A unified Neural Network Approach to E-Commerce Relevance Learning

Yunjiang Jiang
Yue Shang
yunjiang.jiang@jd.com
yue.shang@jd.com
JD.com Silicon Valley Research Center
Mountain View, CA, USA

Rui Li
Wen-Yun Yang
richard.rui.lee@gmail.com
wenyun.yang@jd.com
JD.com Silicon Valley Research Center
Mountain View, CA, USA

Guoyu Tang
Chaoyi Ma
tangguoyu@jd.com
machaoyi@jd.com
JD.com
Beijing, People's Republic of China

Yun Xiao
Eric Zhao
xiaoyun1@jd.com
ericzhao@jd.com
JD.com Silicon Valley Research Center
Mountain View, CA, USA

ABSTRACT

Result relevance scoring is critical to e-commerce search user experience. Traditional information retrieval methods focus on keyword matching and hand-crafted or counting-based numeric features, with limited understanding of item semantic relevance. We describe a highly-scalable feed-forward neural model to provide relevance score for (query, item) pairs, using only user query and item title as features, and both user click feedback as well as limited human ratings as labels. Several general enhancements were applied to further optimize eval/test metrics, including Siamese pairwise architecture, random batch negative co-training, and point-wise fine-tuning. We found significant improvement over GBDT baseline as well as several off-the-shelf deep-learning baselines on an independently constructed ratings dataset. The GBDT model relies on 10 times more features. We also present metrics for select subset combinations of techniques mentioned above.

ACM Reference Format:
Yunjiang Jiang, Yue Shang, Rui Li, Wen-Yun Yang, Guoyu Tang, Chaoyi Ma, Yun Xiao, and Eric Zhao. 2019. A unified Neural Network Approach to E-Commerce Relevance Learning. In *Proceedings of 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data (DLP-KDD'19).* ACM, New York, NY, USA, 6 pages.

1 INTRODUCTION

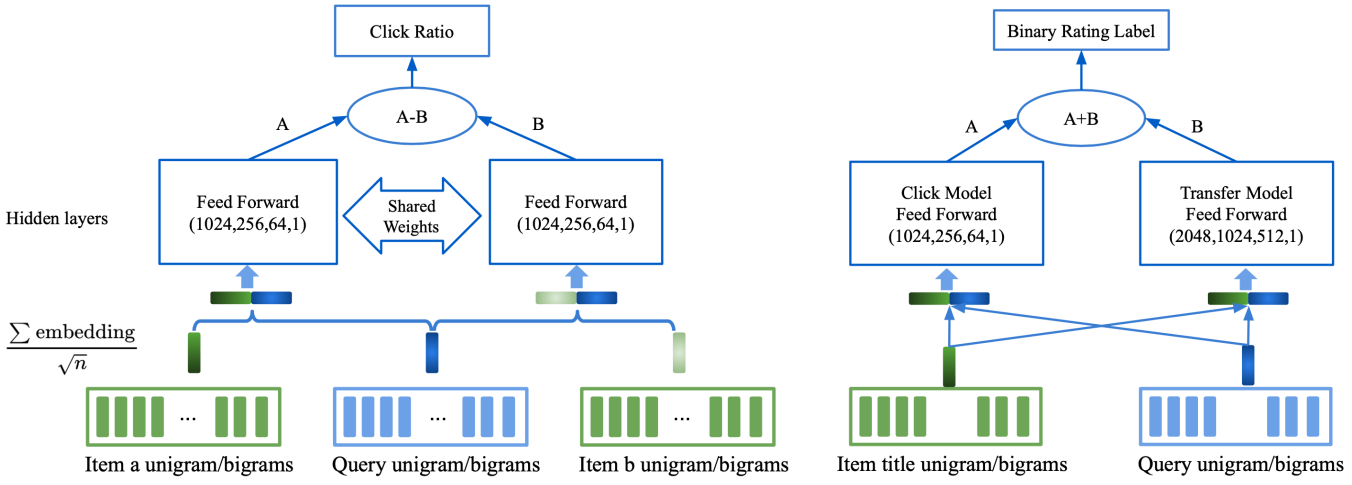
Online shopping has become ubiquitous in the modern world. A good e-commerce search engine, which helps users quickly zero in on their intended products from billions of candidates, is thus essential to users' online shopping experiences.

The most fundamental problem in e-commerce search is learning relevance between query and items. Compared with traditional web search or document retrieval, relevance learning in e-commerce search has two unique requirements.

- Unlike web documents, e-commerce items typically have little business-side textual content, mainly a short title of less than 50 characters; user comments are less reliable and typically more skewed towards older products. Thus simple keyword-based matching may not capture semantic matches effectively. We observe that even serious users enter search queries with words that do not necessarily conform to standard retail jargons. Thus, an ideal model should be able to do fuzzy match between queries and product titles.
- Due to the profit-making nature of e-commerce, personalized recommendation and other promotional logic often play a more important role than relevance in final ranking. On the other hand, irrelevant items should clearly be filtered. An ideal relevance model therefore should calibrate well with an established relevance grading, such as a binary classification for relevant and irrelevant ones or PEGFB scale ratings (Perfect, Excellent Good, Fair, Bad).

In this paper, we share our experiences of learning such an ideal relevance model for e-commerce search. We choose a deep neural network based approach as it can learn latent semantics based on raw text features. As a deep model usually requires a huge amount of data to train and human labels are expensive, we explore two sources of label supervision: 1) **user clicks**, an indirect signal for relevance (i.e., users clicks are affected by many factors such as relevance, price, and sale volumes), and 2) limited **editorial relevance ratings**. Our main contribution is to demonstrate three highly general techniques to enhance the quality of such a hybrid deep neural network, to achieve accurate relevance prediction.

DLP-KDD'19, August 5, 2019, Anchorage, AK, USA



(a) Pairwise Siamese **(b) Point-wise ensemble fine-tuning**

Figure 1: Model Architectures

1.1 Siamese Network for Pairwise Relevance

While user clicks are significantly correlated with result relevance [9, 11], click derived signals, such as CTR, are unsuitable as a point-wise learning target for relevance, especially in e-commerce, because 1) they are not well-calibrated to absolute relevance ratings, such as the widely used PEGFB scale, and 2) point-wise click labels have very skewed distribution (typically concentrated at 0), making it easy for the model to "cheat". However, clicks in aggregate are invaluable at distinguishing between good and bad results under the same query. For this reason, we choose a pairwise architecture that learns preference between a pair of items under the same query. To enable scoring single items during serving, the model takes a Siamese form [2]: the final logit for the item pair is expressed as the difference of two individual item scores from two towers of shared weights, and only one of the two identical towers is used during serving. While most applications of Siamese network in the literature are for pairwise similarity learning [6, 21], we use it for pairwise discrimination. Experiments show that this 2-tower setup actually beats the non-Siamese pairwise baseline on several key metrics (Table 1).

1.2 Efficient Training with Batch Negatives

One challenge we face in using clicks as training labels is the paucity of negative relevance examples. To ensure good user experience, top items in a query session are usually at least somewhat relevant. Thus the model rarely sees completely unrelated (query, item) pairs. This causes a serious distributional skew during serving, where we intend to score thousands of relevant and irrelevant items for a query. As a remedy, we design an efficient batch negative algorithm (Algo 1), to supplement the original n session pairs in each batch with $n(n-1)$ additional random negative (query, item) pairs. Intuitively, this gives the model a more global view of possible items for each query. Amazingly all key metrics on the original examples improve (Table 1). Case study also shows that unrelated items now score much lower under the batch negative enhancement.

1.3 Fine-Tuning with Human Supervision

Lastly, we fine-tune the model with editorial labels, in a **point-wise** fashion. This is needed for two reasons:

- clicks, though abundant, can only approximate result relevance, and are often misleading: a user can click on a result simply because of an attractive title or being part of a malicious click farm to manipulate search ranking.
- the ultimate system goal is to filter irrelevant items (those below the **Good** rating), but **pairwise** models are not well calibrated to an absolute notion of relevance; e.g., a score of 0.0 could mean Excellent or Fair depending on query, because the Siamese model is shift invariant.

Point-wise architecture at this step is conveniently adapted from a single tower of the base Siamese model. This leads to an additional 6% gain in eval AUC.

2 RELATED WORKS

In this section, we review related work on relevance learning, which aims to learn a function to determine relevance between a query and document pair. We categorize these models as traditional learning to rank based methods and deep learning based methods.

2.1 Traditional Learning to Rank Methods

Due to the importance of search engines, using model machine learning techniques to improve relevance ranking, often referred as learning to rank, has been an active research topic in the past few decades. A lot of progresses have been made, including point-wise models like RankNet [3], pairwise models like RankSVM [10] and GBRank [23]), and list-wise models like AdaRank [19] and LambdaMart [4]); see [12] for a complete survey. Here we discuss some key results that our work builds upon. First is using pairwise relevance preference instead of absolute relevance grade as the learning target introduced in [10, 23]; this is the basis of the Siamese pairwise

architecture. We do not use a list-wise approach for optimizing ranking results globally, because unlike web search, relevance is only a signal in final ranking and is mainly used as filtering in e-commerce search. Second is exploiting easy-to-get but noisy click logs instead of expensive but accurate human labels to learn relevance [10]. Our model actually leverages both labels. All the traditional methods rely on manually defined features, such as BM25, matching positions and page-rank, and cannot effectively utilize raw text features. Thus, they often fail to evaluate *semantic* relevance if a document does not contain exact terms in a query.

2.2 Neural Network Based Ranking Methods

In the past few years, a large number of embedding-based neural network models have been successfully applied to learn semantic relevance between queries and documents; [14] provides an excellent survey. These works are roughly divided into two camps: using neural network induced embeddings as features in a 2-stage process, or directly in an end-to-end manner.

In the first category, [18] learns embeddings as features for a final ranking objective. [22] uses the embeddings to promote result diversity. The resulting embeddings are amenable to fast online retrieval and similarity analysis. [1] in addition learns user embeddings for personalized search.

In the second category, DSSM [9] and their followup work CDSSM [17] pioneered the application of deep neural networks in end-to-end relevance learning. Further, new models, such as DRMM [8], Duet [15], DeepRank [16] have been proposed to improve content based models via exploring traditional IR lexical matching signals (e.g., query terms importance, exact matching) in neural networks. This is the approach that we follow. Also we keep the user dimension out of the equation to stay focused on pure relevance learning.

DSSM uses two separate feed-forward towers, also referred to as Siamese network, to encode query and result as embeddings, and optimizes a softmax loss based on the cosine distance of those embeddings. Our notion of Siamese network is different. Each of our training examples consists of a query and a pair of items, where the model allows interaction between the query and either item, but no interaction between the two items (see Figure 1a). We note that, our model can be easily extended to include those new signals beyond just query and title, or new architectures (e.g., RNN, Transformer) in each tower of the Siamese network. Our main contributions are three general techniques: 1) Siamese network for learning pairwise preference, 2) efficient training with batch negatives, and 3) pointwise fine-tuning via human labels.

3 MODEL DESCRIPTION

Much of the relevance ranking literature takes a single query and multiple documents as input, such as DSSM and its variants. This is well suited for the ranking problem. Our ultimate use case, however, is to filter out irrelevant documents, based on a score and a global threshold. In other words, we learn a relevance classification instead of a ranking model.

For simplicity and portability, we use title text as the only item side feature. In practice, numeric features can be sparse, especially for tail queries, whereas item titles are almost always available, and can help better generalize from torso queries. Our model is first trained on user clicks, then fine-tuned using human labels.

3.1 Siamese Pairwise Network

Due to the abundance of clicks in search log, compared to other user signals such as purchase or placement into shopping cart, we choose clicks as weak labels in the initial model warm-up step.

Since clicks are most informative of relative relevance, we first learn preference between a pair of items under each query: each training example consists of the triple (Q, item_a, item_b), where Q stands for query, and item_a and item_b are two co-occurring items under Q, swapped randomly. We call such triples **session pairs**.

The goal of the model is to predict the click ratio of item_a, as a proxy for its relative relevance, that is, we train a classifier F according to

$$F(Q, \text{item}_\text{a}, \text{item}_\text{b}) \sim \frac{\text{click\_count}_\text{a}}{(\text{click\_count}_\text{a} + \text{click\_count}_\text{b})} \tag{1}$$

The obvious simpler alternative is to instead fit a function G to the pointwise click-through rate objective

$$G(Q, \text{item}) \sim \frac{\text{click\_count}}{\text{view\_count}} \tag{2}$$

However this approach suffers badly from label imbalance in the e-commerce search setting, where the click through labels are typically very sparse and have a highly skewed asymmetric distribution. For instance, letting G be identically 0 would already achieve a reasonably good logloss or square loss, which is completely useless for relevance scoring.

By randomly swapping item_a and item_b, the function F cannot rely on any distributional prior to beat random guessing. Since the problem can be viewed as binary classification, logloss is a natural choice.

For the convenience of scoring a single item during serving, we further restrict F to the factorized Siamese form:

$$F(Q, \text{item}_\text{a}, \text{item}_\text{b}) = \text{H}(\text{Q}, \text{item}_\text{a}) - \text{H}(\text{Q}, \text{item}_\text{b}) \tag{3}$$

Taking difference seems to work reasonable well compared to other more elaborate contrastive functions. Since the click ratio labels are not necessarily in $\{0, 1\}$, hinge loss does not apply meaningfully to the within-session pair examples. However it does make sense to apply hinge loss to the batch negative sampled pairs. We found the hybrid session logistic-loss and batch negative hinge-loss approach to improve eval logistic-loss and auc significantly on the session pairs. However it did not improve key metrics during the human-labelled fine-tuning step.

The single tower function H computes the following:

(1) For both query and item, we sum up the unigram/bigram embedding vectors (of the same dimension), divided by sqrt(n), then concatenated to form a 2d tensor E of shape (n, 2d), where n is the training/eval batch size, and d is the embedding dimension.
(2) The prediction in logit space is then calculated as $\text{H}'(\text{E})$, where H' is a feed-forward neural net, consisting of 3 relu layers [1024, 256, 64, 1].

3.2 Efficient Training with Batch Negatives

The Siamese pairwise model trained using session pairs only, as described in the previous section, already exceeds DSSM on several evaluation metrics (Table 1). More careful case study, however, reveals a fundamental problem. Many items have little variation in their scores regardless of which query is used. Clearly query is an important input to any relevance model, and our goal is not to create a query-independent item quality model.

One explanation is that unlike web search, e-commerce item attractiveness often plays a much more important role than relevance in determining users' feedback. Truly irrelevant items (negative examples) are also rare in a session, since our search system has optimized its relevance for years and most of returned items are already relevant.

To address these two problems, we borrow ideas from word2vec [13], and augment the training data with $O(n^2)$ soft negative pairs, constructed within the same batch. In the simplest point-wise setting, we pair up the query of each example in the batch with items of other examples. In the pairwise case, a natural modification is to pair the query and positive item of each example with the positive items of other examples. The motivation is that since the training set is globally uniformly shuffled, the chance that two examples within a single mini-batch have the same query, or even synonymous queries, is extremely low. Thus we can assign with high confidence negative labels to each new pair.

For a, b two items co-occurring under a query q, let click(a; b, q) stand for the number of clicks item a received in the context of q and b. Further, denote by Λ the set of all such co-occurring triples (a, b, q) within a mini-batch, ordered by click(a; b, q) $\geq$ click(b; a, q), followed by alphabetic tie-breaking of the item titles.

Then the mini-batch training objective takes the form trainloss = original loss + BN loss, where

$$\text{original loss} = \sum_{(a,b,q)\in\Lambda} \lambda(a, b, q; \ell(a, b, q)) \tag{4}$$

$$\text{BN loss} = \sum_{(a,b,q),(a',b',q')\in\Lambda, q'\neq q} \lambda(a, a', q; 0) \tag{5}$$

$$\ell(a, b, q) = \frac{\text{click(a; b, q)}}{\text{click(a; b, q)} + \text{click(b; a, q)}} \tag{6}$$

$$\lambda(a, b, q; \ell) = \tau(\text{logit(a, q)} - \text{logit(b, q)}, \ell), \tag{7}$$

and

$$\tau(x, \ell) = -\ell \log \sigma(x) - (1 - \ell) \log \sigma(-x) \tag{8}$$

is the logloss function for logit $x \in \mathbb{R}$ and label $\ell \in [0, 1]$. Here $\sigma(x) := \frac{1}{1+e^{-x}}$ is the sigmoid function.

Thus, the batch negative (BN) loss is the sum of $n(n-1)$ terms, where n is the batch size, because each positive item can be paired with all positive items from the other $n-1$ examples. The relative weights between the two loss components do not seem to matter, as judged by their final eval metrics. Algorithm 1 show the implementation of computing the forward pass with $O(n^2)$ batch negative examples efficiently within the tensorflow framework.

Geometrically, with batch negatives, the query embeddings are not allowed to cluster around a single direction, since unrelated (query, item) pairs must score much lower than related pairs.

On the 6 month data, batch negative AUC reaches 0.99 within 100k steps of 128 batch size; in contrast, original example AUC takes 6 million steps to converge. That is because the batch is now dominated by $O(n^2)$ soft negative examples, which are easy to learn. The inclusion of batch negative examples has little effect on how quickly metrics converge on the original examples, making weight tuning unnecessary.

Algorithm 1: Batch negative augmented forward pass.

input : Batch size: n
input : Feed-Forward network: H'
input : Batched embedding tensors for queries, positive items, negative items: Q, I_+, I_-
output : Logits for the original and batch negative pairs
$E_+ \leftarrow \text{ColumnConcat}(Q, I_+)$
$E_- \leftarrow \text{ColumnConcat}(Q, I_-)$
$E \leftarrow \text{RowConcat}(E_+, E_-)$
foreach $1 <= i <= n$ **do**
  $I_+ \leftarrow \text{RowCyclicPermute}(I_+)$
  $E_{\text{BN}} \leftarrow \text{RowConcat}(Q, I_+)$
  $E \leftarrow \text{RowConcat}(E, E_{\text{BN}})$
end
$\vec{P} \leftarrow \text{RowEvenSplit}(H'(E), n + 1)$
original logits $\leftarrow P_1 - P_2$
BN logits $\leftarrow \text{RowConcat}(P_3 - P_1, \ldots, P_{n+1} - P_1)$
All logits $\leftarrow \text{RowConcat}(\text{original logits}, \text{BN logits})$
return All logits

3.3 Fine Tuning via Relevance Ratings Data

In the last stage, we use human labeled data to fine-tune the click model. Since rating millions of items is prohibitively expensive, it is impractical to train a model from scratch. Instead we take the click model output and its word embeddings as a starting point, and learn a new set of hidden layers from random initialization. In order to assign absolute relevance grade to each (query, item) pair, we also switch from pairwise to a point-wise architecture, with binary labels determined by whether the rating exceeds a certain threshold: Perfect/Excellent/Good ratings are considered positive; see Figure 1 (b) for the model architecture. Finally we add the new point-wise network to the original click point-wise prediction, as a form of ensembling.

The point-wise transfer learned model thus predicts the probability whether a (query, result) pair has a relevance score above Fair (2). More generally, experiments (Table 2) show that the point-wise approach is better suited for optimizing Precision/Recall AUC for both positive and negative examples.

4 EXPERIMENTS

We present two sets of experiments. The first set compares models trained on 10% of 6 month session log (about 50m examples), and 1 million unigram/bigram vocab. No human labels are used here. The second set focuses on the effect of human label transfer learning.

models \ metrics	Eval AUC	Pair Accu	Neg PR AUC	PR AUC	NDCG@10	Mean Avg Prec	Prec@3
Click-Trained Model (part)	**0.5846**	**0.6645**	**0.6065**	**0.8316**	**0.8947**	**0.8721**	**0.7351**
w/o Negative Sampling	0.5825	0.6476	0.5968	0.8314	0.8912	0.8680	0.7299
non-Siamese Pairwise	0.5751	0.6071	-	-	-	-	-
DSSM	0.5692	0.5862	0.5288	0.6495	0.8681	0.8615	0.7114

Table 1: Click model comparison with simplified versions

models \ metrics	Eval AUC	Pair Accu	Neg PR AUC	PR AUC	NDCG@10	Mean Avg Prec	Prec@3
GBDT	-	0.6778	0.6463	0.7759	0.8995	0.8933	0.7225
Click-Trained Model (full)	0.7572	0.6822	0.4832	0.7553	0.9053	0.9054	0.7240
+Pointwise simple	0.8201	0.6678	0.6598	0.7656	0.8956	0.8942	0.7220
+Pairwise ensemble	0.7744	**0.6956**	0.6546	0.7673	**0.9077**	**0.9094**	0.7232
+Pointwise ensemble	**0.8262**	0.6921	**0.6698**	**0.7779**	0.9023	0.9003	**0.7244**

Table 2: Model compare results with GBDT and difference transfer learning configurations

	No Batch Negative Model		With Batch Negative Model	
Item title	"cellphone"	"paper cup"	"cellphone"	"paper cup"
荣耀10 64GB 渐变蓝双摄像头双卡双待4G全面屏手机 (Honor 10 64GB Blue Dual Camera Dual Sim Full Display)	1.6292	1.2842	4.6584	-8,2954
Apple iPhone X 64GB 深空灰色 (Apple iPhone X 64GB Space Grey)	1.5081	1.1767	4.8843	-7.1850
荣耀9i 4GB+64GB 幻夜黑4G全面屏手机双卡双待 (Honor 9i 4GB+64GB Black 4G Full Display Dual SIM)	1.5014	1.0814	4.7038	-9.0472

Table 3: Siamese pairwise click model with/without batch negative for the queries "cellphone" and "paper cup"

The common base model here is trained on the full 6 month of user click data, with a vocab size of about 16 million.

4.1 Experiments Setup

4.1.1 Click Data Generation. Our primary training data consists of user click-through on search session items. We perform several compression steps to strike a balance between session coverage and data preservation:

(1) From each search session, generate (query, item_a, item_b, is_clicked_a, is_clicked_b) 5-tuples, where at least one item is clicked and below the other item in displayed position.
(2) Over a period of 180 days, aggregate the click counts to obtain the 5-tuple's (query, item_a, item_b, click_cnt_a, click_cnt_b).
(3) For each query, we retain at most top 100 5-tuple's of the previous step, sorted by $\text{click\_cnt}_a + \text{click\_cnt}_b$.

The click data is randomly shuffled by query, and we use 90% for training and 10% for eval.

4.1.2 Rating Data Generation. We collect human labelled data totaling about 300k (query, item) pairs. On average, each query has about 5 items. The queries are shuffled at random and split into 65% training, 30% eval and 5% test, from which we construct the pointwise datasets. The training and eval data sets are used for transfer learning. All the experiments are evaluated on test set.

4.2 Evaluation Metrics

Besides the standard IR metrics NDCG@10, MAP, and Precision@3, we calculate pairwise accuracy (Pair Accu) for two items under the same query and Precision/recall AUC (PR-AUC) for both positive and negative labels. PR-AUC is more discriminating than ROC-AUC for tasks with a large skew in the class distribution [7]. In addition, it's a natural generalization of classification accuracy, which we care about for the filtering task. All experiments are evaluated on a 13k (query, item) pair test set.

4.3 Experiment Results

According to the previous discussion, we proposed three general techniques for deep relevance training, including Siamese structure for pairwise relevance learning, efficient batch negative training, and model fine tuning through human labeled point-wise data. To study the effect of these techniques, we compare our model with several simpler versions.

The first set of experiments aim to evaluate the techniques proposed for training a model based on click data only. Particularly, we evaluate the following methods.

- **Base Click Model (part)** is the one trained on 10% dataset, with both Siamese and batch negative techniques.
- **w/o Siamese** is a model without the Siamese structure, which takes a pairwise preference example (Q, item_a, item_b) as input, and outputs their relevance preference. As the model

is trained to score a pairwise preference example, it's hard to evaluate point-wise relevance metrics(e.g., PR AUC) and we just report eval AUC and pairwise accuracy.
- **w/o Batch Negative** is our base click model without batch negative examples.
- **DSSM** is a classical deep matching model for web search [9]. DSSM could be regarded as a simplified version of our model without the Siamese structure and batch negatives.

By comparing the results shown in Table-1, the base click model boosts the performance on all metrics (outperforming DSSM by 13% for pairwise accuracy, 28% for precision-recall AUC, 13% for negative PR-AUC and 3% for NDCG), showing that our proposed general techniques facilitate the learning of relevance from click data and decrease the impact of distributional mismatches.

In the second set of experiments, we compare the proposed transfer learning technique with several variants and baselines.

- **GBDT** (Gradient Boosted Decision Tree) is one of the most popular non-DL based models used in relevance learning ; see [20] for detail. This is our in-house ranking model baseline, implemented with Xgboost [5], with 60 features covering a variety of user feedback signals, hand-picked matching features, including BM25, window size, etc. This is optimized for years and is therefore a strong baseline.
- **Base Click Model (full)** is trained on the whole dataset, with batch negative and Siamese structure, but without transfer learning. We take it as another baseline to highlight the effect of transfer learning.
- **+Pointwise ensemble** is our propose method.
- **+Pairwise ensemble** is similar to our proposed approach but uses pairwise human label training target.
- **+Pointwise simple** is similar to the proposed method but without the base click network.

As shown in Table 2, the transfer learning models outperform GBDT and the base click model in all the metrics.

Ensembling improves the point-wise transfer learning model on all metrics. Pairwise ensemble on the other hand does the best on pairwise accuracy on the validation set, as expected. However, since PR-AUC and negative PR-AUC relate directly to the way we apply the model online, namely filtering of the top 100 results, transfer learning with the point-wise ensemble is our choice (outperforming GBDT by 3.6% on negative PR-AUC).

4.4 Case Studies

To illustrate the power of batch negative co-training, we look at example queries and items and how they score under the models trained with and without batch negatives (vanilla model), both under the Siamese pairwise architecture.

We use the query "cellphone" and the top 3 results returned by our commercial search engine, two of which are Huawei phones and the third one is an Apple iPhone X. We then change the query to "paper cup", which is something completely unrelated and chosen randomly.

Under the session-pairs only model (first 2 columns in Table 3), the three smart phones do receive higher scores under "cellphone" than under "paper cup", but the score ranges are very similar. In contrast, the batch-negative enhanced model produces 10 times greater score separation under the two queries, without significantly dilating the cluster score range itself. Thus the latter model exhibits much greater confidence in scoring and penalizes comically unrelated (query, item) pairs, in accordance with intuition.

5 CONCLUSION

We present a complete description of a 2-stage neural net based relevance scoring system. On an independent test set our model, using only a single text feature (item title), outperforms traditional IR systems based on 60 numerical features by a wide margin, and allows generalization into the long tail where only text features are available. By exploiting several generic architectural enhancements, most notably batch negative co-training, we obtain significant qualitative improvement over the base network. Though we only study feed-forward architectures here for the sake of serving scalability, these techniques can also apply to more computationally intensive ones such as CNN/RNN/attention.

REFERENCES

[1] Q. Ai, Y. Zhang, K. Bi, X. Chen, and W. B. Croft. Learning a hierarchical embedding model for personalized product search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 645–654. ACM, 2017.

[2] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[4] C. J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical report, Microsoft Research, 2010.

[5] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[6] S. Chopra, R. Hadsell, Y. LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005.

[7] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[8] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64, 2016.

[9] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.

[10] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, 2002.

[11] T. Kenter, A. Borisov, C. Van Gysel, M. Dehghani, M. de Rijke, and B. Mitra. Neural networks for information retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1403–1406, 2017.

[12] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, Mar. 2009.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

[14] B. Mitra and N. Craswell. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1):1–126, 2018.

[15] B. Mitra, F. Diaz, and N. Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1291–1299, 2017.

[16] L. Pang, Y. Lan, J. Guo, J. Xu, J. Xu, and X. Cheng. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 257–266, 2017.

[17] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 101–110, 2014.

[18] C. Van Gysel, M. de Rijke, and E. Kanoulas. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 165–174. ACM, 2016.

[19] J. Xu and H. Li. Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 391–398, 2007.

[20] D. Yin, Y. Hu, J. Tang, T. Daly, M. Zhou, H. Ouyang, J. Chen, C. Kang, H. Deng, C. Nobata, et al. Ranking relevance in yahoo search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–332, 2016.

[21] W. Yin, H. Schütze, B. Xiang, and B. Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016.

[22] J. Yu, S. Mohan, D. P. Putthividhya, and W.-K. Wong. Latent dirichlet allocation based diversified retrieval for e-commerce search. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 463–472. ACM, 2014.

[23] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 287–294, 2007.