Augmenting Training Data for Massive Semantic Matching Models in Low-Traffic E-commerce Stores

Ashutosh Joshi[1] Shankar Vishwanath[1] Choon Hui Teo[1]
Vaclav Petricek[1] Vishy Vishwanathan[1] Rahul Bhagat[1] Jonathan May[1,2]
[1]Amazon.com, Inc., [2]Information Sciences Institute, University of Southern California
{jasutos, shavis, choonhui, petricek, vishy, rbhagat}@amazon.com
jonmay@isi.edu

Abstract

Extreme multi-label classification (XMC) systems have been successfully applied in e-commerce (Shen et al., 2020; Dahiya et al., 2021) for retrieving products based on customer behavior. Such systems require large amounts of customer behavior data (e.g. queries, clicks, purchases) for training. However, behavioral data is limited in low-traffic e-commerce stores, impacting performance of these systems. In this paper, we present a technique that augments behavioral training data via query reformulation. We use the Aggregated Label eXtreme Multi-label Classification (AL-XMC) system (Shen et al., 2020) as an example semantic matching model and show via crowd-sourced human judgments that, when the training data is augmented through query reformulations, the quality of AL-XMC improves over a baseline that does not use query reformulation. We also show in online A/B tests that our method significantly improves business metrics for the AL-XMC model.

1 Introduction

E-commerce search engines are primarily keyword-based information retrieval (IR) systems comprising two main operations–matching and ranking (Manning et al., 2008). Lexical and/or semantic matching algorithms generate a recall-focused *matchset* which is then ranked based on the match quality of the document (product) to the query (Joachims et al., 2007). Lexical matching algorithms such as Okapi-BM25 (Robertson and Walker, 1994; Robertson and Zaragoza, 2009) score a query-product pair as a weighted sum of overlapping keywords. These approaches, used in many retrieval tasks (Lee et al., 2019; Boytsov and Nyberg, 2020), do not capture customer behavior signals (purchase, stream, etc) and thus do not capture customer preferences.

Semantic matching learns representations of queries and products based on customer behavior and hence captures the products that customers prefer. Semantic matching can be implemented using dual encoders (Nigam et al., 2019; Huang et al., 2013), that separately build query and product representations, then combine the two in a final shared space to determine the similarity of the pair. It is also implemented using extreme multi-label classification (XMC) systems. In particular, Aggregated Label eXtreme Multi-label Classification (AL-XMC) (Shen et al., 2020) partitions the label (product) space by clustering labels into hierarchically granular clusters in a b-ary tree structure. Irrespective of the approach, semantic matching requires a large amount of behavioral data to train. But in newly launched sites or for new products, this behavioral data is not abundant.

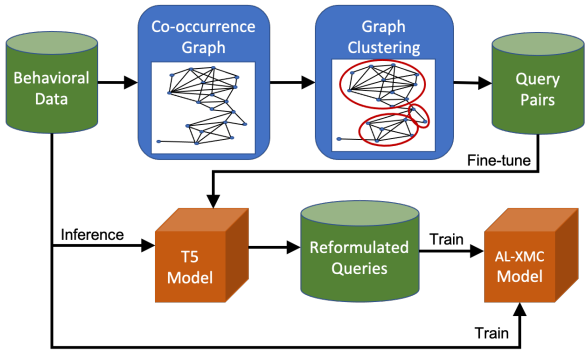

Figure 1: Overview of our approach. We induce query reformulation pairs from behavioral training data to fine-tune a reformulation model (T5). We then augment the original behavioral data with reformulated queries to train an AL-XMC semantic model.

In this paper, we propose a method to augment the data available to train semantic models in low-resource e-commerce stores. We use item-to-item collaborative filtering to identify queries that show strong behavioral associations. Such query pairs elicit a similar behavioral response from customers and so represent the same purchase intent. We use these query pairs to fine-tune the text-to-text-transfer-transformer (T5) language model (Raffel

Proceedings of NAACL-HLT 2022: Industry Track Papers, pages 160 - 167

et al., 2020) to generate query paraphrases, which we then use to augment the data available to train the semantic model.

We test our method on an AL-XMC semantic model, though it can be easily used to augment training data for any semantic model. We show, by offline crowd-sourced human judgments as well as online A/B tests, that our data augmentation technique generates reliable training data that improves the performance of the AL-XMC semantic model.

2 Approach

A pipeline diagram of our system is shown in Figure 1. The main goal of this work is to increase good quality query-product pairs with which to train an AL-XMC model; we do this by generating alternative queries from known queries using a fine-tuned T5 model. To fine-tune the T5 model, we identify purchase-intent-preserving query pairs from historical customer data. In Section 3 we describe our approach to constructing the data used to fine-tune the T5 model.

3 Constructing query reformulations

We regard customer interaction data as tuples of the form $\langle q, t, p \rangle$, each consisting of a query q, interaction type t (eg: search, purchase, etc.) and product p. It is generally the case that newer stores do not have sufficient interaction data to train robust semantic models. To increase the data available to train semantic models, we can either increase the number of queries associated with the product, the number of products associated with the queries, or both. In a typical e-commerce site the number of products is usually fixed. It is not usually possible or desirable to artificially augment this set. Instead we fine-tune a language model to generate *query* reformulations and associate the reformulated queries to products using the same interaction type t as the original query. The reformulated queries increase the variance of the tokens, including those from rare queries, in accordance with their distribution in the target store. In order for this to happen, the reformulated queries need to encompass the same purchase intent as the original queries.

For a language model to create high-fidelity reformulations, we need an adequate corpus of intent-preserving query paraphrase pairs. However, generating such a corpus is not trivial. Queries vary widely in specificity, from highly generic (*gifts for teens*) to highly specific (*HP 63XL*). Also, a product can satisfy multiple shopping intents. For example, the same product can be purchased for *kids laptop* and *cheap laptop*. In this section, we describe our methodology to generate high fidelity query reformulations.

fish tank brushes
fish filter cleaning brush
filter tube brush
slson aquarium filter brush
aquarium tube cleaning brush
fish tank cleaner brush

(a) Similarities of *aquarium filter brush*

cat tag personalized
gotags dog tags
heart pet tag
pet tags engraved cat
cat name tag
cat id tags personalized

(b) Similarities of *heart name tag for dog*

Figure 2: Top 6 similarities of two example queries, sorted by PMI. While all the top matches to *aquarium filter brush* appear to refer to the same product as the key, many of the top matches to *heart name tag for dog* refer to products that are not good matches.

3.1 Identifying related queries using Collaborative Filtering

To generate fine-tuning data for a query reformulation model, we need to identify query pairs that have the same purchase intent. We adapt item-to-item collaborative filtering (Linden et al., 2003) to generate query-to-query similarities. In this formulation, we interpret common product interactions of queries as query co-occurrence. For our case, we say two queries *co-occur* iff they *lead to the purchase of the same product*. Given this interpretation, we can then use different measures like pointwise mutual information (PMI) to quantify dependence between queries. Figures 2a and 2b show the top 6 similar queries (or *similarities*) associated with seed queries *aquarium filter brush* and *heart name tag for dog*, respectively. As we can see from these figures, the similarities are generally related to the seeds. However, as seen in Figure 2b, the similarities, while related, do not preserve purchase intent in all cases.

3.2 Clustering the query graph

Since our purpose is to identify queries that have the same purchase intent, for every seed query, we filter out any similarities with a *query specificity* that is not within 10% of the specificity of the seed.

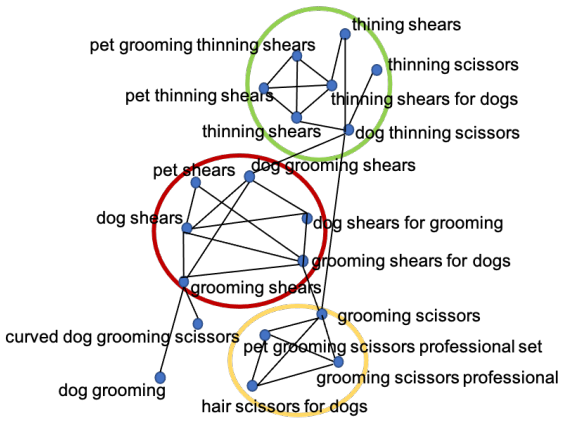

Figure 3: Queries and similarities associated with product *Lily's 8-Inch Japanese 440C Pet Dog Chunker Shears*. This sub-graph shows three clusters that can be visually identified.

We define specificity, $s_Q(q)$, as the inverse normalized click entropy ($H(q)$) of clicks for a query, q.

$$H(q) = -\sum_{p \in c(q)} P(p|q) \log P(p|q)$$

$$s_Q(q) = 1 - \frac{H(q)}{\max_{q' \in Q} H(q')}$$

where $P(p|q)$ is the probability that a customer will click on product p after issuing query q and $c(q)$ is the set of all products clicked by users after issuing q. A very specific query (e.g.: *k-cup coffee pods 72 count*) will have low $H(q)$ and thus high specificity $s_Q(q)$ while broad queries (e.g.: *party dress*) will have high $H(q)$ and low $s_Q(q)$.

Even after specificity filtering, similarities can still show associations that do not preserve query intent (Figure 2b). To further reduce noise, we identify clusters in the query similarities graph. We form this graph by considering every query to be a vertex, and an edge to connect vertices if there is a similarity relation between them. The whole graph, though extremely sparse ($< 0.0001\%$ of the edges of a fully connected graph), can still contain tens of millions of vertices and hundreds of millions of edges for some e-stores. Since our objective is to identify queries that lead to the purchase of similar products, we break the similarities graph into product sub-graphs for each product p as $G_p = (V_p, E_p)$ where $V_p = (V_{Q_p} \cup V_{N_p})$ is the set of vertices formed as follows: V_{Q_p} is the set of queries associated with (i.e. that lead to the purchase of) p, and V_{N_p} is the set of similarities of V_{Q_p}. An edge connecting (v_i, v_j) is in E_p iff $v_i, v_j \in V_p$ and there exists a similarity relation between them. By processing each product sub-graph independently, we can parallelize the clustering problem and operate only on small sub-graphs.

Figure 3 shows the sub-graph of the product *Lily's 8-Inch Japanese 440C Pet Dog Chunker Shears*. The graph naturally separates the queries into multiple sets: thinning shears, grooming shears, and professional grooming scissors, each marked in different colors in the figure.

We cluster each similarities subgraph to identify sets of queries that show high connectivity within the cluster and low connectivity outside it. The aim of clustering is to find groups of behaviorally related queries that satisfy the same customer intent. The subgraph edges already tell us that two queries are related, in that they lead to the purchase of the same product more frequently than would be expected by chance alone. By clustering we can identify groups of mutually related queries. If two queries are behaviorally related to a similar set of queries we can consider them to be related as well.

Graph clustering, though, is an ambiguous problem, with no universal definition of a cluster. Depending on the algorithm used, we can detect one, two, or three clusters in the graph in Figure 4.

Edge clustering (Ahn et al., 2010) combines features of soft clustering, where clusters can share the same members, with hierarchical clustering, where clusters are nested into dendrograms that represent progressive subdivision of a single cluster into a set of singletons. We next describe an edge clustering algorithm that is hierarchical but does not prohibit vertex sharing between clusters.

3.3 Seeding query clusters

Hierarchical clustering recursively merges clusters according to some merge criterion, beginning with each vertex in its own cluster. This imposes a restriction that clusters not share vertices. To avoid this, we let some vertices' base clusters consist of themselves *and* their neighbors (i.e. their similarities). To determine which initial clusters are singleton vertices and which contain vertices and neighbors, we use the *Clustering Coefficient (C3)*, which measures the cliquishness of the neighborhood of a vertex in a graph (Lind et al., 2005). $C3(i)$ is defined as the fraction of the number of triangles observed in the graph out of the total number of possible triangles which may appear. For a vertex i with a degree d_i, the total number of

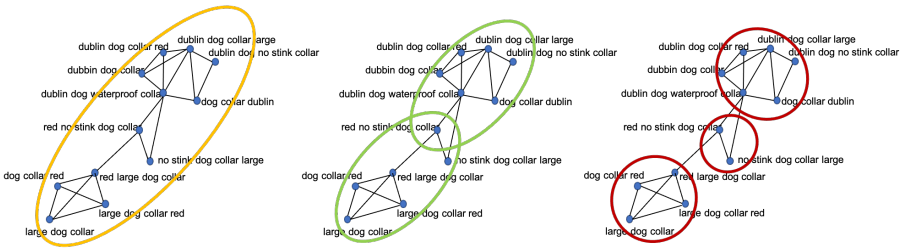

Figure 4: Different ways to cluster a graph

possible triangles is just the number of pairs of neighbors, i.e. $d_i(d_i - 1)/2$. Then $C3$ is given by:

$$C3(i) = \frac{2T_i}{d_i(d_i - 1)} \quad (1)$$

where T_i is the number of observed triangles incident on vertex i.

If a vertex has a high (> 0.33) C3 coefficient, we are assured that the neighborhood has a high internal degree (a large number of edges between vertices in the neighborhood), and so we include its neighbors in its initial cluster. Vertices not meeting this criteria are added as singleton initial clusters.

3.4 Merging query clusters

We then merge clusters hierarchically, based on the size of the vertex intersection between clusters. Starting with the seed clusters (either neighborhoods or singleton vertices), in each round, pairs of clusters are chosen in order by intersection size. A round terminates when for some cluster pair c_i, c_j to be merged, $|c_i \cap c_j| < \theta \times \min(|c_i|, |c_j|)$ for some predefined hyperparameter θ,[1] where $|c_i|$ is the number of vertices in cluster c_i.[2]

3.5 Pruning query clusters

Since we include entire neighborhoods as seed clusters, we may also end up including rogue edges that are tenuously connected to the clusters. After the final merge, we prune the vertices that have an internal-degree to external-degree ratio less than a threshold.[3] This removes noise and bolsters the community structure (high intra-cluster connectivity and low inter-cluster connectivity).

Since query graphs are behavioral, it may not be possible to get semantically distinct clusters. We observe that overly large clusters ($|c| > 10$) include queries with multiple (albeit related) purchase intents. For example, in Figure 5, though mostly semantically grouped, we see a mix of *dog halloween costumes* and *dog sweaters* in one cluster. We thus only consider clusters with fewer than 10 vertices.

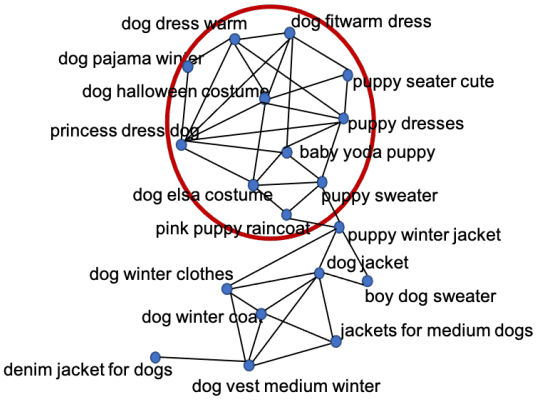

Figure 5: Due to the behavioral nature of the graph, some queries with different intents get grouped together. In this example, 'dog elsa costume' and 'dog dress warm' are grouped together, yet represent different intents.

4 Experiments

From a low resource store we collect 2.4m instances of behavioral association between queries and products. We select behaviors over a threshold level of activity (e.g. clicks and purchases), comprising 0.83m pairs, and use that to train a baseline AL-XMC model; we call this data *local*. To apply query reformulation (*QR*), we first fine-tune a pretrained T5-Base model for a sequence-to-sequence prediction task that uses query pairs from the clusters we generated in Section 3.[4] We fine-tune for two epochs using batches of 32 intent-preserving query pairs and gradient accumulation with a factor of 16, for an effective batch size of 512. We

[1] We use $\theta = 0.4$.

[2] We can interpret the round termination condition as a modified Jaccard similarity, where our condition, $\frac{|c_i \cap c_j|}{\min(|c_i|, |c_j|)} < \theta$, is more amenable to clusters of dissimilar sizes than $\frac{|c_i \cap c_j|}{|c_i \cup c_j|}$, the Jaccard coefficient.

[3] We use 0.5.

[4] We again impose the restriction that the specificity of the target query be within 10% of that of the source query.

size	accuracy
130k	79.67%
460k	86.67%
4m	90.33%
40m	93.00%

Table 1: Impact of data size on fine-tuned T5 reformulation's intent preservation, showing that accuracy generally correlates with data size. The 130k and 460k data points use only reformulated local data, while the 4m and 40m data points use data from other high-resource stores. Our QR experiments use the local-only 460k-tuned T5 model.

optimize with Adam, with a learning rate of $3e-4$. We fine-tune using 8 NVIDIA V100 GPUs; each epoch takes around 24 hours to run.

We use the fine-tuned T5 model to reformulate the query for each of the 2.4m collected query-product-behavior tuples, and associate the reformulation with the product and behavior, as if it was additional data. We can then re-select tuples with behavior over the same threshold level of activity used to form the local data; now, however, more tuples are above threshold and some new tuples have been introduced. Altogether, this procedure dramatically increases the size of the AL-XMC training corpus, to 16.14m training tuples.

In addition to the local baseline and local+QR reformulation approaches, we compare our data augmentation method to integrative knowledge transfer (IKT) (Pan et al., 2008; Zhuo et al., 2008) from a higher-resource store. To do this, we identify queries in the high-resource store that show strong interaction with a product that is available in both stores. We add those query-product associations to the training set in the target store. Such an approach is of course only possible in a mature ecosystem where previously-established high-resource stores exist.

5 Evaluation

To intrinsically evaluate the impact of fine-tuning data size on reformulation intent preservation, we randomly select 300 QR tuples and determine to what degree intent is preserved during reformulation, using a variety of data conditions to fine-tune T5. We count a reformulation as accurate if it has the same manually judged purchase intent as the input query and a search engine returns a similar set of products for both queries. Table 1 shows that for the reformulation fine-tuned on the 460k pairs obtained as described in Section 3, accuracy is about 87%. To put this in context, Table 1 also shows results for T5 reformulation when fine-tuned on a 130k subset of that data, as well as on larger query reformulation fine-tuning data sets obtained by including query pairs from other high-resource stores. The trend indicates that intent preservation is generally a function of the amount of data used to fine-tune the T5 model. In the rest of this work we use a T5 reformulation model fine-tuned on the 460k pairs obtained as described in Section 3, since the availability of high-resource store data is not guaranteed.

To extrinsically evaluate, we consider the effectiveness of the AL-XMC models when built under different data augmentation conditions. We consider both offline and online evaluation paradigms.

5.1 Offline evaluation

Typically, evaluations are done by computing precision and recall metrics using customer purchases as ground truth. Low-resource stores however, do not have sufficient purchase data and suffer from significant display bias—if a relevant product isn't shown to a customer, they cannot purchase it, resulting in an artificial drop in precision of a new (not deployed) model. Thus, we train crowd-sourced judges using the Toloka platform[5] to evaluate if the products predicted by the AL-XMC model for a particular query are indicative of the query text. We use 1,000 randomly sampled query-product pairs to evaluate the models. Table 2 gives the product accuracy improvement and coverage (average number of products generated per query) for AL-XMC models. AL-XMC models trained using any data-augmentation (local+$*$) significantly outperform the model trained using only local data, both in terms of product accuracy (p-value in a one-sided t-test is < 0.01 for all three results) and coverage. Our QR data augmentation scheme outperforms IKT (local+IKT18) when the size of data augmentation is similar. Using multiple high-resource stores and hand-tuning the parameters for IKT, we can increase the training data by almost 3x to a total size of 47.58m. However, even with this massive data increase (local+IKT48), IKT only achieves a product accuracy comparable to our augmentation scheme. If no high-resource store is available, IKT is not even an option. Query reformulation, on the other hand, only requires information from the

[5] https://toloka.yandex.com/

Training data	Product accuracy improvement	Avg number of products/query	Num training pairs
local	N/A	16.85	0.83m
local+IKT18	13.58%	22.27	18.11m
local+IKT48	19.85%	25.94	47.58m
local+QR (ours)	19.57%	21.08	16.14m

Table 2: Data augmentation method comparison. Product accuracy (proportion of relevant products) improvement measures percent increase from local baseline. Product accuracy numbers use offline crowd sourced evaluations. p-value is < 0.01 for all three data augmentation cases in a one-sided t-test.

Training data	Improvement in SCR	P(Treatment(SCR)> Control(SCR))	Improvement in CRR	P(Treatment(CRR)> Control(CRR))
local+QR	0.26%	0.82	-0.30%	0.15

Table 3: Online A/B test where control is the local model. For Search Click Rate (SCR), positive effect is better, for Customer Reformulation Rate (CRR) negative effect is better.

low-resource store and a language model like T5.

5.2 Online evaluation

For the analysis of a single A/B test, it is common to compute a p-value from the t-statistic. However, in modern industrial settings, a large number of randomized experiments are run every day. In this setting, using only the Null Hypothesis test essentially ignores the information from the whole population of experiments. Stein's paradox (Stein, 1956) states that when estimating multiple parameters, there exist combined estimators more accurate on average than any method that handles the parameters separately. In the case of A/B tests, this means that the true effect of any one experiment can benefit by including the information from prior tests. Also, while the Null Hypothesis test is proper for testing whether the true effect is below or above zero, it is inconvenient to determine the true effect with respect to a loss/utility function. For our evaluations, we use an empirical Bayesian approach for analysis of large-scale experiments proposed by Guo et al. (2020), that uses the Normal-Normal model to determine the posterior probability of a positive return on a loss metric. Adding a risk buffer, we consider our effect positive if the posterior probability of a positive return, i.e., the treatment metric being greater than the control metric, is > 0.66.

We ran a 14 day A/B test on a popular e-commerce site in a low-resource English-speaking store, using the AL-XMC model trained with purchase data augmented with query reformulations as Treatment and the purchase data alone as Control, and observed a significant improvement in business metrics—Search Click Rate (SCR, the proportion of all searches that have at least one click) and Customer Reformulation Rate (CRR, the proportion of all searches where the customer had to reissue the query with different wording). A reduction in the CRR implies that the search returned relevant products in the first query. Table 3 shows a significant increase in SCR and a significant reduction (probability of positive effect < 0.33) in CRR.

6 Discussion

We compare the generated queries to those transferred by IKT48 and find that only 20k of the reformulated queries were not present in the IKT48 set. This indicates that the T5 model is able to generate high fidelity queries that customers are likely to use. In addition, data augmentation with query reformulations is able to achieve the same results as that with IKT48 with only about a third of the data. Thus, the reformulated queries are able to capture the relevant information of the target locale. Although IKT48 contains nearly the same information as QR, it is also noisier in the sense that it transfers queries that are relevant in the source (high-resource) store but may not be relevant in the target low-resource store. This noise necessitates several times more data to achieve the same performance. Model training with the larger IKT data set takes approximately 4 times as long as training with the query reformulations dataset.

7 Related Work

Data augmentation in the e-commerce space has dealt with reformulating queries in several ways.

Kuzi et al. (2016) expand user queries using semantically similar terms according to a learned embedding space, while Wang et al. (2021) use query annotations to identify words to expand queries. Both methods rely on a large amount of natural query reformulation by users to train their models, which we avoid. Zhang et al. (2015) and Wang and Yang (2015) propose synonym or paraphrase replacement reformulation approaches. The noisiness of these approaches, some of which expect significant context, are a bad fit for the product query use case. Other works that leverage data enhancement via model-based generation include Mao et al. (2021), who augment queries in black-box question answering tasks, Sennrich et al. (2016), who generate bilingual parallel data via backtranslation, and Fadaee et al. (2017) who, similar to us, though in a machine translation context, reformulate training data via language model-based generation, in this case focusing on rare word replacement.

8 Conclusion

In this paper, we have presented a method to augment training data for semantic models in low-resource e-commerce stores. Our method is generic enough to be applied as training data augmentation for any model and does not require transfer from high-resource store data. We have shown that augmenting training data using query reformulations improves upon a baseline store-specific AL-XMC semantic matching model in both offline evaluations as well as online business metrics. Our methods increase the training data of a test low-traffic store from 0.83m to 16.14m, resulting in a quality improvement of 19.57% over the baseline model. We also see a significant boost to business metrics in online A/B tests. Next we will explore similar data augmentation techniques for generating multilingual query reformulations, using the mT5 pretrained model (Xue et al., 2021). In addition to low traffic stores, this technique may even be applied to yet-to-be-launched locales where training data is missing completely, by forming pseudo-queries from product descriptions.

9 Ethics Statement

We have performed our research so far only for the English language. Though we believe that similar results can be obtained for non-English languages we have yet to demonstrate this. We only use query and product interaction data for our work. Any identifiable user information is completely stripped before we can access the data. As our method is ultimately used to retrieve a set of products in an e-commerce store, incorrect predictions will not cause harm to the user besides an unsatisfactory experience.

References

Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. 2010. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764.

Leonid Boytsov and Eric Nyberg. 2020. Flexible retrieval with NMSLIB and FlexNeuART. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 32–43, Online. Association for Computational Linguistics.

Kunal Dahiya, Deepak Saini, Anshul Mittal, Ankush Shaw, Kushal Dave, Akshay Soni, Himanshu Jain, Sumeet Agarwal, and Manik Varma. 2021. Deepxml: A deep extreme multi-label learning framework applied to short text documents. In *Proc. 14th ACM Int'l Conf on Web Search and Data Mining*, WSDM '21, page 31–39.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

F. Richard Guo, James McQueen, and Thomas S. Richardson. 2020. Empirical bayes for large-scale randomized experiments: a spectral approach. *arXiv: Methodology*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. ACM International Conference on Information and Knowledge Management (CIKM).

Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai. 2007. Learning to rank for information retrieval (lr4ir 2007). *SIGIR Forum*, 41(2):58–62.

Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query expansion using word embeddings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 1929–1932, New York, NY, USA. Association for Computing Machinery.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Pedro G. Lind, Marta C. González, and Hans J. Herrmann. 2005. Cycles and clustering in bipartite networks. *Physical Review E*, 72(5).

G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80.

Christopher D. Manning, Prabhakar Raghavan, and Hinriche Schütze. 2008. XML retrieval. In *Introduction to Information Retrieval.* Cambridge University Press.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4089–4100, Online. Association for Computational Linguistics.

Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian (Allen) Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic product search. In *Proceedings of the 25th ACM SIGKDD*, KDD '19, page 2876–2885.

Sinno Jialin Pan, Dou Shen, Qiang Yang, and James T. Kwok. 2008. Transferring localization models across space. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, page 1383–1388. AAAI Press.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

S. E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, page 232–241, Berlin, Heidelberg. Springer-Verlag.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Yanyao Shen, Hsiang-fu Yu, Sujay Sanghavi, and Inderjit Dhillon. 2020. Extreme multi-label classification from aggregated labels. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Charles Stein. 1956. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 197–206, Berkeley, Calif. University of California Press.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, Lisbon, Portugal. Association for Computational Linguistics.

Yaxuan Wang, Hanqing Lu, Yunwen Xu, Rahul Goutam, Yiwei Song, and Bing Yin. 2021. Queen: Neural query rewriting in e-commerce. In *The Web Conference 2021, Workshop on Knowledge Management in E-Commerce*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA. MIT Press.

Hankui Zhuo, Qiang Yang, Derek Hao Hu, and Lei Li. 2008. Transferring knowledge from another domain for learning action models. In *PRICAI 2008: Trends in Artificial Intelligence*, pages 1110–1115, Berlin, Heidelberg. Springer Berlin Heidelberg.