Semantic Search with Keyword Queries

Master's Thesis

Eugen Sawin

Chair of Algorithms and Data Structures
University of Freiburg

Full-Text Search

User Query

Films directed by 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...

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

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...

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

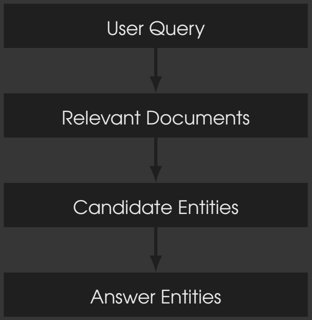
Entity Retrieval

User Query
↓
Relevant Documents
↓
Candidate Entities
↓
Answer Entities

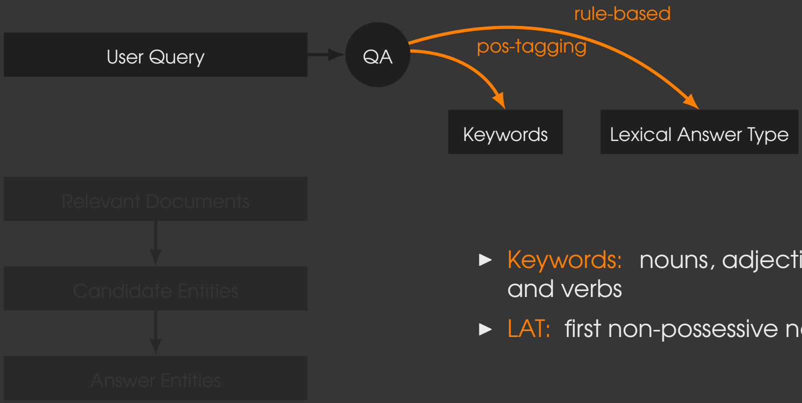
Deep Search

Answer Entities
↓
Semantic Answer Type
↓
Semantic Query
↓
Semantic Answer Entities

Entity Retrieval Phase

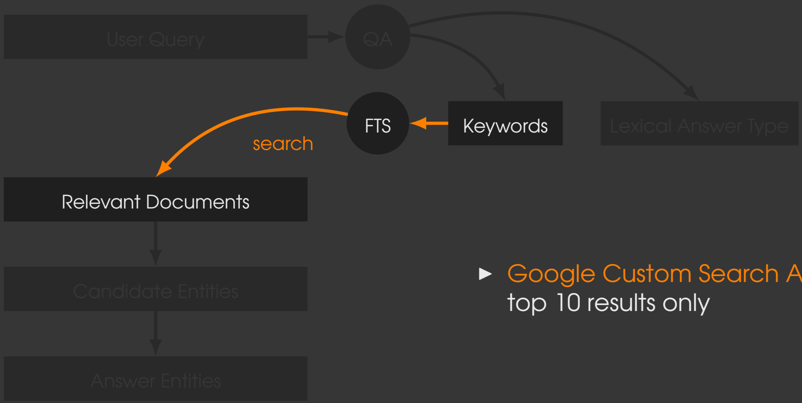


Query Analysis



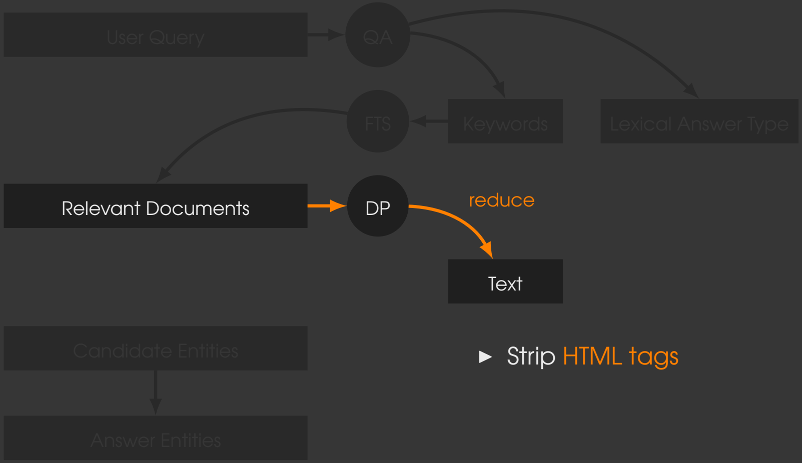
- **Keywords:** nouns, adjectives and verbs
- **LAT:** first non-possessive noun

Document Retrieval



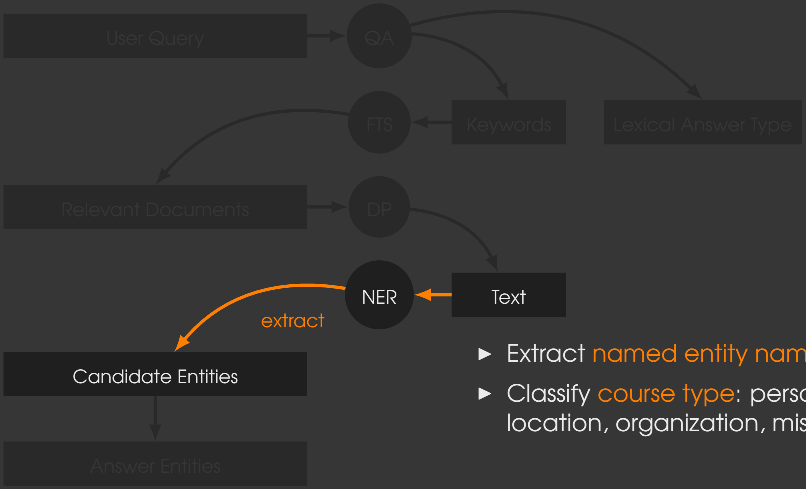
- Google Custom Search API: top 10 results only

Document Segmentation



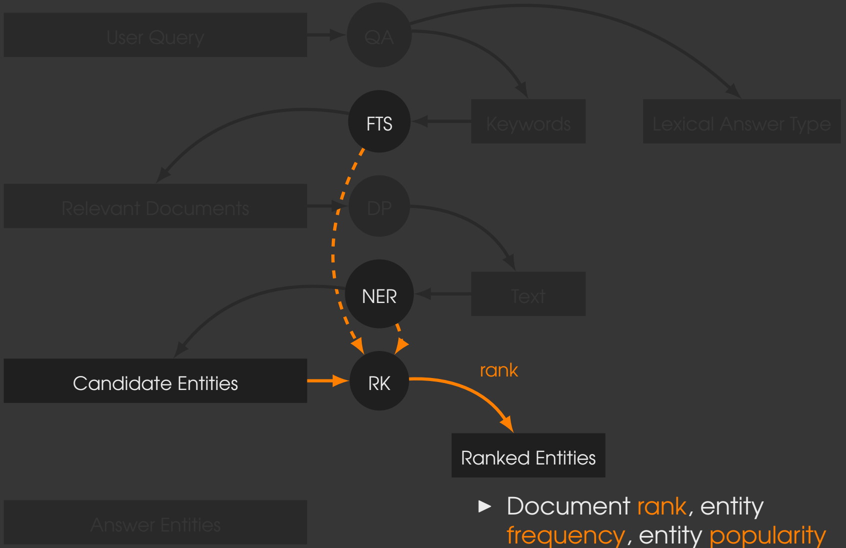
- Strip HTML tags

Entity Extraction



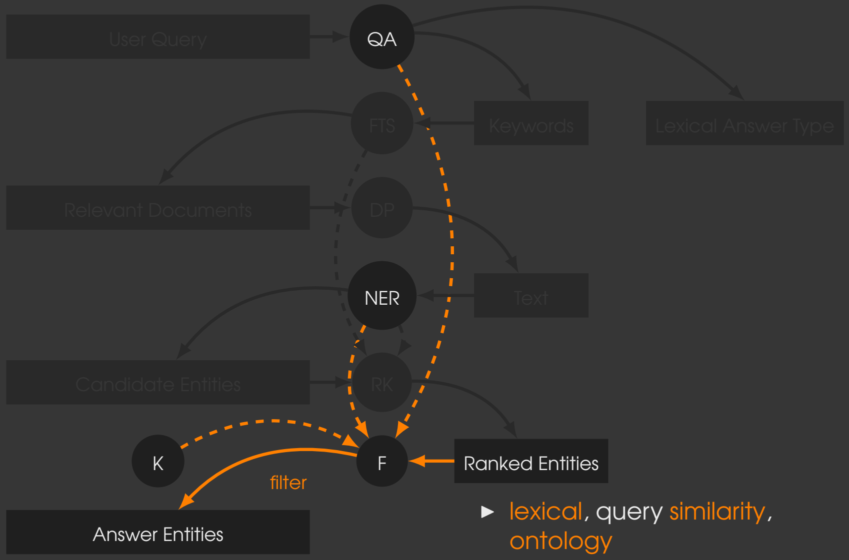
- Extract named entity names
- Classify course type: person, location, organization, misc

Entity Ranking



- Document rank, entity frequency, entity popularity

Entity Filtering



- lexical, query similarity, ontology

Deep Search Phase

Answer Entities

↓

Semantic Answer Type

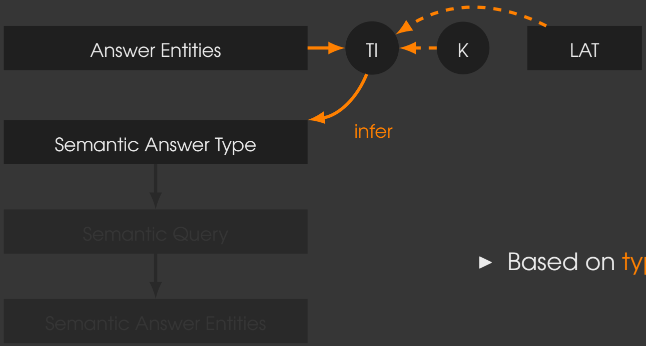
↓

Semantic Query

↓

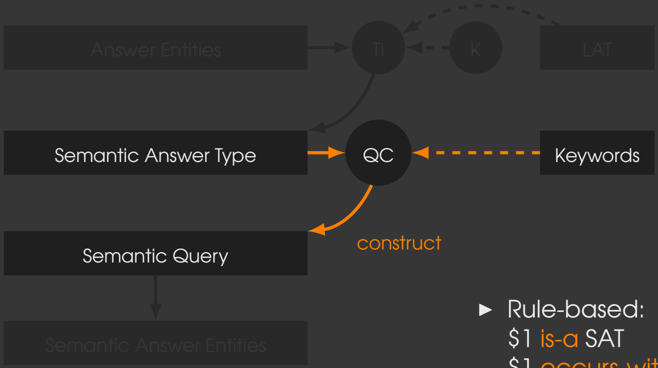
Semantic Answer Entities

Type Inference



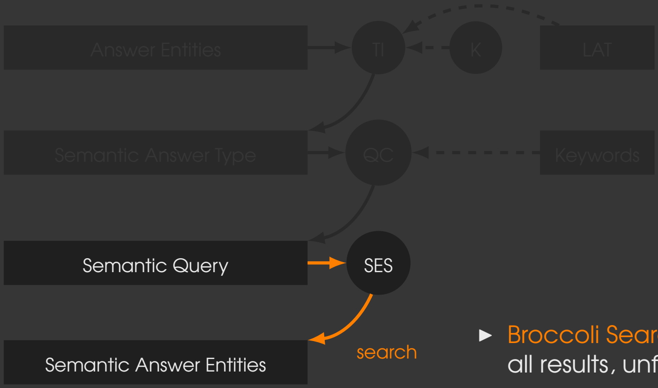
- Based on type frequency

Semantic Query Construction



- Rule-based:
  $1 is-a SAT
  $1 occurs-with keywords

SEMANTIC SEARCH



- Broccoli Search API: all results, unfiltered

Entity Retrieval

Strict Matching Results

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

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

ENTITY RETRIEVAL

APPROXIMATE MATCHING RESULTS

Filter: ont	Filter: qsim	Filter: ctype	R (%)	P (%)	R@S (%)	P@S (%)	F@S (%)
			88	4	28	30	24
•			72	7	27	39	28
	•		87	4	49	49	43
		•	76	7	25	31	23
•	•		70	7	41	57	41
•		•	57	9	22	36	24
	•	•	78	7	48	59	47
•	•	•	63	11	39	61	41

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

Selection Optimality

Matching Type	$F@S_{opt}$ (%)	$R@S$ (%)	$P@S$ (%)	$F@S$ (%)	Q_s (%)
strict	45	43	34	34	78
approximate	65	59	49	48	71

Selection quality compared to the optimal selection S_{opt}

Two-Phase Approach Results

Phase	Matching Type	*R* (%)	*P* (%)	*P@R* (%)	*R@S* (%)	*P@S* (%)	*F@S* (%)
ER	strict	56	6	38	33	38	31
ER	approximate	78	7	56	47	60	46
DS	strict	44	9	24	20	22	19
DS	approximate	54	12	31	27	31	25

Overall results for both phases.

Conclusion

- Competitve 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" $\rightarrow$ person
    - "Where" $\rightarrow$ location

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" $\rightarrow$ person
    - "Where" $\rightarrow$ 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

$$score(e) = \sum_{s \in Subscores} \frac{w_s \cdot s(e)}{s_{max}} \qquad s_{max} = \max_{n \in E} s(n)$$

$$Subscores = \{s_C, s_H, s_{CD}, s_{HD}\}$$

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

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

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

Description

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

$$w_{rank} = 1 - \frac{rank}{1 + \lambda \cdot rank_{max}}$$

Answer Selection

Moving Average Pivot

Formula

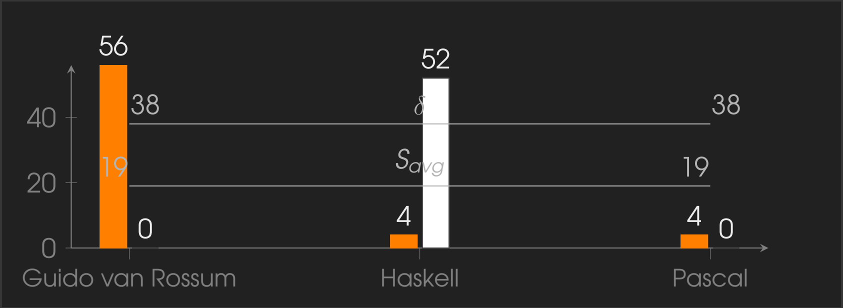
$$E_s = \{e \in E_c \mid e_s \geq \delta\} \qquad \delta = S_{avg} + (2\gamma - 1)(S_{max} - S_{avg})$$

Description

- E_s: selection set
- E_c: candidate set
- e_s: entity score
- δ: score threshold
- $S_{avg_r} = \alpha \cdot e_{r-1_s} + (1-\alpha) \cdot S_{avg_{r-1}}$ with $S_{avg_1} = e_{1_s}$
- $\alpha = \frac{2}{|E_c|+1}$
- γ: in range [0, 1]

Answer Selection

Example



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

TREC Entity Track: Related Entity Finding

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

	Question Answering	**Entity Retrieval**
Emphasis	query semantics	entity ranking
Result type	factoid	entities
in most cases	factoids contain	entities