Training Neural Rankers with Weak Supervision

Mostafa Dehghani, Hamed Zamani,
Aliaksei Severyn, Sascha Rothe,
Jaap Kamps, and W. Bruce Croft

University of Amsterdam
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.
The goal in this architecture is to learn a **scoring** function.

Point-wise

Loss: (linear regression, with MSE)

\[
\mathcal{L}(b; \theta) = \frac{1}{|b|} \sum_{i=1}^{|b|} \left( S(\{q, d\}_i; \theta) - s_{\{q,d\}_i} \right)^2
\]
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)

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

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

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

- Pair-wise

- Loss: (logistic regression)

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

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

● **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
How do the neural models with different training objectives and input representations compare?

<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>
How do the neural models with different training converge and there is no over fitting. Figure 2 illustrates the loss curves for the training and validation sets (see Section 5.1) per training step for different models. Using embedding vector representation not only provides more information, but also lets the network to learn to distinguish between examples whose scores are close. How are the models related to go beyond the weak supervision signals in the training data. Although we have tried different neural models described above, we compare their performances, especially in weak supervision scenarios as the network is more precise in terms of performance, especially in weak supervision scenarios as the network enables understanding of the interactions between query terms.

### 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!
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:

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</td>
<td>P@20</td>
</tr>
<tr>
<td>Pretrained (external) + Uniform weighting</td>
<td>0.1656</td>
<td>0.2543</td>
</tr>
<tr>
<td>Pretrained (external) + IDF weighting</td>
<td>0.1711</td>
<td>0.2755</td>
</tr>
<tr>
<td>Pretrained (external) + Weight learning</td>
<td>0.1880</td>
<td>0.2890</td>
</tr>
<tr>
<td>Pretrained (target) + Uniform weighting</td>
<td>0.1217</td>
<td>0.2009</td>
</tr>
<tr>
<td>Pretrained (target) + IDF weighting</td>
<td>0.1402</td>
<td>0.2230</td>
</tr>
<tr>
<td>Pretrained (target) + Weight learning</td>
<td>0.1477</td>
<td>0.2266</td>
</tr>
<tr>
<td>Learned + Uniform weighting</td>
<td>0.2612</td>
<td>0.3602</td>
</tr>
<tr>
<td>Learned + IDF weighting</td>
<td>0.2676</td>
<td>0.3619</td>
</tr>
<tr>
<td>Learned + Weight learning</td>
<td>0.2837*</td>
<td>0.3802*</td>
</tr>
</tbody>
</table>

(a) Robust04  
(b) ClueWeb
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!
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>

*Note: The superscript ▲ indicates statistical significance at the 0.05 level using the paired two-tailed t-test, with Bonferroni correction.*
How useful is learning with weak supervision as partially supervised training?

**Take Home Message:**

You want to train a neural network for your task but you’ve got just a small amount of supervised data?

You can compensate it by pertaining your network on weakly annotated data.
Avoiding your teacher’s mistake!

- Training a neural ranker with **controlled** weak supervision

  - **MAIN GOAL**: Controlling the effect of imperfect weak training instances by down-weighting them.
Training

- Full Supervision mode

- Weak Supervision mode

Source: https://mostafadehghani.com
Thank you!

mostafadehghani.com
@m__dehghani