APPENDIX

A BEAM SEARCH ALGORITHM FOR INFERENCE

In this section, we present the beam search algorithm for inference in detail. Given user embedding x and structure model with parameter θ, the beam search algorithm (1) picks the top B nodes at the first layer; (2) picks the top B nodes among the successors of the chosen nodes at the previous layer; (3) outputs the final B nodes at the final layer. The algorithm is shown in Algorithm 1. In each layer, choosing top B from $K \times B$ candidates has a time complexity of $O(KB \log B)$. The total complexity is $O(DKB \log B)$.

Algorithm 1 Beam search algorithm

```
**Input** user $x$, structure model with parameter $\theta$, beam size $B$.
Let $C_1 = \{c_{1,1}, \dots, c_{1,B}\}$ be top $B$ entries of $\{p(c_1|x,\theta) : c_1 \in \{1,\dots,K\}\}$.
**for** $d = 2$ to $D$ **do**
    Let $C_d = \{(c_{1,1}, \dots, c_{d,1}), \dots, (c_{1,B}, \dots, c_{d,B})\}$ be the top $B$ entries of the set of all successors
    of $C_{d-1}$ defined as follows.
        $\{p(c_1,\dots,c_{d-1}|x,\theta)p(c_d|x,c_1,\dots,c_{d-1},\theta) : (c_1,\dots,c_{d-1}) \in C_d, c_d \in \{1,\dots,K\}\}.$
**end for**
**Output** $C_D$, a set of $B$ paths.
```

B COORDINATE DESCENT ALGORITHM FOR PENALIZED PATH ASSIGNMENT

In this section, we illustrate the coordinate descent algorithm used in path assignment with penalty in detail. Recall that in penalized M-step, we want to maximize the following objective over all assignments $\{\pi_j(v)\}_{j=1}^J$'s.

$$\arg\max_{\{\pi_j(v)\}_{j=1}^J} \sum_{v=1}^{V} \left(N_v \log \left(\sum_{j=1}^{J} s[v, \pi_j(v)] \right) - \log N_v \right) - \alpha \cdot \sum_{c \in [K]^D} f(|c|). \tag{6}$$

Now we apply the coordinate descent algorithm on item v while fix the assignments of all other items. Notice that the term $-\log N_v$ is irrelevant to $\pi_j(v)$ and hence can be dropped. For each item v, the partial objective function can be written as

$$\arg\max_{\pi_1(v),\dots,\pi_J(v)} N_v \log \left(\sum_{j=1}^{J} s[v, \pi_j(v)] \right) - \alpha \sum_{j=1}^{J} f(|\pi_j(v)|), \tag{7}$$

The coordinate descent algorithm is given as Algorithm 2. In practise, three to five iterations are enough to ensure the algorithm converges. The time complexity grows linearly with vocabulary size V, multiplicity of paths J as well as number of candidate paths S.

C MORE DETAILS OF THE EXPERIMENTS

C.1 INFERENCE TIME

We present here the average inference times on Amazon books dataset for Deep Retrieval and brute-force method.

We conclude that the inference speed of DR is 4 times faster than brute-force method. Moreover, the inference time of DR grows sub-linearly as the number of items increases, whereas the inference time of brute-force methods grows linearly. So DR enjoys more advantage on time efficiency for larger datasets.

Algorithm 2 Coordinate descent algorithm for penalized path assignment

```
Input: Score functions $\log s[v, c]$. Number of iterations $T$.
Initialize $|c| = 0$ for all paths $c$.
for $t = 1$ to $T$ do
  for all items $v$ do
    $\text{sum} \leftarrow 0$.
    for $j = 1$ to $J$ do
      if $t > 1$ then
        $|\pi_j^{(t-1)}(v)| \leftarrow |\pi_j^{(t-1)}(v)| - 1$.
      end if
      for all candidate paths $c$ of item $v$ such that $c \notin \{\pi_l^{(t)}(v)\}_{l=1}^{j-1}$ do
        Compute penalized scores
          $$\tilde{s}[v, c] = \log\left(s[v, c] + \text{sum}\right) - \alpha\left(f(|c| + 1) - f(|c|)\right).$$
      end for
      $\pi_j^{(t)}(v) \leftarrow \arg\max_c \tilde{s}[v, c]$.
      $\text{sum} \leftarrow \text{sum} + s[v, \pi_j^{(t)}(v)]$.
      $|\pi_j^{(t)}(v)| \leftarrow |\pi_j^{(t)}(v)| + 1$.
    end for
  end for
end for
Output: path assignments $\{\pi_j^{(T)}(v)\}_{j=1}^{J}$.
```

C.2 Choice of number of layers D

Here we present the result for $D = 2, 3$ and 4 for Amazon books dataset.

We observe that (1) Using $D = 2$ with the same K would hurt performance; (2) Using $D = 4$ with the same K would not help. (3) Models with the same number of possible paths ($K = 1000, D = 2$ and $K = 100, D = 3$) have similar performance. However the number of parameters is of order KD^2, so a deeper model can achieve the same performance with fewer parameters. As a trade-off between model performance and memory usage, we choose $D = 3$ in all experiments.

C.3 Top path size v.s. penalty factor

Here we present the relationship between top path size and penalty factor in Amazon book dataset. As we can see, a smaller penalty factor leads to a larger path size hence heavier computation in the following reranking stage.

C.4 Precision and F-measure against hyperparameters

Here we plot the precision@200 and F-measure@200 against hyperparameters in the Amazon books datasets. The results of precision@200 are shown in Figure 3 and the results of F-measure@200 are shown in Figure 4. We can see that both the precision and the F-measure follow the same trend as the recall shown in Section 4.3.

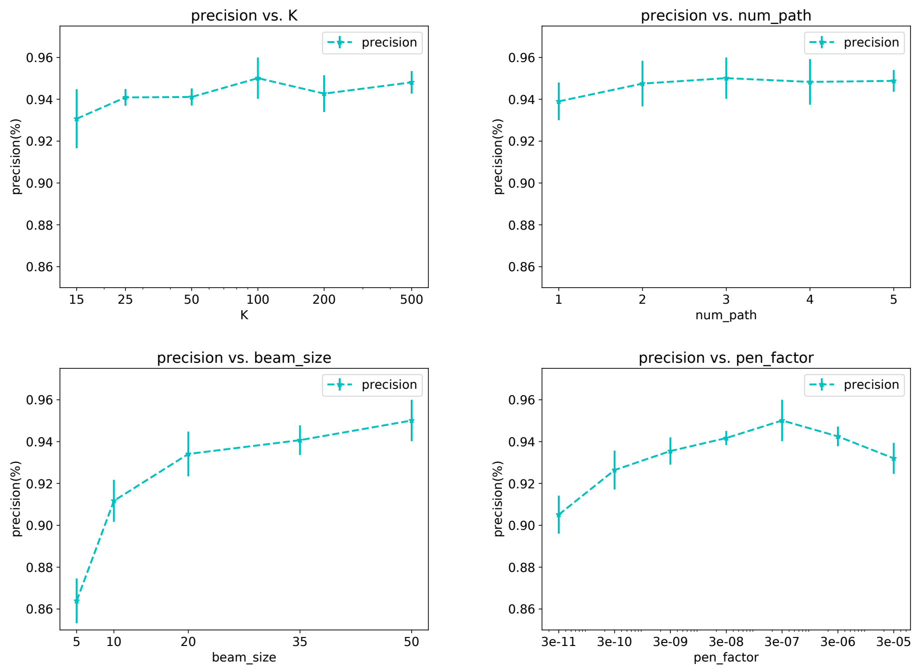

Figure 3: Relationship between precision@200 in Amazon Books experiment and model width K, number of paths J, beam size B and penalty factor α, respectively.

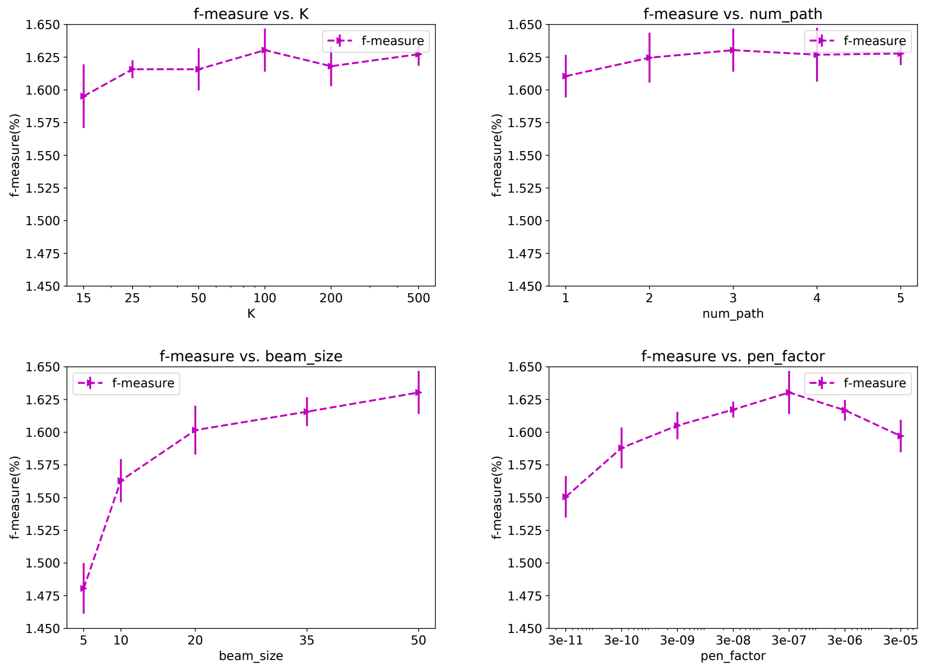

Figure 4: Relationship between F-measure@200 in Amazon Books experiment and model width K, number of paths J, beam size B and penalty factor α, respectively.

Table 3: Comparison of inference time on Amazon Books.

Deep Retrieval	0.266 ms per instance
Brute-force	1.064 ms per instance

Table 4: Comparison of performance for different model depth D.

(K, D)	Precision @ 200	Recall @ 200	F-measure @ 200
(100, 2)	0.93%	13.34%	1.59%
(100, 3)	0.95%	13.74%	1.63%
(100, 4)	0.95%	13.67%	1.63%
(1000, 2)	0.95%	13.68%	1.62%

Table 5: Relationship between the path with most items (called top path) and penalty factor α

Penalty factor	3e-10	3e-9	3e-8	3e-7	3e-6
Top path size	3837 $\pm$ 197	1948 $\pm$ 30	956 $\pm$ 29	459 $\pm$ 13	242 $\pm$ 1