Hunter Gatherer: UDEM at 1CLICK-2

Pablo Duboue, Jing He, Jian-Yun Nie
DIRO, Université de Montréal, Canada
1CLICK-2

• Task
  – Return short text containing relevant information for queries

• Problems
  – Retrieving relevant information
  – Organizing relevant information
Our Framework

* The candidate hunter and gatherer is inspired by DeepQA framework
Candidate Hunter

Query

Corpus Index

Candidate Hunter

Whitney Houston Death

1963 January 11 actress drug death Bathtub Beverly Hilton Hotel 2012 California ...

Vital string candidates
Candidate Hunter

• Assumption
  – Relevant Information can be covered in top ranked passages

• Method
  – Passage Retrieval (Main Search)
  – Identifying Candidates
Candidate Hunter: Main Search

• Parse Query String
  – Named Entity Recognition

  Whitney Houston death → “Whitney Houston” “death”

• Build Indri Query for Passage Retrieval

  “Whitney Houston” “death” → #combine[passage120:50](#1(Whitney Houston) death)

• Retrieving Top K passages
Candidate Hunter: Identifying Candidates

• Selecting Candidates from Top K Passages
  – Terms
  – Named Entities
  – Pattern-based Candidates
    • Important attributes (birthday for a person, area for a country, etc.)
    • Information Extractor
      – Model: CRF
Our Framework

Query

Corpus Index

Candidate Hunter

Evidence Gatherer

Vital string candidates

Candidate scores

1963 January
11 actress drug death Bathtub Beverly Hilton Hotel 2012 California ...

1. Death 5817.2
2. 2012 4132.1
3. January 11 4005.6
4. Drug 3816.4
Evidence Gatherer

• Assumption
  – More evidence about query + candidate $\rightarrow$ more relevant candidate

• Method
  – Passage retrieval to gather evidences
  – Combining evidences to estimate relevance
Evidence Gatherer: Evidence Search

• Building Evidence Gathering Query
  – Original Query + Candidate
  – Candidate Types

<table>
<thead>
<tr>
<th>Text</th>
<th>Query</th>
<th>Phrase Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B</td>
<td>#1(A B)</td>
<td>named entity</td>
</tr>
<tr>
<td>A B</td>
<td>#combine(0.5 #1(A B) 0.5 #combine(A B))</td>
<td>pattern phrases</td>
</tr>
</tbody>
</table>

Whitney Houston death

Beverly Hilton Hotel

#combine[passage120:50](
  \#1(Whitney Houston)
  \#1(Beverly Hilton Hotel)
  death)
Evidence Gatherer: Evidence Search

• Heuristic Formula

\[ R(q, u) = \lambda_1 \cdot \sum_{p \in MS, u \in p} (R(q, p) + \alpha) + \lambda_2 \cdot \sum_{p \in ES} (R(q, p) + \beta) \]

Main search (MS)                        Evidence search (ES)

• Learn to Rank
  – Training Data: 60 Wikipedia articles
    • Query: title
    • Good Candidates: candidates from first passage of the Wikipedia article
    • Search Database: Clueweb09B
  – Model: GBDT
On February 11, 2012, Houston was found dead in suite 434 at the Beverly Hilton Hotel...
Summarizer: Organizing Relevant Information

• Problem
  – Fit relevant information into limited length of text

• Method
  – Maximum Marginal Relevance (MMR)
  – Integer Linear Programming (ILP)
Summarizer: MMR

- Greedy algorithm to select sentences iteratively
  \[ s^* = \arg \max_s [\lambda \cdot R(Q, s) - (1 - \lambda) \cdot \max_{s' \in S} \text{sim}(s, s')] \]

- Relevance Function
  - Score sum of candidates in a sentence

- Redundancy Function
  - Jaccard similarity of bigrams
Summarizer: ILP

• Global Optimization Problem
  – Define
    • Sentence length (vector): $l$
    • Candidate score (vector): $w$
    • Candidates contained in sentences (matrix): $M$
    • Which sentences are selected (vector): $s$
    • Which candidates are selected (vector): $e$
  – Problem
    \[
    \text{max } e^T w
    \]
    \[
    \text{s.t. } 1) l^T s \leq k; \\
    2) Ms \geq e
    \]
## Submissions

<table>
<thead>
<tr>
<th>Run</th>
<th>Hunter</th>
<th>Gatherer</th>
<th>Summarizer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1.</td>
<td>Term, NE</td>
<td>Heuristics formula</td>
<td>MMR</td>
</tr>
<tr>
<td>Run 2.</td>
<td>Same as Run1 as a mobile run</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run 3.</td>
<td>Term, NE, Pattern Info</td>
<td>Learnt scorer</td>
<td>MMR</td>
</tr>
<tr>
<td>Run 4.</td>
<td>Term, NE</td>
<td>Heuristics formula</td>
<td>ILP</td>
</tr>
</tbody>
</table>
Desktop Mandatory Results

<table>
<thead>
<tr>
<th>RUN</th>
<th>All</th>
<th>ACTOR</th>
<th>ATHLE</th>
<th>ARTIST</th>
<th>POLIT</th>
<th>FACIL</th>
<th>GEO</th>
<th>DEFIN</th>
<th>QA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1</td>
<td>0.047</td>
<td>0.040</td>
<td>0.028</td>
<td>0.039</td>
<td>0.037</td>
<td>0.060</td>
<td>0.025</td>
<td>0.066</td>
<td>0.068</td>
</tr>
<tr>
<td>Run 3</td>
<td>0.050</td>
<td>0.058</td>
<td>0.016</td>
<td>0.038</td>
<td>0.086</td>
<td>0.058</td>
<td>0.016</td>
<td>0.077</td>
<td>0.053</td>
</tr>
<tr>
<td>Run 4</td>
<td>0.080</td>
<td>0.068</td>
<td>0.084</td>
<td>0.074</td>
<td>0.025</td>
<td>0.079</td>
<td>0.062</td>
<td>0.076</td>
<td>0.146</td>
</tr>
<tr>
<td>MAX</td>
<td>0.080</td>
<td>0.068</td>
<td>0.084</td>
<td>0.074</td>
<td>0.086</td>
<td>0.083</td>
<td>0.080</td>
<td>0.088</td>
<td>0.146</td>
</tr>
<tr>
<td>MIN</td>
<td>0.047</td>
<td>0.040</td>
<td>0.016</td>
<td>0.018</td>
<td>0.025</td>
<td>0.005</td>
<td>0.016</td>
<td>0.055</td>
<td>0.053</td>
</tr>
<tr>
<td>AVRG</td>
<td>0.059</td>
<td>0.053</td>
<td>0.034</td>
<td>0.032</td>
<td>0.049</td>
<td>0.070</td>
<td>0.044</td>
<td>0.067</td>
<td>0.096</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>0.055</td>
<td>0.053</td>
<td>0.028</td>
<td>0.027</td>
<td>0.039</td>
<td>0.076</td>
<td>0.035</td>
<td>0.066</td>
<td>0.089</td>
</tr>
</tbody>
</table>

- Better performance of Run 4 (with ILP)
  → It’s important to organize relevant information intelligently

* MAX, MIN, AVRG, MEDIAN for all Desktop Mandatory Results
Desktop Mandatory Results

<table>
<thead>
<tr>
<th>RUN</th>
<th>Category</th>
<th>All</th>
<th>ACTOR</th>
<th>ATHLE</th>
<th>ARTIST</th>
<th>POLIT</th>
<th>FACIL</th>
<th>GEO</th>
<th>DEFIN</th>
<th>QA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1</td>
<td></td>
<td>0.047</td>
<td>0.040</td>
<td>0.028</td>
<td>0.039</td>
<td>0.037</td>
<td>0.060</td>
<td>0.025</td>
<td>0.066</td>
<td>0.068</td>
</tr>
<tr>
<td>Run 3</td>
<td></td>
<td>0.050</td>
<td>0.058</td>
<td>0.016</td>
<td>0.038</td>
<td>0.086</td>
<td>0.058</td>
<td>0.016</td>
<td>0.077</td>
<td>0.053</td>
</tr>
<tr>
<td>Run 4</td>
<td></td>
<td>0.080</td>
<td>0.068</td>
<td>0.084</td>
<td>0.074</td>
<td>0.025</td>
<td>0.079</td>
<td>0.062</td>
<td>0.076</td>
<td>0.146</td>
</tr>
<tr>
<td>MAX</td>
<td></td>
<td>0.080</td>
<td>0.068</td>
<td>0.084</td>
<td>0.074</td>
<td>0.086</td>
<td>0.083</td>
<td>0.080</td>
<td>0.088</td>
<td>0.146</td>
</tr>
<tr>
<td>MIN</td>
<td></td>
<td>0.047</td>
<td>0.040</td>
<td>0.016</td>
<td>0.018</td>
<td>0.025</td>
<td>0.005</td>
<td>0.016</td>
<td>0.055</td>
<td>0.053</td>
</tr>
<tr>
<td>AVRG</td>
<td></td>
<td>0.059</td>
<td>0.053</td>
<td>0.034</td>
<td>0.032</td>
<td>0.049</td>
<td>0.070</td>
<td>0.044</td>
<td>0.067</td>
<td>0.096</td>
</tr>
<tr>
<td>MEDIAN</td>
<td></td>
<td>0.055</td>
<td>0.053</td>
<td>0.028</td>
<td>0.027</td>
<td>0.039</td>
<td>0.076</td>
<td>0.035</td>
<td>0.066</td>
<td>0.089</td>
</tr>
</tbody>
</table>

- It performs well for QA queries
  → Naturally, the DeepQA framework helps QA queries
Desktop Mandatory Results

- Minor improvement from Run 3 compared to Run 1
  - Some improvement is from person type queries
Existing Problems and Future Works

• Spam
  → Filtering methods

• Candidate Selection
  → Unsupervised parsing for chunk detection

• Summarization Granularity
  – Sentences are too long
    → Sentence compression
  – Only part of sentence is relevant
    → Breaking multi-clause sentences by text simplification