Towards a search system for the Web exploiting spatial data of a web document

Stefan Dlugolinsky, Michal Laclavik, Ladislav Hluchy
Institute of Informatics, Slovak Academy of Sciences
Introduction

- Full text search services like
  - Google
  - Yahoo!
  - MSN
  - ...

determine the relevance of the web document by its content and importance (PageRank).

- Sometimes, we would like to prefer results, whose relevance is determined by a distance from us, where the distance is measured from the business location presented in the web document.

- Examples:
  - We are going to visit Bilbao and we are looking for a hotel.
  - We are looking for the nearest shopping centre, restaurant, pharmacy, etc.
Geographic distance of a web document

Under a document's geographic distance, we understand the distance between a search reference point and a geographic location related to the document.
Introduction

- Present search services like

  2. Yahoo Maps http://maps.yahoo.com/
  3. ZlateStranky.sk (Slovak Yellow pages) http://www.zlatestranky.sk/
  5. MAPY.SK http://mapy.atlas.sk/
  6. Umkreisfinder http://www.umkreisfinder.de/

  search documents by distance, **BUT**

  - the search domain is created manually by users (1, 2), or built from the content of service catalogs – registered companies (3, 4)
  - some search services (5) search only within a set of geographic object names (street names, city names, ...)
  - geographic location must be explicitly set by special HTML meta tag, or is known by an URL of the document (6)
Introduction

Our goal was to create a search service which could:
- automatically create its search domain (web documents)
- automatically determine geographic positions related to web documents
- search web documents by geographic distance according to search query

Above tasks rose up main problems, which should be treated:
- Assigning geographic locations to a web document
  - Extraction of geographic entities from the documents
  - Geocoding extracted entities
- Indexing documents by related geographic positions
- Search by distance
The core of the system

- We used NUTCH search engine as the basis of our search system, which:
  - is open source and platform independent (Java)
  - is built on LUCENE text search engine library – an implementation of high-performance indexing and searching
  - includes web crawler for crawling web documents
  - includes parsers for different types of documents (HTML, PDF, RTF, ...)
  - except LUCENE, it has also integrated HADOOP, an open-source implementation of frameworks for reliable, scalable, distributed computing and data storage
  - has a plug-in system for easy extension
DistanceSearch plug-in

- We have built the search system extending NUTCH by our plug-in DistanceSearch and also by customizing the user interface.

- The DistanceSearch plug-in deals with three main issues:
  - **Extraction** of geographic coordinates from web documents and geocoding
  - **Indexing** by geographic distance
  - **Search** by geographic distance
Determining the geographic location of the document

Strategy:

- Extract all the possible geographic entities from the textual content of the document:
  - GPS coordinates (more accurate, no geocoding)
  - Postal addresses (less accurate, requires geocoding)

- Geocode all the found street addresses to geographic coordinates

Extracted coordinates are the input for the indexing process

<table>
<thead>
<tr>
<th>GPS coords.</th>
<th>Latitude and Longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addresses</td>
<td>48.1668595, 17.1032454</td>
</tr>
<tr>
<td></td>
<td>48.1441842, 17.1095594</td>
</tr>
<tr>
<td></td>
<td>48.1099847, 17.1146532</td>
</tr>
</tbody>
</table>
Geographic entity extraction

- **Extraction model**
  - Based on **regular expressions (regexes)**
  - Extracted entities are in the form of **key-value** pairs
    - for example. "sk.address.street.name" => "Jasovska"
  - We define extraction patterns by custom meta-language in XML
    - simple modification of extraction patterns
    - easily expandable with new extraction patterns
    - extraction model supports macros and naming of regex back references (it is easier to create complex patterns and reuse them in other patterns)
An example of address extraction pattern

```xml
<pattern name="Postcode" class="Postcode">
  <regexp><![CDATA[[0-9]{3}\s*[0-9]{2}]></regexp>
</pattern>

<pattern name="Word">
  <regexp><![CDATA[([p{Lu}\p{Ll}]*|\p{Ll}+)</regexp>
</pattern>

<pattern name="Name">
  <regexp><![CDATA[\p{Lu}\p{Ll}+]></regexp>
</pattern>

<pattern name="StreetName" class="StreetName">
  <regexp><![CDATA[\p{#Name}\.?( +\p{#Word}\.?)?( +\p{#Word}\.?)?( +\p{#Word}\.?)?]]></regexp>
</pattern>

<pattern name="BuildingNumber" class="BuildingNumber">
  <regexp><![CDATA[[1-9][0-9]?(/-|)\?([1-9][0-9][0-9]?)?([\p{a-zA-Z}])?]]></regexp>
</pattern>

<pattern name="CityName" class="CityName">
  <regexp><![CDATA[\p{#Name}( +\p{#Name})?]]></regexp>
</pattern>

<pattern class="Address">
  <regexp><![CDATA[\p{#CityName} +\p{#BuildingNumber}(\|s*|s+)\p{#Postcode}\s+\p{#CityName}]]></regexp>
</pattern>
```
Geographic entity extraction

Extraction patterns:

- 2 extraction patterns for Slovak postal address extraction. Patterns deal with the diversity of city and street naming, also building numbering:
  - (StreetName> <BuildingNumber>?)?<Postcode> <CityName>
  - (StreetName> <BuildingNumber>?)?<CityName> <Postcode>

- 4 extraction patterns for GPS coordinate extraction
Geocoding of postal addresses

- **Geocoding**
  a process of converting postal addresses or known geographic object names (landmarks, institutions, etc.) into geographic coordinates.

- The original idea was to create our own geocoding service using **gazetteer**.
  - We designed, implemented and verified data model of the gazetteer, filled it with addresses and got hierarchical structure of related street names, postcodes, city names, county names and region names, according to Slovak address format
  - However, we did not have very precise geographic data. Therefore we were able to geocode addresses only up to the level of street names (without exploiting extracted building numbers, what would be more precise)

- For the reason to be more precise in geocoding, we have decided to use **external geocoding services**
  - Google
  - Yahoo

- We cache all calls to geocoding services into the database (it is due to each service request limits and recrawling optimalization)
We wanted to index web documents by **more than just a one geographic location** - the problem: LUCENE’s indexing capabilities inside NUTCH (supports only string values) and tightness between indexing and searching (need to be fast).

- Efforts to represent latitude and longitude in one string, a hash.
- The idea:
Indexing

  - Developed with a grant support of NASA (for use in digital astronomical maps).
  - Built on the same principle, but divides the spherical surface into the triangles:

  ![](image)

  - The precision of HTM index grows with the level of triangle subdivision (the resolution at the level 25 is about 0.6 m)
  - We use HTM index to index document by its related geographic coordinates.
    - An example: 48.1668595 17.1032454 => N33133121001121003031

**Feature:** If there are two points with common prefixes, they are close to each other. The more longer prefix they share, the closer they lie to each other.
The principle of search:

Extension of the generated NUTCH query by an area search query based on HTM region prefixes.

1. A set of common HTM prefixes is computed for triangles within the search area.
2. There is a **prefix subquery** created for each computed common HTM prefix:

\[ PK_i = \text{PrefixQuery}("htm", \text{prefix}_i) \]

3. Prefix subqueries are joined into an **area search query**:

\[(PK_1 \text{ OR } PK_2 \text{ OR } ... \text{ PK}_n)\]

4. The area search query is then added to the generated NUTCH query:

\[(\text{NUTCH query}) \text{ AND } (PK_1 \text{ OR } PK_2 \text{ OR } ... \text{ PK}_n)\]

Extended Nutch query is fulfilled for each document, which is relevant to original query and “lies” in the search region.
Evaluation of address extraction

- Set of test web documents
  - Google catalogue, section “Bratislava” and Slovakia

<table>
<thead>
<tr>
<th>Documents</th>
<th>3183</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevant documents</td>
<td>380</td>
</tr>
<tr>
<td>Addresses in documents</td>
<td>847</td>
</tr>
<tr>
<td>Correct address extractions</td>
<td>741</td>
</tr>
<tr>
<td>False address extractions</td>
<td>730</td>
</tr>
<tr>
<td>Missed addresses</td>
<td>106</td>
</tr>
<tr>
<td>Precision</td>
<td>50.37%</td>
</tr>
<tr>
<td>Recall</td>
<td>87.49%</td>
</tr>
</tbody>
</table>

$\text{Precision} = \frac{|R \cap I|}{|I|} = \frac{741}{1471} = 0.5037 = 50.37\%$

$\text{Recall} = \frac{|R \cap I|}{|R|} = \frac{741}{847} = 0.8749 = 87.49\%$
Distributed crawling and its results

- We installed crawler on a cluster with 8 nodes, each with the following configuration:

<table>
<thead>
<tr>
<th>Processor</th>
<th>Intel® Core™ 2 Quad CPU Q9550 @ 2.83GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>4 GB</td>
</tr>
<tr>
<td>HDD</td>
<td>WDC WD7500AACS-0 (750 GB)</td>
</tr>
<tr>
<td>OS</td>
<td>Linux 2.6.24-19-generic X86_64 GNU/Linux</td>
</tr>
</tbody>
</table>

- We have configured Hadoop to simultaneously run 8 jobs on every node, i.e. 64 parallel jobs on whole cluster.

<table>
<thead>
<tr>
<th>Crawling start:</th>
<th>Crawling time</th>
<th>20h 46m 7s</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.zoznam.sk/">http://www.zoznam.sk/</a></td>
<td>Crawled data size (index + cache)</td>
<td>8.3 GB</td>
</tr>
<tr>
<td><a href="http://www.centrum.sk/">http://www.centrum.sk/</a></td>
<td>Index size</td>
<td>535.7 MB</td>
</tr>
<tr>
<td><a href="http://www.best.sk/">http://www.best.sk/</a></td>
<td>Documents crawled</td>
<td>408 096</td>
</tr>
<tr>
<td><a href="http://szm.sk/">http://szm.sk/</a></td>
<td>Documents without entity extraction</td>
<td>349 561 (85.66 %)</td>
</tr>
<tr>
<td><a href="http://www.atlas.sk/">http://www.atlas.sk/</a></td>
<td>Documents with entity extraction</td>
<td>58 535 (14.34 %)</td>
</tr>
<tr>
<td><a href="http://www.katalog.sk/">http://www.katalog.sk/</a></td>
<td>Extracted entities</td>
<td>245 673</td>
</tr>
<tr>
<td><a href="http://www.azet.sk/">http://www.azet.sk/</a></td>
<td>Geo-coding requests</td>
<td>21 463</td>
</tr>
<tr>
<td>max. level 10</td>
<td>Average number of entities per document</td>
<td>0.60</td>
</tr>
<tr>
<td>limit 100 000 / level</td>
<td>Recognized GPS coordinates</td>
<td>256</td>
</tr>
<tr>
<td></td>
<td>Not geo-coded address extractions</td>
<td>123 731</td>
</tr>
<tr>
<td></td>
<td>Geo-coded address extractions</td>
<td>121 686</td>
</tr>
</tbody>
</table>
## Comparison with similar services

<table>
<thead>
<tr>
<th></th>
<th>DistanceSearch (NUTCH)</th>
<th>GeoPosition (NUTCH)</th>
<th>Google Maps</th>
<th>Yahoo Local Maps</th>
<th>Kompas.sk</th>
<th>ZlateStranky.sk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Search domain</strong></td>
<td>web</td>
<td>web</td>
<td>db</td>
<td>db</td>
<td>db</td>
<td>db</td>
</tr>
<tr>
<td><strong>Geo-entity extraction from document textual content</strong></td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Search in map viewport</strong></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Display results on a map</strong></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Order results by distance</strong></td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Searcher user interface

Map viewport
• defines search area (reference point is in the middle of the map)
• display of results

Search results details

Search query input field

List of results
• results are ordered by distance from the reference point
Conclusion

- **Advantages:**
  - Automatic address extraction and geocoding with high recall
  - Address and GPS extraction patterns are easily extensible
  - Documents are indexed by more than just a one related geographic location
  - Distributed crawling+extraction+indexing
  - External geocoding services can be used
  - Caching geocoding calls (decreases direct calls to geocoding services when recrawling)
  - Results are shown on a map

- **Disadvantages and future work:**
  - Lower extraction precision, but new patterns can improve it. Currently we have integrated also GATE (General Architecture for Text Extraction) in Nutch and trying it.
  - Geographic entity extraction does not handle with microformats, RDF in XHTML, embedded maps (Google Maps, Yahoo Local Maps, ...)
  - Geographic positions are associated only with their parent pages, but sometimes they could be associated with pages linked to their parent page (a future work challenge)
Thank you
DistanceSearch plug-in

Diagram of DistanceSearch plug-in integration in NUTCH
Geocoding of postal addresses