Neural Ranking Models with Weak Supervision

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft
Or “How to beat your teacher?”
(My) Motivation

- Let’s play with different neural network architectures to make them work for solving IR problems.

- Training neural rankers using available query documents judgments. NOT working!

- NNs are general function approximates, right? Let’s just re-invent BM25.

- Hah, cool, my network learns to beat BM25 from BM25 itself!
Motivation

● Deep neural nets are data hungry
  ○ For many tasks, the more data you have, the better your model will be!

● This amount of data is not always available for many IR tasks
  ○ Unsupervised neural network based methods.
  ○ **Our idea**: Using a well established unsupervised methods as training signal.

**Weak supervision**: Connecting symbolic IR with data driven methods
General Idea

To leverage a large amounts of unsupervised data to infer “weak” labels and use that signal for learning supervised models as if we had the ground truth labels.
Weak supervision for Ranking

- Pseudo-Labeling
  - BM25 plays the role of “pseudo-labeler” in our learning scenario.
  - A target collection and a large set of training queries (without relevance judgment),
  - Using the pseudo-labeler to rank/score the documents for each query in the training query set.
Research Question

- **RQ1** Can labels from an unsupervised IR model such as BM25 be used as weak supervision signal to train an effective neural ranker?

- **RQ2** What input representation and learning objective is most suitable for learning in such a setting?

- **RQ3** Can a supervised learning model benefit from weak supervision step, especially in cases when labeled data is limited?
Ranking Architectures: Score model

- The goal in this architecture is to learn a **scoring** function

- Point-wise

- Loss: (linear regression, with MSE)

\[
\mathcal{L}(b; \theta) = \frac{1}{|b|} \sum_{i=1}^{|b|} (S(q, d)_i; \theta) - s_{q,d}_i)^2
\]
Ranking Architectures: Rank model

- The goal in this architecture is to learn a ranking function.
- Pair-wise at training/ Point-wise at inference
- Loss: (Hinge Loss)

\[
L(b; \theta) = \frac{1}{|b|} \sum_{i=1}^{|b|} \max \left\{ 0, \varepsilon - \text{sign}(s_{\{q,d_1\}_i} - s_{\{q,d_2\}_i}) \right\}
\]

\[
= (S(\{q,d_1\}_i; \theta) - S(\{q,d_2\}_i; \theta))
\]
Ranking Architectures: RankProb model

- The goal in this architecture is to learn a ranking function

- Pair-wise

- Loss: (logistic regression)

\[
P_{\{q,d_1,d_2\}_i} = \frac{s_{\{q,d_1\}_i}}{s_{\{q,d_1\}_i} + s_{\{q,d_2\}_i}}
\]

\[
\mathcal{L}(b; \theta) = -\frac{1}{|b|} \sum_{i=1}^{|b|} P_{\{q,d_1,d_2\}_i} \log(\mathcal{R}(\{q, d_1, d_2\}_i; \theta))
\]

\[
+ (1 - P_{\{q,d_1,d_2\}_i}) \log(1 - \mathcal{R}(\{q, d_1, d_2\}_i; \theta))
\]
Input Representations

- **Dense Vector Representation:**
  - Fully Featurized
  - Exactly the BM25 input:
    \[ \psi(q, d) = \left[ N ||avg(l_d) ||l_d|| \{df(t_i)||tf(t_i, d)\}_{1 \leq i \leq k} \right] \]

- **Sparse Vector Representation:**
  - Bag of words
    \[ \psi(q, d) = [tfv_c||tfv_q||tfv_d] \]
Input Representations

- **Embedding Vector Representation**
  - Joint Embedding Matrix for terms in Query and Document
    - learning representation of terms
  - Compositionally function (From words’ representation to query/document representation)
  - learning global weight of terms
Experimental Setup

- **Target data collections:**
  - ClueWeb09 CatB dataset
  - Robust dataset

- **Training Query set:**
  - AOL (after some filtering, we got more than 6m queries for each set)

- **Hyper-parameters:**
  - Width and depth of the network, learning rate, drop-out, embedding size
    - Optimized using batched GP bandits with an expected improvement acquisition function
Why do dense vector representation and sparse vector representation fail to replicate the BM25 performance, while they are not as effective as BM25 when using neural networks? How do the neural models with different training objectives and input representations compare?

### Table 2: Performance of the different models

<table>
<thead>
<tr>
<th>Method</th>
<th>Robust04</th>
<th>ClueWeb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAP</td>
<td>P@20</td>
</tr>
<tr>
<td>BM25</td>
<td>0.2503</td>
<td>0.3569</td>
</tr>
<tr>
<td>Score + Dense</td>
<td>0.1961^</td>
<td>0.2787^</td>
</tr>
<tr>
<td>Score + Sparse</td>
<td>0.2141^</td>
<td>0.3180^</td>
</tr>
<tr>
<td>Score + Embed</td>
<td>0.2423^</td>
<td>0.3501</td>
</tr>
<tr>
<td>Rank + Dense</td>
<td>0.1940^</td>
<td>0.2830^</td>
</tr>
<tr>
<td>Rank + Sparse</td>
<td>0.2213^</td>
<td>0.3216^</td>
</tr>
<tr>
<td>Rank + Embed</td>
<td>0.2811^</td>
<td>0.3773^</td>
</tr>
<tr>
<td>RankProb + Dense</td>
<td>0.2192^</td>
<td>0.2966^</td>
</tr>
<tr>
<td>RankProb + Sparse</td>
<td>0.2246^</td>
<td>0.3250^</td>
</tr>
<tr>
<td>RankProb + Embed</td>
<td>0.2837^</td>
<td>0.3802^</td>
</tr>
</tbody>
</table>

Note that although the model with dense vector representation is much faster in the inference time, some small amount of noise, which is a common phenomenon, is observed for the other evaluation metrics. Regarding the input representations, embedding vector representation not only reduces the vectors dimension by projecting them to a two-dimensional space, using t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization would lead to poor performance vectors are, the more similar the models are in terms of nDCG@20. The performance of the model, in terms of nDCG@20, increases when we use the ranking objective instead of learning to predict calibrated scores allows the model to learn latent features that would block it from going beyond the weak supervision. Using embedding vector representation not only provides a better understanding of the interactions between query and limit the ability of the models to learn latent features that would block it from going beyond the weak supervision, would not perturb the ranking as easily. Close. How are the models related? The network with more information, but also lets the network to learn imperfections from weak annotations. Although differently compared, at the 0.05 level using the paired two-tailed t-test. For the sake of visualization, we represent each model by a vector, called the performance vector, whose elements correspond to per query performance. For the sake of visualization, we represent each model by a vector, called the performance vector, whose elements correspond to per query performance.
Take Home Message:

1. Define an objective which enables your model to go beyond the **imperfection** of the weakly annotated data (ranking instead of calibrated scoring).

2. Let the network decide about the representation. Feeding the network with featurized input kills the model creativity!
Why do dense vector representation and sparse vector representation fail to replicate the performance of BM25?
Why do dense vector representation and sparse vector representation fail to replicate the performance of BM25?

Take Home Message:

Training a network with limited capacity with feature engineered input data, you more likely to overfit and lose generalization.
How meaningful are the compositionality weights learned in the embedding vector representation?

(a) Robust04  
(Pearson Correlation: 0.8243)

(b) ClueWeb  
(Pearson Correlation: 0.7014)
How meaningful are the compositionality weights learned in the embedding vector representation?

Take Home Message:

Cool…Hah?

By just seeing individual local instances from the data, the network learns such a global statistic.
How well other alternatives for the embedding and weighting functions in embedding vector representation perform?

<table>
<thead>
<tr>
<th>Embedding type</th>
<th>Robust04</th>
<th>ClueWeb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAP P@20 nDCG@20</td>
<td>MAP P@20 nDCG@20</td>
</tr>
<tr>
<td>Pretrained (external) + Uniform weighting</td>
<td>0.1656</td>
<td>0.0612</td>
</tr>
<tr>
<td>Pretrained (external) + IDF weighting</td>
<td>0.1711</td>
<td>0.0712</td>
</tr>
<tr>
<td>Pretrained (external) + Weight learning</td>
<td>0.1880</td>
<td>0.0756</td>
</tr>
<tr>
<td>Pretrained (target) + Uniform weighting</td>
<td>0.1217</td>
<td>0.0679</td>
</tr>
<tr>
<td>Pretrained (target) + IDF weighting</td>
<td>0.1402</td>
<td>0.0779</td>
</tr>
<tr>
<td>Pretrained (target) + Weight learning</td>
<td>0.1477</td>
<td>0.0816</td>
</tr>
<tr>
<td>Learned + Uniform weighting</td>
<td>0.2612</td>
<td>0.0912</td>
</tr>
<tr>
<td>Learned + IDF weighting</td>
<td>0.2676</td>
<td>0.1032</td>
</tr>
<tr>
<td>Learned + Weight learning</td>
<td>0.2837*</td>
<td>0.1387*</td>
</tr>
</tbody>
</table>

Note that in the models with pre-trained word embeddings, embedding, pre-trained embedding, and learned embedding results, regardless of the weighting approach, learning embeddings significantly performs better than random initialization when limited amount of training data is available. When enough amount of training data is fed to the networks, initializing with pre-trained embedding as initialization, with respect to the model initialized by a pre-trained embedding matrix trained on Google News and leveraging a non-linear neural network in different settings: dense sparse vector representation, RankSVM works as well as neural network, although RankSVM with linear kernel is not able to completely approximate learning as part of the model. Note that all of these results are still significantly better than RankSVM.

A reason could be learning both embedding and weight-learning from weak supervision signals for ranking. For the ClueWeb collection, the performance, it has less impact than learning embeddings. Figure 4: Strong linear correlation between weight learned and inverse document frequency.

Figure 5 presents the learning curve of the models. According to this figure, the model initialized by a pre-trained embedding trained on Google News corpus, instead of random values, converges to the same performance. An interesting observation is that the improvements over all other models are statistically significant when limited amount of training data is available. When enough amount of training data is available, the performance, it has less impact than learning embeddings. In addition to the aforementioned experiments, we have also examined different inputs: dense vector representation, the neural networks learn an embedding that is approximately 2100 times larger than the Robust04. In addition to the aforementioned experiments, we have also examined different inputs: dense vector representation, the neural networks learn an embedding that is approximately 100 times larger than the Robust04. While this is not the case for the Robust04 collection, deploying word embeddings trained on the target collection outperforms models with uniform weights. Although weight learning can significantly improve the performance, it has less impact than learning embeddings.
Take Home Message:

If you get enough data, you can learn embedding which is better fitted to your task by updating them just based on the objective of the downstream task.

But you need a lot of data: THANKS TO WEAK SUPERVISION!
Are deep neural networks a good choice for learning to rank with weak supervision?

<table>
<thead>
<tr>
<th>Method</th>
<th>Robust04</th>
<th>ClueWeb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAP</td>
<td>P@20</td>
</tr>
<tr>
<td>RankSVM + Dense</td>
<td>0.1983</td>
<td>0.2841</td>
</tr>
<tr>
<td>RankSVM + Sparse</td>
<td>0.2307</td>
<td>0.3260</td>
</tr>
<tr>
<td>RankSVM + (Pretrained (external) + IDF weighting)</td>
<td>0.1539</td>
<td>0.2121</td>
</tr>
<tr>
<td>Score (one layer with no nonlinearity) + Embed</td>
<td>0.2103</td>
<td>0.3986</td>
</tr>
</tbody>
</table>
Are deep neural networks a good choice for learning to rank?

Take Home Message:

1. Having non-linearity in neural networks does not help that much when we do not have representation learning as part of the model.

2. The most important superiority of deep neural nets, which is their ability to learn effective representations, kicks in when your network is deep enough.
How useful is learning with weak supervision as pretraining for supervised ranking?

<table>
<thead>
<tr>
<th>Method</th>
<th>Robust04</th>
<th>ClueWeb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAP</td>
<td>P@20</td>
</tr>
<tr>
<td>Weakly supervised</td>
<td>0.2837</td>
<td>0.3802</td>
</tr>
<tr>
<td>Fully supervised</td>
<td>0.1790</td>
<td>0.2863</td>
</tr>
<tr>
<td>Weakly supervised + Fully supervised</td>
<td>0.2912↑</td>
<td>0.4126↑</td>
</tr>
</tbody>
</table>
How useful is learning with weak supervision as a substitute for a small amount of supervised data? No worries, you can compensate it by pertaining your network on weakly annotated data.

**Take Home Message:**

You love neural networks but you’ve got just a small amount of supervised data? No worries, you can compensate it by pertaining your network on weakly annotated data.

<table>
<thead>
<tr>
<th>Method</th>
<th>Robust04</th>
<th>ClueWeb 09-Category A</th>
<th>ClueWeb 09-Category B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully supervised</td>
<td>0.4509</td>
<td>0.1520</td>
<td>0.2461</td>
</tr>
<tr>
<td>Weakly supervised</td>
<td>0.3077</td>
<td>0.0680</td>
<td>0.0862</td>
</tr>
<tr>
<td>RankSVM + (Pretrained (external) + IDF weighting)</td>
<td>0.1983</td>
<td>0.3802</td>
<td>0.4389</td>
</tr>
<tr>
<td>RankSVM + Dense</td>
<td>0.1790</td>
<td>0.2967</td>
<td>0.2330</td>
</tr>
<tr>
<td>RankSVM + (Pretrained (external) + IDF weighting) + Score (one layer with no nonlinearity) + Embed</td>
<td>0.2307</td>
<td>0.3260</td>
<td>0.3794</td>
</tr>
<tr>
<td>RankSVM + (Pretrained (external) + IDF weighting) + Score-Embed</td>
<td>0.2837</td>
<td>0.3802</td>
<td>0.4389</td>
</tr>
</tbody>
</table>
Four Comments on 'Neural Ranking Models with Weak Supervision'

Fernando Diaz
New York, NY
diazf@acm.org

These are four comments on a forthcoming paper at SIGIR [3].

1. WEAK SUPERVISION IN INFORMATION RETRIEVAL

Weak supervision has been used in several places before [2,4]. In particular, [4] uses BM25 to train a learning to rank model and is comparable to Rank+Dense, perhaps with better features and normalization though.

2. SCORE NORMALIZATION

The Score Model introduced in Section 4.1 attempts to predict the absolute BM25 score of each document. However, the BM25 score can vary dramatically between different initial queries and needs to be normalized before any learning is performed on top of it [5]. This is why, when retrieval scores are used in learning to rank data sets, the feature values are normalized within-query. I suspect that, if normalized, the performance of the Score Model will improve substantially.

3. COMPARISON TO PSEUDO-RELEVANCE FEEDBACK

Theoretically, this model is basically taking a large set of queries and, in the worst case, memorizing the BM25 scores. The narrow, deep architecture means that this cannot, in practice, happen, so some generalization has to occur. That is, if you increase the capacity of your model, you'll be able to perfectly reconstruct the BM25 score. The limited capacity of the architecture acts to aggressively regularize the model against this. However, we know that, as a model is more aggressively regularized, that it will redistribute weight toward correlated features [7]. The model cannot 'memorize the BM25 score' so it redistributes the weights to other terms in the top-ranked documents, essentially encoding something like an RM3 model [1]. If you were to train this on a large, external corpus, you could probably do even better by encoding the EE model [6]. Empirically, you absolutely need to show that these models are under-performing RM3, especially in light of the theoretically similarities.

4. COMPARISON BETWEEN RANK AND RANKPROB MODELS

I am guessing that the learned weights of the RankProb Model will have the symmetry you find in the Rank Model, with the only difference between in the number of learned parameters. This may explain why there does not seem to be statistical significance between Rank and RankProb Models for embedding vectors.

The only possible theoretical gain I can see from the RankProb Model is by capturing some of context of the initial retrieval. So, if, instead of $d$, you just included side information about all of the other documents in the top 2000 (e.g. mean term vector or even just a relevance-biased sampled document), you could boost performance even more. This new model sits somewhere between all three of the models in Figure 1.

REFERENCES


moustafa.dehghani.com

@md__dehghani
Four Comments on 'Neural Ranking Models with Weak Supervision'

Fernando Diaz
New York, NY
diazf@acm.org

These are four comments on a forthcoming paper at SIGIR [3].

1. WEAK SUPERVISION IN INFORMATION RETRIEVAL

Weak supervision has been used in several places before [2, 4]. In particular, [4] uses BM25 to train a learning to rank model and is comparable to Rank+Dense, perhaps with better features and normalization though.

2. SCORE NORMALIZATION

The Score Model introduced in Section 4.1 attempts to predict the absolute BM25 score of each document. However, the BM25 score can vary dramatically between different initial queries and needs to be normalized before any learning is performed on top of it [5, Figure 4]. This is why, when retrieval scores are used in learning to rank data sets, the feature values are normalized within-query. I suspect that, if normalized, the performance of the Score Model will improve substantially.

3. COMPARISON TO PSÉUDO-RELEVANCE FEEDBACK

Theoretically, this model is basically taking a large set of queries and, in the worst case, memorizing the BM25 scores. The narrow, deep architecture means that this cannot, in practice, happen, so some generalization has to occur. That is, if you increase the capacity of your model, you'll be able to perfectly reconstruct the BM25 score. The limited capacity of the architecture acts to aggressively regularize the model against this. However, we know that, as a model is more aggressively regularized, that it will redistribute weight toward correlated features [7]. The model cannot 'memorize the BM25 score' so it redistributes the weights to other terms in the top-ranked documents, essentially encoding something like an RM3 model [1]. If you were to train this on a large, external corpus, you could probably do even better by encoding the EE model [6]. Empirically, you absolutely need to show that these models are under-performing RM3, especially in light of the theoretically similarities.

4. COMPARISON BETWEEN RANK AND RANKPROB MODELS

I am guessing that the learned weights of the RankProb Model will have the symmetry you find in the Rank Model, with the only difference between in the number of learned parameters. This may explain why there does not seem to be statistical significance between Rank and RankProb Models for embedding vectors. The only possible theoretical gain I can see from the RankProb Model is by capturing some of the context of the initial retrieval. So, if, instead of $d$, you just included side information about all of the other documents in the top 2000 (e.g. mean term vector or even just a relevance-biased sampled document), you could boost performance even more. This new model sits somewhere between all three of the models in Figure 1.

REFERENCES


Pearson Correlation: 0.8460
Thank you!

mostafadehghani.com

@m__dehghani