SEMANTIC SEARCH WITH KEYWORD QUERIES

Master’s Thesis

Eugen Sawin

Chair of Algorithms and Data Structures
University of Freiburg
**User Query**

Films directed by Stanley Kubrick
**User Query**

Films directed by Stanley Kubrick

**Results**

- **Stanley Kubrick - IMDb**
  www.imdb.com/name/nm0000040/

- **Stanley Kubrick - Wikipedia**
  en.wikipedia.org/wiki/Stanley_Kubrick

- **Stanley Kubrick, Film Director Dies at 70**
  www.nytimes.com/.../movies/stanley-kubrick...
FULL-TEXT SEARCH

User Query

Films directed by Stanley Kubrick

Results

▸ Stanley Kubrick - IMDb
  www.imdb.com/name/nm0000040/

▸ Stanley Kubrick - Wikipedia
  en.wikipedia.org/wiki/Stanley_Kubrick

▸ Stanley Kubrick, Film Director Dies at 70
  www.nytimes.com/.../movies/stanley-kubrick...

We asked for films – got documents
SEMANTIC SEARCH

User Query

Films directed by Stanley Kubrick

Results

▶ A Clockwork Orange
▶ 2001: A Space Odyssey
▶ Dr. Strangelove or How I Learned...
Semantic Search

User Query
Films directed by Stanley Kubrick

Results
- A Clockwork Orange
- 2001: A Space Odyssey
- Dr. Strangelove or How I Learned...
User Query

Films directed by Stanley Kubrick

Results

- A Clockwork Orange
- 2001: A Space Odyssey
- Dr. Strangelove or How I Learned...

We asked for films – got film entities
Why Semantic Search?

- Over 40% of web searches are entity searches
- Focused results save time
- Suitable for machine consumption and voice output
- *(Finds results where document retrieval fails)*
Why semantic search?

- Over 40% of web searches are **entity searches**
- Focused results **save time**
- Suitable for machine consumption and **voice output**
- (Finds results where document retrieval fails)

Evolution of **intelligent search**
Why keyword queries?
Why keyword queries?

- Simple interface
- No expert knowledge required
  - Query languages
  - System-imposed limitations
- Effective for both text and voice input
- Users don’t need to adapt
**Why keyword queries?**

- Simple interface
- No expert knowledge required
  - Query languages
  - System-imposed limitations
- Effective for both **text** and **voice** input
- Users **don’t need to adapt**

Semantic search for **human beings**
**TWO-PHASE APPROACH**

**Entity Retrieval**
- User Query
- Relevant Documents
- Candidate Entities
- Answer Entities

**Deep Search**
- Answer Entities
- Semantic Answer Type
- Semantic Query
- Semantic Answer Entities
ENTITY RETRIEVAL PHASE

User Query

Relevant Documents

Candidate Entities

Answer Entities
Query Analysis

User Query → QA

Keywords → Lexical Answer Type

- **Keywords**: nouns, adjectives and verbs
- **LAT**: first non-possessive noun

Rule-based pos-tagging
DOCUMENT RETRIEVAL

User Query → QA → FTS → Keywords → Relevant Documents

- Search
- Google Custom Search API: top 10 results only

Candidate Entities → Answer Entities

Lexical Answer Type
DOCUMENT SEGMENTATION

User Query → QA → FTS → DP → Text

- Relevant Documents
- Candidate Entities
- Answer Entities
- Keywords
- Lexical Answer Type

reduce

▶ Strip HTML tags
**Entity Extraction**

- **User Query**
- **QA**
- **FTS**
- **DP**
- **NER**
- **Text**

- Extract named entity names
- Classify course type: person, location, organization, misc
ENTITY RANKING

- User Query
- Relevant Documents
- Candidate Entities
- Answer Entities

QA → FTS → NER → RK

Keywords
Lexical Answer Type
Text

Document rank, entity frequency, entity popularity
ENTITY FILTERING

- User Query
- Relevant Documents
- Candidate Entities
- Answer Entities
- QA
- FTS
- DP
- NER
- RK
- F
- Ranked Entities

- Keywords
- Text
- Lexical Answer Type

- filter

- lexical, query similarity, ontology
Deep Search Phase

Answer Entities

Semantic Answer Type

Semantic Query

Semantic Answer Entities
**Type Inference**

- **Answer Entities**
- **Semantic Answer Type**
- **Semantic Query**
- **Semantic Answer Entities**

- **Based on type frequency**
**Semantic Query Construction**

- **Semantic Answer Type**
- **Semantic Query**
- **Keywords**
- **Rule-based:**
  - $1 \text{ is-a SAT}$
  - $1 \text{ occurs-with keywords}$
**Semantic Search**

- **Answer Entities**
- **Semantic Answer Type**
- **Semantic Query**
- **Semantic Answer Entities**

- **TIK**
- **LAT**
- **QC**

- **Broccoli Search API:** all results, unfiltered
## ENTITY RETRIEVAL

### STRICT MATCHING RESULTS

<table>
<thead>
<tr>
<th>Filter</th>
<th>R (%)</th>
<th>P (%)</th>
<th>R@S (%)</th>
<th>P@S (%)</th>
<th>F@S (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ont qsim ctype</td>
<td>62</td>
<td>3</td>
<td>17</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>•</td>
<td>54</td>
<td>6</td>
<td>19</td>
<td>27</td>
<td>20</td>
</tr>
<tr>
<td>• •</td>
<td>61</td>
<td>3</td>
<td>36</td>
<td>32</td>
<td>30</td>
</tr>
<tr>
<td>• • •</td>
<td>51</td>
<td>5</td>
<td>14</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td>• • •</td>
<td>53</td>
<td>6</td>
<td>32</td>
<td>43</td>
<td>32</td>
</tr>
<tr>
<td>• • •</td>
<td>42</td>
<td>7</td>
<td>15</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td>• • •</td>
<td>56</td>
<td>6</td>
<td>35</td>
<td>38</td>
<td>32</td>
</tr>
<tr>
<td>• • •</td>
<td>48</td>
<td>8</td>
<td>30</td>
<td>44</td>
<td>31</td>
</tr>
</tbody>
</table>

Average results with ontology filter (ont), query similarity filter (qsim) and coarse type filter (ctype)
## Entity Retrieval

### Approximate Matching Results

<table>
<thead>
<tr>
<th>Filter</th>
<th>R (%)</th>
<th>P (%)</th>
<th>R@S (%)</th>
<th>P@S (%)</th>
<th>F@S (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ont</td>
<td>qsim</td>
<td>ctype</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>4</td>
<td>28</td>
<td>30</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>7</td>
<td>27</td>
<td>39</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>4</td>
<td>49</td>
<td>49</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>7</td>
<td>25</td>
<td>31</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>7</td>
<td>41</td>
<td>57</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>9</td>
<td>22</td>
<td>36</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>7</td>
<td>48</td>
<td>59</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>11</td>
<td>39</td>
<td>61</td>
<td>41</td>
<td></td>
</tr>
</tbody>
</table>

Average results with **ontology filter** (ont), **query similarity filter** (qsim) and **coarse type filter** (ctype)
## Selection Optimality

<table>
<thead>
<tr>
<th>Matching Type</th>
<th>$F@S_{opt}$ (%)</th>
<th>$R@S$ (%)</th>
<th>$P@S$ (%)</th>
<th>$F@S$ (%)</th>
<th>$Q_s$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>strict</td>
<td>45</td>
<td>43</td>
<td>34</td>
<td>34</td>
<td>78</td>
</tr>
<tr>
<td>approximate</td>
<td>65</td>
<td>59</td>
<td>49</td>
<td>48</td>
<td>71</td>
</tr>
</tbody>
</table>

Selection quality compared to the optimal selection $S_{opt}$
## Two-Phase Approach Results

<table>
<thead>
<tr>
<th>Phase</th>
<th>Matching Type</th>
<th>R (%)</th>
<th>P (%)</th>
<th>P@R (%)</th>
<th>R@S (%)</th>
<th>P@S (%)</th>
<th>F@S (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ER</td>
<td>strict</td>
<td>56</td>
<td>6</td>
<td>38</td>
<td>33</td>
<td>38</td>
<td>31</td>
</tr>
<tr>
<td>ER</td>
<td>approximate</td>
<td>78</td>
<td>7</td>
<td>56</td>
<td>47</td>
<td>60</td>
<td>46</td>
</tr>
<tr>
<td>DS</td>
<td>strict</td>
<td>44</td>
<td>9</td>
<td>24</td>
<td>20</td>
<td>22</td>
<td>19</td>
</tr>
<tr>
<td>DS</td>
<td>approximate</td>
<td>54</td>
<td>12</td>
<td>31</td>
<td>27</td>
<td>31</td>
<td>25</td>
</tr>
</tbody>
</table>

Overall results for both phases.
CONCLUSION

- **Competitive results** in entity retrieval phase
  - Simple and effective filtering
  - Near-optimal selection method
  - High noise in entity extraction

- **Unsatisfactory** deep search results
  - Unreliable semantic type detection
  - Ignored relation between entities
**Future Work**

- **Further optimize results** in entity retrieval phase
  - Add document segmentation
  - Increase number of retrieved documents
  - More robust named entity extraction
  - Enable entity linking

- **Improve** semantic query construction
  - Semantic type classification based on Freebase
  - Rule-based semantic type detection
    - "Who" → person
    - "Where" → location

Leverage existing semantic search framework
**Future Work**

- **Further optimize results** in entity retrieval phase
  - Add document segmentation
  - Increase number of retrieved documents
  - More robust named entity extraction
  - Enable entity linking

- **Improve** semantic query construction
  - Semantic type classification based on *Freebase*
  - Rule-based semantic type detection
    - “Who” → person
    - “Where” → location

Leverage existing **semantic search framework**
PYTHIA

SEMANTIC SEARCH ORACLE

Quote

“For all the things we have to learn before we can do them, we learn by doing them.” Aristotle

Repository

github.com/eamsen/pythia
**Entity Ranking**

**Overall**

**Formula**

\[
\text{score}(e) = \sum_{s \in \text{Subscores}} \frac{w_s \cdot s(e)}{s_{max}}
\]

\[s_{max} = \max_{n \in E} s(n)\]

\[\text{Subscores} = \{s_C, s_H, s_{CD}, s_{HD}\}\]

**Description**

- \(w_s\): weighting parameter
- \(s_C\): document entity freq.
- \(s_H\): snippet entity freq.
- \(s_{CD}\): documents freq.
- \(s_{HD}\): snippets freq.
**Entity Ranking**

**Subscores**

**Formula**

\[
s(e) = |\text{Occurs}(e)| \quad \text{for } s \in \{s_{CD}, s_{HD}\}
\]

\[
s(e) = \sum_{\langle \text{freq}, \text{rank} \rangle \in \text{Occurs}(e)} \frac{w_{\text{rank}} \cdot \text{freq}}{\log(cf(e) + cf_{\text{base}})} \quad \text{for } s \in \{s_C, s_H\}
\]

**Description**

- \(w_{\text{rank}}\): weighting constants
- \(cf\): corpus entity freq. (popularity)
- \(cf_{\text{base}}\): in range \([1, \infty)\)
- \(\lambda\): dampening parameter

**Formula for \(w_{\text{rank}}\)**

\[
w_{\text{rank}} = 1 - \frac{\text{rank}}{1 + \lambda \cdot \text{rank}_{\text{max}}}
\]
**Answer Selection**

**Moving Average Pivot**

**Formula**

\[ E_s = \{ e \in E_c \mid e_s \geq \delta \} \]
\[ \delta = S_{avg} + (2\gamma - 1)(S_{max} - S_{avg}) \]

**Description**

- **\( E_s \):** selection set
- **\( E_c \):** candidate set
- **\( e_s \):** entity score
- **\( \delta \):** score threshold
- **\( S_{avg_r} \):** \( \alpha \cdot e_{r-1_s} + (1 - \alpha) \cdot S_{avg_{r-1}} \) with \( S_{avg_1} = e_{1_s} \)
- **\( \alpha \):** \( \frac{2}{|E_c|+1} \)
- **\( \gamma \):** in range \([0, 1]\)
Query “inventor of the python programming language”: moving average score $S_{avg} \approx 19$, extrema $S_{min} = 3$ and $S_{max} = 83$, with $\gamma = 0.65$ we get the threshold $\delta \approx 38$. 
What is a named entity?

Example

Milky Way, Mars, Alan Turing, you

Properties

- Name
- Type
- Distinct identity
Query
<query>
  <entity_name>Daft Punk</entity_name>
  <entity_url>daftpunk.com</entity_url>
  <target_entity>organisation</target_entity>
  <narrative>
    What recording companies sell Daft Punk songs?
  </narrative>
</query>

Answer Records
virginrecords.com  1 0.98 .../wiki/Daft_Punk
somarecords.com  2 0.97 .../wiki/Daft_Punk
disney.go.com/music  3 0.89 .../wiki/Daft_Punk
## Connection to Question Answering

<table>
<thead>
<tr>
<th>Question Answering</th>
<th>Entity Retrieval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emphasis</td>
<td>query semantics</td>
</tr>
<tr>
<td>Result type</td>
<td>factoid</td>
</tr>
<tr>
<td><em>in most cases</em></td>
<td>factoids contain</td>
</tr>
</tbody>
</table>

In most cases, factoids contain entities.