POLITECNICO DI MILANO
Corso di Laurea in Ingegneria Informatica

Enriching search results with semantic metadata

Giuseppe Alberto Mangano
665701

Relatore: Prof. Marco Colombetti
Correlatore: Ing. David Laniado

Information Retrieval

Information retrieval (IR) is finding material
(usually documents) of an unstructured nature (usually text)
that satisfies an information need from within large collections
(usually stored on computers)

[Manning et al., 2009]

Syntactic Search: overview

- Syntactic Search
  - the first, and currently the most used method
  - simple matching between query and document terms
  - good results with very large sets of documents
- Vector Space Model
  - the classic VSM: TF-IDF (Salton, Wong, Yang - 1975)
  - a user will mainly use free text queries
  - three main stages:
    - document indexing
    - weighting of indexed terms
    - computing similarities between query and documents

Syntactic Search: limitations

the indexed document:

bed and breakfast in Legnano

can be retrieved with queries such as:

“**bed and breakfast**”, “**Legnano**”

but cannot be matched with:

“**sleep**”, “**Milan**”

even though the document may be relevant
to the information needs of a user that inputs these terms

Semantic Search

- based on the computation of semantic relations between concepts
- it exploits the **meaning** of words using data from semantic networks to generate more relevant results

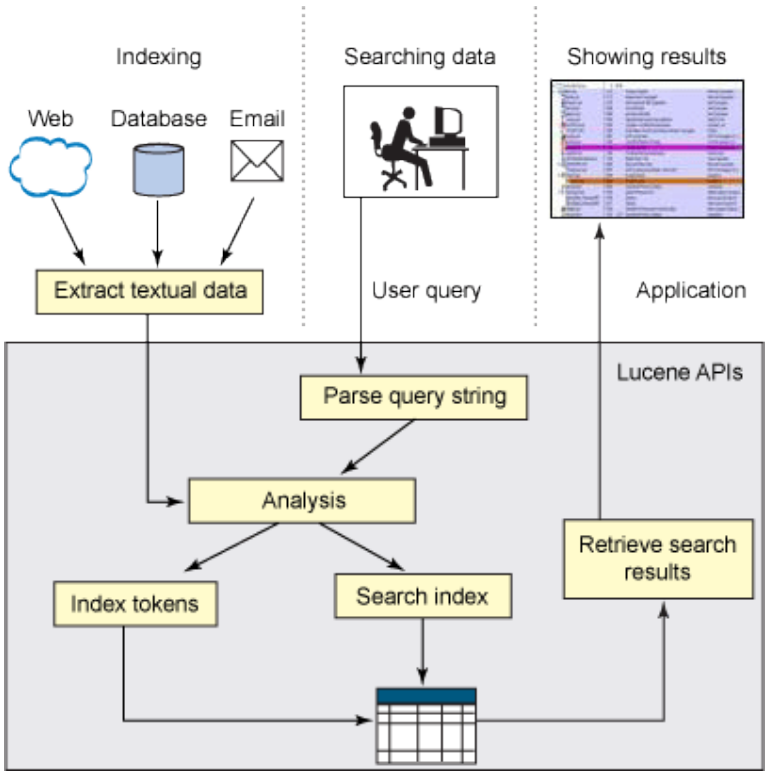
- index expansion
  - performed by associating to certain terms of a document other terms obtained from semantic networks
  - the document can be retrieved by matching the searched terms with the ones added semantically
- query expansion
  - performed by expanding the terms of the query to match additional documents already indexed

Goal

- Create a search engine prototype that enhances traditional Syntactic Search methods with the semantic expansion of terms present in documents and query strings.
  - Employing metadata in the form of payloads associated to terms added in the expansion, we want to ensure control over the ranking process to directly reflect the possible decrease in relevancy of documents retrieved using semantics.

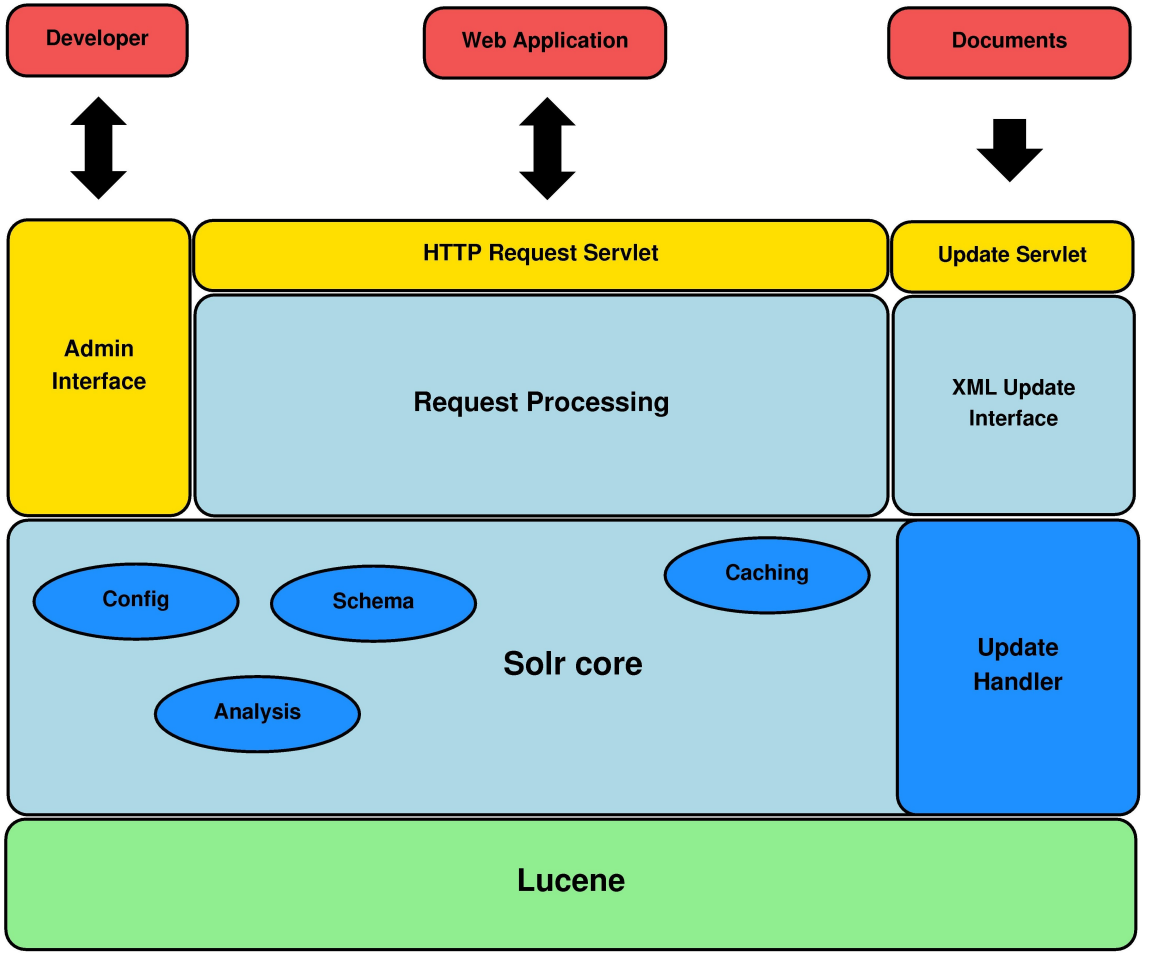
Apache Lucene

- a free/open source information retrieval library originally created in Java
- Lucene is an API (not an application) that handles the indexing, searching and retrieving of documents



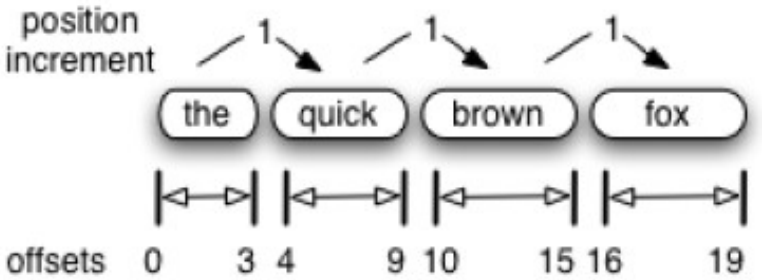
Apache Solr

- Solr is an open source standalone enterprise search server based on Lucene



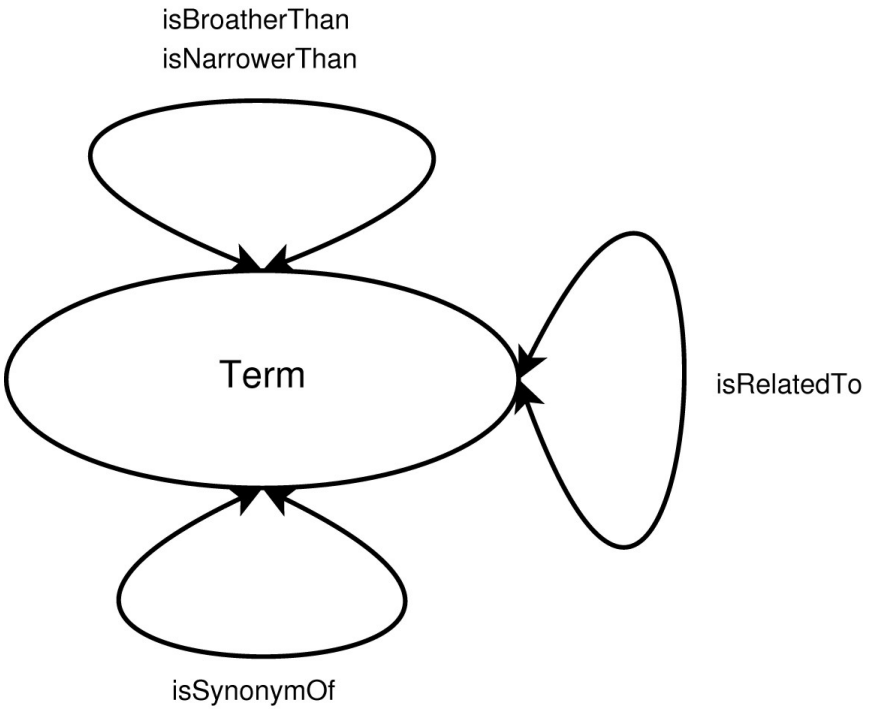
Lucene's Token Stream

- The fundamental output generated by the analysis process
- Each token usually represents an individual word of that text
- A token carries with it a text value (the word itself) as well as some metadata: the start and end offsets in the original text, a token type, a position increment and an optional payload.
- The token position increment value relates the current token to the previous one



Data sources

- GeoNames
  - a geographical database available through various Web Services, under a Creative Commons attribution license.
  - it covers all countries and contains over eight million placenames and other data such as latitude, longitude, elevation, population, administrative subdivision, and postal codes.
- Ontologies



dog ***isNarrowerThan*** pet
pet ***isBroaderThan*** cat
pet ***isNarrowerThan*** animal
bed and breakfast ***isRelatedTo*** sleep

Expansion Example

bed and breakfast in Legnano — ORIGINAL DOCUMENT

[bed] [and] [breakfast] [in] [Legnano] — WHITESPACE TOKENIZATION

[bed] [and] [breakfast] [in] [Legnano] — ADDED GEONAMES TERMS
[6537118]-{0.1}
[Europe]-{0.0256}
[Italy]-{0.064}
[Lombardy]-{0.16}
[Milan]-{0.4}

[bed and breakfast] [in] [Legnano] — ADDED ONTOLOGY TERMS
[sleep]-{0.2}
[accomodation]-{0.4}
[6537118]-{0.1}
[Europe]-{0.0256}
[Italy]-{0.064}
[Lombardy]-{0.16}
[Milan]-{0.4}

Implementation (1)

- SemanticFilter (our custom analyzer)
  - GeoNames parser
    - Java API for XML Processing
  - Ontology parser
    - JENA (a Semantic Web framework for Java)
  - Shingle matching algorithm (for multiword terms)
  - Payloads
    - a byte array of information associated to a term
    - encodeFloat of Lucene's PayloadHelper class
    - setPayload of Lucene's Token class

Implementation (2)

- PayloadBoostingSimilarity
  - extends Lucene's DefaultSimilarity (scoring)
  - uses PayloadHelper's decodeFloat
  - overrides scorePayload (which returns 1 by default)
- BoostingTermQuery
  - a payload-aware Query
  - it invokes the overridden scorePayload method
- PayloadQParserPlugin
  - we extend Solr's QParserPlugin to create custom query structures

Index Expansion

Document tokenization

term position	1	2	3	4	5
term text	bed	and	breakfast	in	Legnano
term type	word	word	word	word	word
source start,end	0,3	4,7	8,17	18,20	21,28
payload					

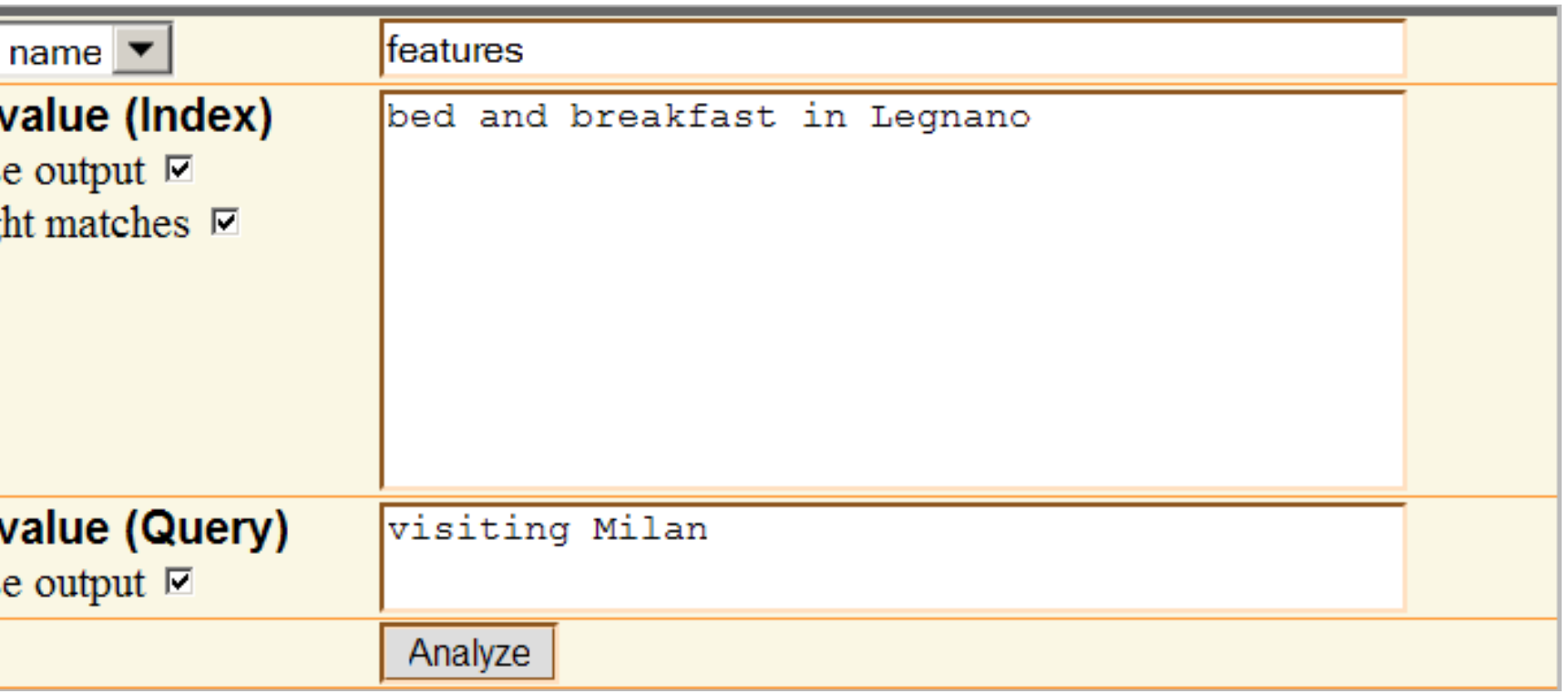
GeoNames parser

term position	1	2	3	4	5
term text	bed	and	breakfast	in	Legnano
					6537118
					Europe
					Italy
					Lombardy
					Milan
term type	word	word	word	word	processed
					geonames-id
					geonames-hierarchy-4
					geonames-hierarchy-3
					geonames-hierarchy-2
					geonames-hierarchy-1
source start,end	0,3	4,7	8,17	18,20	21,28
					21,28
					21,28
					21,28
					21,28
					21,28
payload					3dcccccd00000000
					3cd1b71700000000
					3d83126f00000000
					3e23d70a00000000
					3eccccccd00000000

Ontology parser

term position	1	2	3
term text	bed and breakfast	in	Legnano
	sleep		6537118
	accommodation		Europe
			Italy
			Lombardy
			Milan
term type	processed	word	processed
	isRelatedTo		geonames-id
	isNarrowerThan-1		geonames-hierarchy-4
			geonames-hierarchy-3
			geonames-hierarchy-2
			geonames-hierarchy-1
source start,end	0,17	18,20	21,28
	0,17		21,28
	0,17		21,28
			21,28
			21,28
			21,28
payload	3e4ccccd00000000		3dcccccd00000000
	3ecccccd00000000		3cd1b71700000000
			3d83126f00000000
			3e23d70a00000000
			3ecccccd00000000

Query input


Field
name
features
Field value (Index)
verbose output
highlight matches
bed and breakfast in Legnano
Field value (Query)
verbose output
visiting Milan
Analyze

Query processing

term position	1	2
term text	visiting	Milan
term type	word	word
source start,end	0,8	9,14
payload		

TOKENIZATION

term position	1	2
term text	visiting	Milan
term type	word	processed
source start,end	0,8	9,14
payload		

ANALYSIS

Match highlighting

term position	1	2	3
term text	bed and breakfast	in	Legnano
	sleep		6537118
	accommodation		Europe
			Italy
			Lombardy
			Milan
term type	processed	word	processed
	isRelatedTo		geonames-id
	isNarrowerThan-1		geonames-hierarchy-4
			geonames-hierarchy-3
			geonames-hierarchy-2
			geonames-hierarchy-1
source start,end	0,17	18,20	21,28
	0,17		21,28
	0,17		21,28
			21,28
			21,28
			21,28
payload	3e4ccccd00000000		3dcccccd00000000
	3ecccccd00000000		3cd1b71700000000
			3d83126f00000000
			3e23d70a00000000
			3ecccccd00000000

Scoring (1)


```
- <response>
  - <lst name="responseHeader">
      <int name="status">0</int>
      <int name="QTime">266</int>
    - <lst name="params">
        <str name="debugQuery">true</str>
        <str name="q">{!payloads f=features}visiting Milan</str>
      </lst>
    </lst>
  - <result name="response" numFound="2" start="0">
    - <doc>
      - <arr name="features">
          <str>nightlife in Milan</str>
        </arr>
        <str name="id">1</str>
        <int name="popularity">0</int>
        <str name="sku">1</str>
        <date name="timestamp">2009-09-20T19:19:25.031Z</date>
      </doc>
    - <doc>
      - <arr name="features">
          <str>bed and breakfast in Legnano</str>
        </arr>
        <str name="id">0</str>
        <int name="popularity">0</int>
        <str name="sku">0</str>
        <date name="timestamp">2009-09-20T19:19:24.531Z</date>
      </doc>
    </result>
```

Scoring (2)


```
− <lst name="debug">
    <str name="rawquerystring">{!payloads f=features}visiting Milan</str>
    <str name="querystring">{!payloads f=features}visiting Milan</str>
  − <str name="parsedquery">
      BoostingTermQuery(features:visiting) BoostingTermQuery(features:Milan)
    </str>
    <str name="parsedquery_toString">features:visiting features:Milan</str>
  − <lst name="explain">
   − <str name="1">
        0.052230984 = (MATCH) sum of: 0.052230984 = (MATCH)
        weight(features:Milan in 1), product of: 0.33131006 =
        queryWeight(features:Milan), product of: 0.5945349 = idf(features: Milan=2)
        0.55725926 = queryNorm 0.15764986 = (MATCH) fieldWeight(features:Milan
        in 1), product of: 0.70710677 = (MATCH) btq, product of: 0.70710677 =
        tf(phraseFreq=0.5) 1.0 = scorePayload(...) 0.5945349 = idf(features: Milan=2)
        0.375 = fieldNorm(field=features, doc=1)
      </str>
   − <str name="0">
        0.017410329 = (MATCH) sum of: 0.017410329 = (MATCH)
        weight(features:Milan in 0), product of: 0.33131006 =
        queryWeight(features:Milan), product of: 0.5945349 = idf(features: Milan=2)
        0.55725926 = queryNorm 0.052549955 = (MATCH) fieldWeight(features:Milan
        in 0), product of: 0.28284273 = (MATCH) btq, product of: 0.70710677 =
        tf(phraseFreq=0.5) 0.4 = scorePayload(...) 0.5945349 = idf(features: Milan=2)
        0.3125 = fieldNorm(field=features, doc=0)
      </str>
    </lst>
```

Conclusions

- Traditional syntactic-only search, albeit reliable and efficient, is greatly limited by the gap between the way machines work and the way we think
- Our search engine enriches search results with documents that traditional search engines fail to retrieve, while ensuring control over the ranking process

- **FUTURE DEVELOPMENTS**
  - handling Polysemy
  - storing data in an SQL database
  - tuning boost values
  - query expansion
    - eg. D: “bed and breakfast in Monza”; Q: “visiting Legnano”
    - Solr's query side support for payloads - SOLR-1337 (5th August '09)

Q & A

Questions?