**EXAMPLE TASK: WEB SEARCH**

**Focus:** Long queries (5+ words)

**PRIOR WORK**

**Non-parsing approaches (e.g. [1]):**
Fail to exploit long-range dependencies.

- **query:** otter opens clam with rock

  1st order HMM
  
  Intended meaning: “rock” = “tool used by predator”
  
  Inferred meaning: “rock” = “pearl”, because of its close proximity to “clam”

**Syntactic parsing approaches (e.g. [2]):**
Fail to exploit meaningful dependencies.

- **Issue #1:** Queries lacking verbs, prepositions, punctuation, etc. are incorrectly parsed.
  - Ex: “electrical” should have parent “fire”
  - query: electrical fire causes precautions safety

- **Issue #2:** Even correct parses fail to link terms whose interaction is meaningful for query disambiguation.
  - Ex: “rock” needs a more direct link with “otter”
  - query: otter opens clam with rock

**N-gram-based parsing approaches (e.g. [3]):**
Parses linking frequently co-occurring words are better, but don’t exploit the available direct supervision.

- **Supervision:** Human-annotated relevance scores (between 0 and 4) for many document-query pairs.
  - query: otter opens clam with rock

**scores** | **titles (possibly relevant documents)**
--- | ---
4 | Sea otter breaks open mollusk against a rock
3 | Wild otters and their use of rocks as tools
2 | Facts about the giant otter of the Amazon river
1 | Clams camouflaged on a rocky river bottom
0 | You otter investigate this really great website

---

**我们的主要贡献:**

**MEASURING SUCCESS IN IR**

The higher a relevant document appears on a list of search results for a given query, the larger the NDCG.

\[
NDCG@L = \frac{1}{Z} \prod_{i=1}^{L} 2^{v_i - 1}
\]

- \(v_i\): relevance of the \(i\)th document to the query
- \(Z\): normalization constant s.t. NDCG@L = 1 for a perfect ranking of the top L documents

**PARSING MODEL**

- \(T_w\): parse of \(w\)
- \(p_w(T_w) = \prod_{w_i \rightarrow w_j \in T_w} \theta_{w_i,w_j}\)

**Goal:** Use IR supervision to learn \(\theta\) that maximize NDCG.

**TREE EDIT DISTANCE RANKER**

There are many ways to use the dependencies of a query parse to rank documents. In this work, we use tree edit distance (TED).

\[
f(q, d) = \text{substitution} + \text{deletion} + \text{insertion}
\]

**Example costs:**

**SMOOTH NDCG-BASED OBJECTIVE**

NDCG is non-smooth, so we follow recent work [4] in defining a related but smooth objective to optimize.

\[
C_{k, h, s} = \log \left(1 + \exp \left[ f(q^{(k)}, d^{(h)}) - f(q^{(k)}, d^{(s)}) \right] \right)
\]

**Full objective:**

\[
\min_{\theta} \sum_{k=1}^{Q} \sum_{h=1}^{D(h)} \sum_{s=1}^{D(h)} C_{k, h, s}
\]

s.t. \(\theta\) are in the probability simplex

Optimization: Gradient descent on the Lagrangian dual.

---

**TRAINING ALGORITHM**

Note that for additional correlation with NDCG, as in [5], gradients are scaled by the NDCG gain of swapping documents:

\[
\gamma = \frac{2^{v_{h,s}} - 2^{-v_{h,s}}}{2}
\]

1. Initialize \(\theta\) randomly
2. while objective gradient is significant do
   1. Parse each \(w \in Q \cup D\): \(\arg\max_{\theta} p_w(T_w)\)
   2. foreach \(q \in Q, d \in D\) do
      1. Compute tree edit distance \(f(q, d)\)
   3. end
   4. end
   5. Update \(\theta\) according to \(\gamma\)-scaled gradients

---

**RESULTS FOR NDCG@10**

**Baseline (ML):** Instead of directly optimizing NDCG, the baseline uses the Viterbi Expectation-Maximization algorithm to maximize the likelihood of the parse trees.

<table>
<thead>
<tr>
<th>Query length</th>
<th># of queries</th>
<th>ML trained</th>
<th>Our method</th>
<th>Absolute improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>211</td>
<td>32.16</td>
<td>32.27</td>
<td>0.11</td>
</tr>
<tr>
<td>6</td>
<td>92</td>
<td>30.05</td>
<td>30.33</td>
<td>0.28</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>27.69</td>
<td>28.20</td>
<td>0.51†</td>
</tr>
<tr>
<td>≥8</td>
<td>56</td>
<td>24.52</td>
<td>25.18</td>
<td>0.66†</td>
</tr>
</tbody>
</table>

Superscript † indicates statistical significance (p < 0.05).