches to search perform better in head queries vs. tail queries, we propose a method that further improves head query performance dramatically. We propose XWalk, a random-walk based graph approach to candidate retrieval for product search that borrows from recommendation system techniques. XWalk is highly efficient to train and inference in a large-scale high traffic e-commerce setting, and shows substantial improvements in head query performance over state-of-the-art neural retreivers. Ensembling XWalk with a neural and/or lexical retriever combines the best of both worlds and the resulting retrieval system outperforms all other methods in both offline relevance-based evaluation and in online A/B tests.

Keywords

e-commerce search, product search, graph, random walks, implicit feedback

1. Introduction

Modern large-scale search systems are tiered [1] with at least two layers. The *candidate retrieval* layer generates a small subset of potentially relevant documents from a corpus many orders of magnitude larger in size, while emphasizing efficiency and recall. The *re-ranking* layer uses more computationally expensive methods to re-rank the candidates generated by the retrieval stage to produce a high-precision final result list. Better recall in candidate retrieval leads to better overall accuracy. In this paper, we focus on improve search through improving recall in the candidate retrieval layer.

Most evaluations for search systems use an evaluation query set in which every query is assumed to be equally important and has equal impact on the accuracy metric. However, in reality, query frequency distributions are exponential [2]. Consequently, in e-commerce, head queries account for the vast majority of gross merchandise sales and head query performance is far more impactful to business metrics than torso or tail performance. State-of-the-art supervised neural dense retrievers [3, 4, 5, 6, 7] typically perform better in head queries than tail, due to the higher availability of training data in the head region. However, we show that further substantial improvements to head query performance are possible. We borrow ideas from the recommendation systems community and propose XWalk, a graph-based approach to candidate retrieval.

eCom'23: ACM SIGIR Workshop on eCommerce, July 27, 2023, Taipei, Taiwan

jeskreiswinkler@etsy.com (J. Eskreis-Winkler); ykim@etsy.com (Y. Kim); astanton@etsy.com (A. Stanton)

Historically, graph-based approaches in search were used to create features (e.g. PageRank, click graphs [8, 9]) for the re-ranker layer, but have not been used directly for retrieval. Recently, graph neural networks (GNNs) have achieved state of the art performance in recommendation and are being adapted for search [10, 11, 12, 13]. However, large-scale GNNs are complex and slow to train.

The recommendation systems have long used implicit interaction graphs to directly generate recommendations. Commonly, users and product listings are represented as nodes in a graph and edges represent a logged interaction between a user and product listing (e.g. the user purchasing the listing). Random walks in graphs is a powerful technique used to generate recommendations from interaction graphs [14, 15, 16, 17]. Random walk based approaches are frequently used in large, real-time recommendation systems due to their effectiveness and efficiency [17, 16]. In addition, when using implicit feedback (e.g. logged interaction data such as user clicks) Park et al. [14] showed that random walk based approaches can perform better than matrix factorization approaches.

XWalk uses a random walk based approach to perform candidate retrieval for product search. In XWalk, we cast search as a query-to-listing recommendation problem (as opposed to user-to-listing), that is, we transform our query log into a implicit interaction graph between queries and product listings, and perform candidate retrieval by "recommending" listings to queries. Our approach trains using a fraction of the time and resources used by neural dense retrievers and GNNs, and is highly efficient in inference – XWalk scales to real-time search over graphs of billions of nodes and tens of billions of edges. XWalk also excels in head queries, where implicit feedback signals are plentiful.

While XWalk on its own suffers in tail and novel queries, we show that when results from XWalk are ensembled with a typical retriever that uses text similarity, even one as basic as plain BM25, it substantially improves overall candidate retrieval accuracy compared to strong neural dense retrieval and hybrid retrieval baselines, especially over the head query region, which is responsible for the overwhelming majority of sales in e-commerce. Furthermore, we show that XWalk is complementary to *both* dense retrieval and BM25, and demonstrate the strength of ensembling all three approaches.

To summarize, our novel contributions are: a) showing that XWalk substantially improves performance in the head query region, which accounts for the overwhelming majority of sales in e-commerce; b) presenting an efficient random walk inference algorithm that can effectively serve queries at scale; c) showing that XWalk is complementary to other common retrieval methods and showing the strength of a simple ensemble approach that combines XWalk, BM25, and dense retrieval.

2. Method

We take inspiration from the recommendation space and recast the search problem as a query-to-product listing recommendation problem using implicit feedback: predict the best k product listings L_{q_i} to "recommend" to a query q_i, by learning from implicit user feedback, i.e. a query log. From the query log, we construct an undirected, weighted bipartite graph $G = (Q, L, E, W)$ where Q are nodes representing queries, L are nodes representing product listings, E are edges

$E = \{e_{i,j} = (q_i, l_j) \mid q_i \in Q \wedge l_j \in L\}$, and W are edge weights.

2.1. Graph Construction (Offline Training)

Given a query log which records for each query q_i the set of listings L_i^{click}, L_i^{cart}, $L_i^{purchase}$ that the user clicked on, added to their shopping cart, and purchased, respectively, we construct our graph through the following process:

1. For each unique (by text string) query in the query log $\hat{q}_i$, add $\hat{q}_i$ to Q.
2. For each unique (by listing ID) listing in the query log l_j, add l_j to L.
3. Collate the query log by query-listing pairs $(\hat{q}_i, l_j)$, counting the number of occurrences of $click_{i,j}$, $cart_{i,j}$, and $purchase_{i,j}$ interactions for each unique $(\hat{q}_i, l_j)$ pair.
4. For each $(\hat{q}_i, l_j)$, add $e_{i,j}$ to E and its weight $w_{i,j}$ to W, where $w_{i,j}$ is calculated Equation 1.

Intuitively, edge weights represent the popularity or trustworthiness of the edge, i.e. if many different users bought listing l_i from query q_j, $w_{i,j}$ will be higher because we are more confident in the relationship represented by the edge. To weight edges, we use a simple linear combination:

$$w_{i,j} = C_1 \cdot |click_{i,j}| + C_2 \cdot |cart_{i,j}| + C_3 \cdot |purchase_{i,j}| \tag{1}$$

In practice, the best coefficients are $C_1 < C_2 < C_3$, as the goal is to bias walks toward listings which convert well for a given query.

2.1.1. Graph representation for efficient inference

XWalk is designed for sparse graphs scaling up to billions of nodes and tens of billions of edges. The costliest part of random walk graph inference is sampling edges to walk, especially from high degree nodes. For efficient inference, we choose our graph representation carefully.

We store edge weights as cumulative distribution functions in order to use Inverse Transform Sampling, which allows sampling in $O(log(N))$ time. Note, we choose this approach over the alias method, which allows for constant time sampling, due to the doubling of memory needed for the transform. As XWalk's space complexity is dominated by edges and corresponding weights, we develop other methods for efficient sampling (Section 2.2).

To transform edge weights in to CDF format, for each node n, we sort its adjacent edges $E_{n,*}$ in decreasing order of their weights $W_{n,*}$, such that $w_{n,i} > w_{n,i+1}$. We then compute the cumulative distribution of all weights:

$$CDF_{n,i} = \frac{\sum_{j=0}^{i} w_{n,j}}{\sum_{i=0}^{|E_{n,*}|} w_{n,i}} \tag{2}$$

To sample an edge from $E_{n,*}$, we randomly sample $p \sim Uniform(0, 1)$ and find the corresponding edge through binary search. This formulations provides us a few valuable advantages:

1. Weighted sampling is $O(log(|E_{n,*}|)$. Given some nodes have degrees in the millions, logarithmic growth is critical for performance.

2. Normalizing the CDF to 1 allows us to reconstruct the the transition probability for outbound edges. This is key for the Metropolis-Hastings sampling strategy (Section 2.2).
3. Better cache coherence as the bulk of the weights are located near the front of the distribution.

Finally, we convert the graph into Compressed Sparse Row format, guaranteeing a $O(1)$ lookup cost for edges.

Note that all of the above graph construction steps are simple ETL (extract, transform, load) operations with no expensive parameter training steps. Compared to neural dense retrievers, "training" an XWalk graph model takes only a fraction of the cost and time.

2.2. Graph Inference (At Query Time)

Inferencing a graph with random walks is challenging to do efficiently. Despite the $O(1)$ edge lookup guarantee of the Compressed Sparse Row format used in graph construction, a naive walk approach that uses depth first search and binary search node lookups create random memory access patterns which result in high rates of costly cache misses [18]. We present an approach for XWalk that scales to graphs of billions of nodes and tens of billions of edges.

At query time, XWalk retrieves relevant listings for a query q_i by sampling nodes in G using k-hop fixed paths [15, 16] with node q_i as the starting point. When k is an odd number, the last node in a k-hop path will always be a listing node (L) due to the bipartite nature of G. XWalk returns listings ranked by the frequency of which they were sampled.

To reduce costly random memory access patterns, we use a breadth first search instead of depth first search for our random walks. We also improve upon the Inverse Transform Sampling strategy by using the Metropolis-Hastings algorithm (a Markov chain Monte Carlo method) in most places. Given the sorted CDF format of edge weights (Eq. 2), we can reconstruct the original edge transition probabilities: $P(n_j|E_{n_i,*}) = w_{n_i,j} - w_{n_i,j-1}$. As Metropolis-Hastings requires a symmetric distribution, we take the absolute value of the proposal index for each edge and sample from the Normal distribution. Ablation testing indicated XWalk is not sensitive to the variance for the proposal distribution, σ^2. We set $\sigma^2 = 0.2$.

Metropolis-Hastings improves the cost of c edge samples to $O(\log(|E_{n,*}|)) + c$ compared to $c * O(\log(|E_{n,*}|))$ of Inverse Transform Sampling. In cases where c is large (e.g. the initial query node), the computational improvements are substantial. A known limitation of MCMC methods is the auto-correlation of samples, usually requiring a mix time prior to sampling. Therefore, for our first sample, we use Inverse Transform Sampling to get an unbiased starting point and use Metropolis-Hastings for subsequent samples. In preliminary testing we found no reduction in model accuracy for this implementation compared to using only Inverse Transform Sampling while seeing the expected substantial latency benefits.

Our overall random walk strategy is presented in Algorithm 1.

2.3. Extending the Graph

Our e-commerce platform is a two-sided marketplace and our inventory comes from independent sellers. Thus, listings are naturally grouped by shops. In addition, sellers may add tags to their listings to better describe them (e.g. "christmas", "gift", etc.).

Algorithm 1: XWalkBFSSampler

```
1  Global variables: Var of Normal distribution $\sigma^2$, Dictionary of nodes to counts $Counter$
2  Input: Starting node $n$, Number of walks $c$, Walk-length $k$, Edges $E$, Weights $W$, Multiplier $m$ (default 1)
3  $p \sim Uniform(0, 1)$
4  $i = BinarySearch(E_{n,*}, W_{n,*}, p)$
   /* the $i$'th node of ordered neighbors of $n$                    */
5  $Counter[node(E_{n,i})] += m$
6  for step = {2, .., c} do
7      $j = Metropolis(i, E_{n_i,*}, W_{n_i,*}, \sigma^2)$
       /* the $j$'th node of ordered neighbors of $n_i$              */
8      $Counter[node(E_{n_i,j})] += 1$
9      $i = j$
10 end
11 if k > 0 then
12     $counts = \emptyset$
13     for $\{n_i, count\} \in Counter$ do
14         $counts = counts \cup XWalkBFSSampler(n_i, count, k-1, E, W)$
15     end
16     return $counts$
17 else
18     $Nodes = Sort(node \in Counter)$ in non-increasing order $\{Counter[n_1] \geq Counter[n_2] \ldots \geq Counter[n_{|Counter|}]\}$
19     return Nodes
20 end
```

For the sake of notation simplicity, we described the graph construction and inference above assuming our graph only contains two types of nodes, Q and L. However, in practice, we extend the graph by adding shop nodes (S) and tag nodes (T) to the graph; this allows us to retrieve listings without implicit user feedback (e.g. the cold start problem) and further increase connectivity of the graph. Note that G remains bipartite: $\{Q, S, T\}$ is a separate partition from L and thus the algorithms described in this section can be used unchanged. The weights of edges between shops/tags and listings are set to 1. $w_{q_i,s_j} = w_{q_i,t_j} = 1$.

3. Experiments

For our experiments, we sought to closely emulate a real-world e-commerce setting, where the main source of training data is implicit user feedback from query logs, and models are evaluated under a realistic query popularity distribution. Unfortunately, most public search datasets do not reflect a realistic query distribution and rarely have implicit user feedback as training data. While recommendation system datasets have implicit user feedback, they do not have a text query that is usable by BM25 or dense retrieval, which retrieve based on query to listing text similarity. Therefore, we curated a training and evaluation dataset from our e-commerce platform.

3.1. Dataset Creation

For training data, we collected 365 days of implicit feedback data, comprising of records of queries and the product listings that were clicked, added to cart, or purchased from a given query. Queries are represented by their query text. Listings are represented by their unique ID and the title of the product. In addition, as mentioned in Section 2.3, listings are associated with seller-provided tags, and each listing belongs to exactly one seller's shop.

Over the time period used for this experiment there were 137,824,871 unique listings, 147,174,817 unique queries, 62,803,463 unique tag, and 3,018,713 unique shops. There were a total of 1,349,734,328 query-listing interactions recorded, where 3.46% were purchases, 6.19% were cart adds and 90.3% were clicks. Altogether, there were 1,395,759,140 edges. Example records are found in Table 1.

query	ID	listing title	interaction	shop	tags
wedding dress	l12	beautiful bridal wedding gown	click	s00	wedding
wedding gown	l12	custom embroidered wedding dress	purchase	s00	wedding, gown
wedding dress	l34	ethereal dress with chiffon skirt	click	s11	dress, chiffon

Table 1
Hand-created example of implicit feedback training data.

Evaluation data was curated to be a representative query distribution, sampled from a single day immediately following the last day of the training data window. We randomly sampled 11,521 queries that resulted in at least one purchase. As the sample is intended to be reflective of the true query popularity distribution, we did not de-duplicate the query set. Figure 1 shows the distribution of the query frequency in the evaluation set.

For each query, the listings that were purchased from that query are considered the relevant document. 82.3% of queries had only one purchase, 12.0% had two purchases and 5.6% had more than two purchases. For each one of these queries, we assigned them to a head/torso/tail frequency bin based on how frequently they occurred in the previous 365 day period. The bins were created such that the total counts of requests are roughly equal among those bins. Of the evaluation queries, 31.0% were in the head bin, 47.9% were in the torso bin, and 43.9% were in the tail bin.

3.2. Experiment set up

We compare XWalk against two other methods of candidate retrieval. First is lexical retrieval using BM25 scoring (BM25). We use Pyserini [19] to build a Lucene index based on listing titles in our dataset and then retrieve candidates using BM25 rankings using bag-of-words representations. We used the default analyzer and default BM25 parameters (k1=1.2, b=0.75).

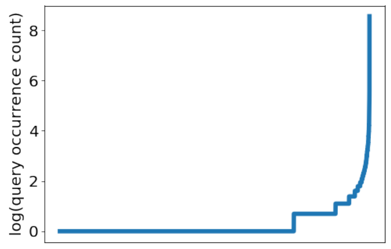

Figure 1: Distribution of evaluation queries' log frequency, ordered from least frequent to most frequent.

The second baseline is a state of the art neural dense retrieval system [20] trained on search traffic for candidate retrieval (NIR). NIR uses a smaller time window of training data (30 days) due to the time and expense of training on larger data sets. NIR is a Transformer-based, two tower model that uses a multi-part hinge loss to distinguish between interactions that involve a purchase, cart add, favorite, click, or nothing. The model was trained over one epoch. It was designed for better semantic matching between queries and listings by incorporating title, query as well as additional features such as tags, and listing taxonomy.

In addition to the above, we also compare results against hybrid systems of NIR+BM25 [6]. To ensemble the results from each retrieval engine, we use Reciprocal Rank Fusion, a simple but effective fusion technique [6]. Higher recall in candidate retrieval result in higher overall search accuracy [6]. As we are focusing on the first-pass candidate retrieval stage of search, we use recall and mean average precision (MAP) at 100, and 1000 to measure the quality of candidates retrieved.

4. Analysis

As shown in Table 2, when compared independently against other methods, XWalk out-performs other methods in most metrics despite the fact that it is unable to return results for novel queries, due to its strength in the head query bin. When combined with BM25, it outperforms in every metric, both NIR and the hybrid NIR+BM25. Finally, the ensemble of all three methods (XWalk+BM25+NIR) substantially outperforms all other configurations.

We see in Table 2 that BM25 is significantly weaker in performance compared to NIR and XWalk and does not always improve the overall results of NIR and XWalk, especially for MAP. While BM25 can improve recall by adding listings that were not retrieved by NIR and XWalk, its poor ranking drags down MAP in the hybrid systems. For the most popular short queries, BM25 is not able to distinguish between the many listings with titles that token match similarly to the query. Whereas methods like XWalk and NIR are able to provide a more reliable ranking of the highly purchaseable listings based on training data.

However, in Table 3 we see that XWalk is complementary to BM25; XWalk is stronger in the head and torso bins while BM25 outperforms XWalk in the tail bin. This is due to the fact that XWalk suffers from cold start problems: it performs best with many prior examples and is unable to handle novel queries. BM25, as a lexical matching system, is more able to handle novel queries. Furthermore, XWalk+BM25 is still yet complementary with NIR. The semantic matching of dense retrieval excels in the tail, where queries are typically longer. When all three systems are ensembled, it is the highest performing across all query bins. XWalk's success in the head query bin is particularly notable – in an e-commerce setting, the head query bin is responsible for a large majority of merchandise sales.

	r@100	r@1000	M@100	M@1000
BM25	0.192	0.394	0.034	0.035
NIR	0.547	0.740	0.107	0.109
XWalk	0.600	0.723	0.153	0.154
NIR+BM25	0.497	0.780	0.097	0.100
XWalk+BM25	0.599	0.829	0.129	0.132
XWalk+BM25+NIR	**0.701**	**0.915**	**0.194**	**0.198**

Table 2
recall (r@100, r@1000) and MAP (M@100, M@1000) for retrieval models and combinations.

	tail	torso	head
BM25	0.471	0.420	0.299
NIR	0.738	0.728	0.759
XWalk	0.260	0.813	0.899
NIR+BM25	0.804	0.779	0.762
XWalk+BM25	0.595	0.875	0.914
XWalk+BM25+NIR	**0.836**	**0.931**	**0.942**

Table 3
Comparison of retrieval models in terms of recall@1000 stratified by query popularity.

5. Online Testing

We tested XWalk in a live online A/B experiment on a large e-commerce platform. The experiment ran for 23 and 25 days on our mobile and web version of our platform, respectively. Our search system is a two-stage search system, which uses an ensemble of candidate retrievers in the first pass, followed by a second pass re-ranker. In our A/B experiment, an ensemble of NIR+Solr as the candidate retrieval system was compared against an ensemble of XWalk+NIR+Solr.

We saw a statistically significant and substantial increase in conversion rate for the search system including XWalk in both the web and mobile platforms, $+1.2\%$ on web and $+1.98\%$ on mobile. In addition, in a production setting, we saw that XWalk was our lowest latency retrieval engine. The 99th percentile latency is only 58% of the NIR engine and 22% that of our Solr inverted index.

6. Conclusion

Head queries are responsible for the large majority of purchases in e-commerce. We presented XWalk, a novel candidate retrieval engine, which by frames search as a query-to-product recommendation problem, leverages powerful, highly efficient graph methods to substantially improve head query performance in product search. XWalk is also complementary to other common retrieval engines such as BM25 and dense retrieval, and ensembling produces a powerful retrieval engine.

References

[1] L. Wang, J. Lin, D. Metzler, A cascade ranking model for efficient ranked retrieval, in: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 105–114. URL: https://doi.org/10.1145/2009916.2009934. doi:10.1145/2009916.2009934.

[2] R. Baeza-Yates, A. Tiberi, Extracting semantic relations from query logs, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07, Association for Computing Machinery, New York, NY, USA, 2007, p. 76–85. URL: https://doi.org/10.1145/1281192.1281204. doi:10.1145/1281192.1281204.

[3] K. Lee, M.-W. Chang, K. Toutanova, Latent Retrieval for Weakly Supervised Open Domain Question Answering, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 6086–6096. URL: https://aclanthology.org/P19-1612. doi:10.18653/v1/P19-1612.

[4] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense Passage Retrieval for Open-Domain Question Answering, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 6769–6781. URL: https://aclanthology.org/2020.emnlp-main.550. doi:10.18653/v1/2020.emnlp-main.550.

[5] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. N. Bennett, J. Ahmed, A. Overwijk, Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval, in: International Conference on Learning Representations, 2021. URL: https://openreview.net/forum?id=zeFrfgyZln.

[6] T. Chen, M. Zhang, J. Lu, M. Bendersky, M. Najork, Out-of-Domain Semantics to the Rescue! Zero-Shot Hybrid Retrieval Models, in: M. Hagen, S. Verberne, C. Macdonald, C. Seifert, K. Balog, K. Nørvåg, V. Setty (Eds.), Advances in Information Retrieval, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2022, pp. 95–110. doi:10.1007/978-3-030-99736-6_7.

[7] S. Wang, S. Zhuang, G. Zuccon, BERT-based Dense Retrievers Require Interpolation with BM25 for Effective Passage Retrieval, in: Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 317–324. URL: https://doi.org/10.1145/3471158.3472233. doi:10.1145/3471158.3472233.

[8] S. Jiang, Y. Hu, C. Kang, T. Daly, D. Yin, Y. Chang, C. Zhai, Learning Query and Document Relevance from a Web-scale Click Graph, in: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR

’16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 185–194. URL: https://doi.org/10.1145/2911451.2911531. doi:10.1145/2911451.2911531.

[9] Y. Zhang, D. Wang, Y. Zhang, Neural IR Meets Graph Embedding: A Ranking Model for Product Search, in: The World Wide Web Conference, WWW ’19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 2390–2400. URL: https://doi.org/10.1145/3308558.3313468. doi:10.1145/3308558.3313468.

[10] X. Li, M. de Rijke, Y. Liu, J. Mao, W. Ma, M. Zhang, S. Ma, Learning Better Representations for Neural Information Retrieval with Graph Information, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, ACM, Virtual Event Ireland, 2020, pp. 795–804. URL: https://dl.acm.org/doi/10.1145/3340531.3411957. doi:10.1145/3340531.3411957.

[11] H. Zamani, W. B. Croft, Learning a Joint Search and Recommendation Model from User-Item Interactions, in: Proceedings of the 13th International Conference on Web Search and Data Mining, ACM, Houston TX USA, 2020, pp. 717–725. URL: https://dl.acm.org/doi/10.1145/3336191.3371818. doi:10.1145/3336191.3371818.

[12] X. Xia, S. Wang, H. Zhang, S. Wang, S. Xu, Y. Xiao, B. Long, W.-Y. Yang, SearchGCN: Powering Embedding Retrieval by Graph Convolution Networks for E-Commerce Search, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 2633–2634. URL: https://doi.org/10.1145/3404835.3464927. doi:10.1145/3404835.3464927.

[13] K. Zhao, Y. Zheng, T. Zhuang, X. Li, X. Zeng, Joint Learning of E-commerce Search and Recommendation with a Unified Graph Neural Network, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, ACM, Virtual Event AZ USA, 2022, pp. 1461–1469. URL: https://dl.acm.org/doi/10.1145/3488560.3498414. doi:10.1145/3488560.3498414.

[14] H. Park, J. Jung, U. Kang, A comparative study of matrix factorization and random walk with restart in recommender systems, in: 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 756–765. doi:10.1109/BigData.2017.8257991.

[15] F. Christoffel, B. Paudel, C. Newell, A. Bernstein, Blockbusters and Wallflowers: Accurate, Diverse, and Scalable Recommendations with Random Walks, in: Proceedings of the 9th ACM Conference on Recommender Systems, RecSys ’15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 163–170. URL: https://doi.org/10.1145/2792838.2800180. doi:10.1145/2792838.2800180.

[16] C. Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich, 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, WWW ’18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, pp. 1775–1784. URL: https://doi.org/10.1145/3178876.3186183. doi:10.1145/3178876.3186183.

[17] B. Paudel, F. Christoffel, C. Newell, A. Bernstein, Updatable, Accurate, Diverse, and Scalable Recommendations for Interactive Applications, ACM Trans. Interact. Intell. Syst. 7 (2016) 1:1–1:34. URL: https://doi.org/10.1145/2955101. doi:10.1145/2955101.

[18] K. Yang, X. Ma, S. Thirumuruganathan, K. Chen, Y. Wu, Random Walks on Huge Graphs

at Cache Efficiency, in: Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles CD-ROM, ACM, Virtual Event Germany, 2021, pp. 311–326. URL: https://dl.acm.org/doi/10.1145/3477132.3483575. doi:10.1145/3477132.3483575.

[19] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, R. Nogueira, Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations, in: Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021), 2021, pp. 2356–2362.

[20] P. Nigam, Y. Song, V. Mohan, V. Lakshman, W. Ding, A. Shingavi, C. H. Teo, H. Gu, B. Yin, Semantic product search, CoRR abs/1907.00937 (2019). URL: http://arxiv.org/abs/1907.00937. arXiv:1907.00937.