A personalized search engine for microblog contents

Cimmino Fabio 807070
Lotterio Roberto 807500

Data Collection

To collect tweet for our project we used Twitter4J, an unofficial java library for Twitter API. The collected tweets cover a period of one month and are divided into five categories :

- Cinema
- Music
- News
- Sport
- Science

We took mixed tweets, which therefore include popular tweets (tweet with high number of like/retweet or tweet of popular people/channel) and common tweets.

Dataset Creation

Once the tweets were downloaded, we created the dataset used for the project. so for each tweet we have memorized the following fields:

- Date and time
- Username
- Content of tweet
- Number of likes
- Number of retweets
- URL’s tweet

Stages of Text Processing

- **Date and Time:** we trasformed the date and time that provided us Twitter in the standard format «dd-MM-yyyy HH:mm».
- **Content of the tweet:** we have created a Custom Analyzer which performs the following steps:
  1. we used the Lucene's Standard Tokenizer for tokenization
  2. each token has been transformed into lowercase characters
  3. we removed the stop words using a pre-existing set of English words
  4. removed words of less than two characters, numbers and special characters
  5. stemming with Porter's algorithm

Index creation

The fields in the index can be indexed and/or stored:

- Indexed: the field is analyzed and indexed, and can be searched
- Stored: the field's full text is stored and will be returned with search results

Fields	Tokenized	Indexed	Stored
Date and time	No	No	Yes
Username	No	Yes	Yes
Tweet's content	Yes	Yes	Yes
# likes	No	Yes	Yes
# retweets	No	Yes	Yes

Scoring Function

$$score(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{t \in q} \left(tf(t \in d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d) \right)$$

- tf(t,d) = defined as the number of times term t appears in the currently scored document d.
- idf(t) = stands for Inverse Document Frequency.
- coord(q, d) = is a score factor based on how many of the query terms are found in the specified document.
- queryNorm(q) = is a normalizing factor used to make scores between queries comparable.
- t.getBoost() = is a search time boost of term t in the query q.
- norm(t,d) = encapsulates a few boost and length factors.

Query Formulation

- The first step in the query process standardizes the user request using the analysis process previously performed for the creation of the index.
- The query entered by the user will be searched in the field relating to the content of the tweet and it is analyzed with the Custom Analyzer.
- The tokens and the search terms are identified in the string entered by the user and sent to the search engine for the matching phase.
- The query supports using parentheses to group clauses to form sub queries. This can be very useful if you want to control the boolean logic for a query.

Custom Flexible Scoring and Dimensions of Relevance

- In addition to the topical relevance, the search can be customized by displaying the results for a dimension of relevance: like or retweet. In this way, a boost is made on the final score, multiplying it by the selected field.

- You have to be able to directly access the like and retweet values of a specific document, therefore these fields are not managed by the inverted file but are stored in a data structure with direct access.

User Profile

- The search for tweets can be customized with the interests represented by a user profile.
- A user profile is composed of multiple layers where each layer indicates the user preferences associated with a specific category (music, science, news, sports and cinema).
- The interests of each user were selected from the dataset containing the tweets going to inspect both their natural language content and hashtags.

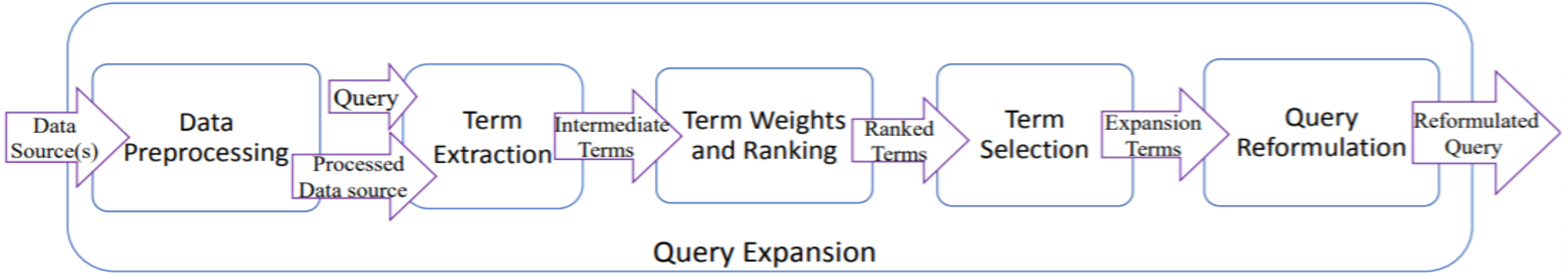
Simple Search and Personalized Search

- The search can be simple or personalized:
  - Simple: a based search
  - Personalized: the search can be customized for user profiler, category and relevance

- For the creation of the final custom query we made the following assumptions:
  - the terms entered in the search field must appear necessarily among the retrieved documents
  - the terms of the user profile may not appear but, if they appear, the score of the retrieved document will be more highly.

Query Expansion

- Expanding the query can be very useful because the terms the user searched for may not appear in our documents.



- To expand the query we used the following two methods in the literature [1]:
  - Pseudo relevance feedback
  - Automatic Local Analysis with Local Clustering (Association, Metric and Scalar clusters)

---

[1] Relevance Feedback and Query Expansion, Baeza-Yates & Ribeiro-Neto, Modern Information Retrieval, 2nd Edition

Pseudo Relevance Feedback

- We used Rocchio's algorithm to implement Pseudo Relevance Feedback in the Vector Space Model.
- So we want to find a query vector that maximizes similarity with relevant documents (C_r) while minimizing similarity with nonrelevant documents (C_{nr}).

$$\vec{p}_{opt} = \text{argmax}_{\vec{q}}[\text{sim}(\vec{q}, C_r) - \text{sim}(\vec{q}, C_{nr})]$$

- The set of relevant documents is unknown. Therefore we produce the following modified query m by adding the original query:

$$\vec{q}_m = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j$$

Association Clusters

- We define the correlation factors $C_{u,v}$ between any pair of terms k_u and k_v, where $C_{u,v} \in C_l$ (local association matrix), as follow:

$$c_{u,v} = \sum_{d_j \in D_l} f_{u,j} \times f_{v,j}$$

- Let $C_u(n)$ be a function that return the *n* largest factor $C_{u,v} \in C_l$, with $u \neq v$. Then, $C_u(n)$ denotes a local association cluster, a neighborhood, around the term k_u.
- For each term $k_u \in q$, select *m* neighbor terms from the cluster $C_u(n)$ and add them to the query:

$$q_m = q \cup \{k_v | k_v \in C_u(n), k_v \in q\}$$

Metric Clusters

- A metric cluster re-defines the correlation factors $C_{u,v}$ as a function of their distances in documents:

$$c_{u,v} = \sum_{d_j \in D_l} \sum_{n} \sum_{m} \frac{1}{r(k_u(n,j), k_v(m,j))}$$

- Where $r(k_u(n,j), k_v(m,j))$ is a function that computes the distance between:
  - the *nth* occurrence of term k_u in document d_j
  - the *mth* occurrence of term k_v in document d_j

Scalar clusters

- The correlation between two local terms can also be defined by comparing the neighborhoods of the two terms.

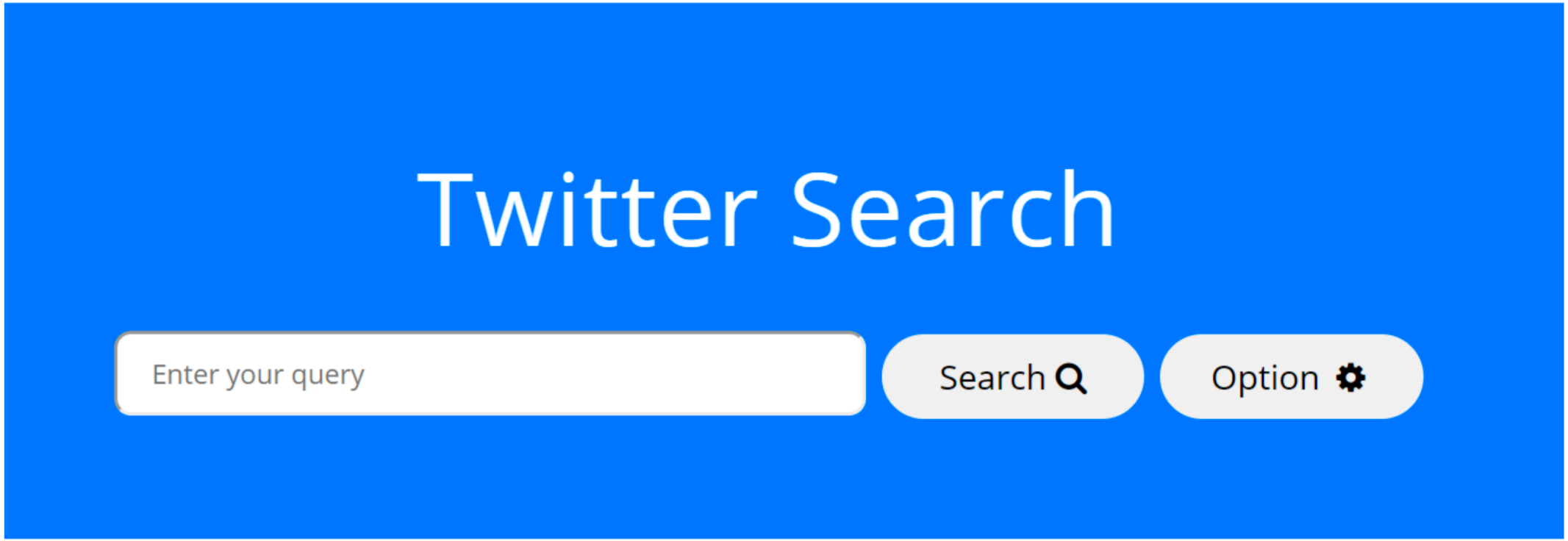
- Similar *neighborhoods* have some synonymity relationship. We can quantify this relationship comparing the neighborhoods of the terms through a scalar measure (for instance, the cosine of the angle between the two vectors):

$$c_{u,v} = \frac{\vec{s}_u \cdot \vec{s}_v}{|\vec{s}_u| \times |\vec{s}_v|}$$

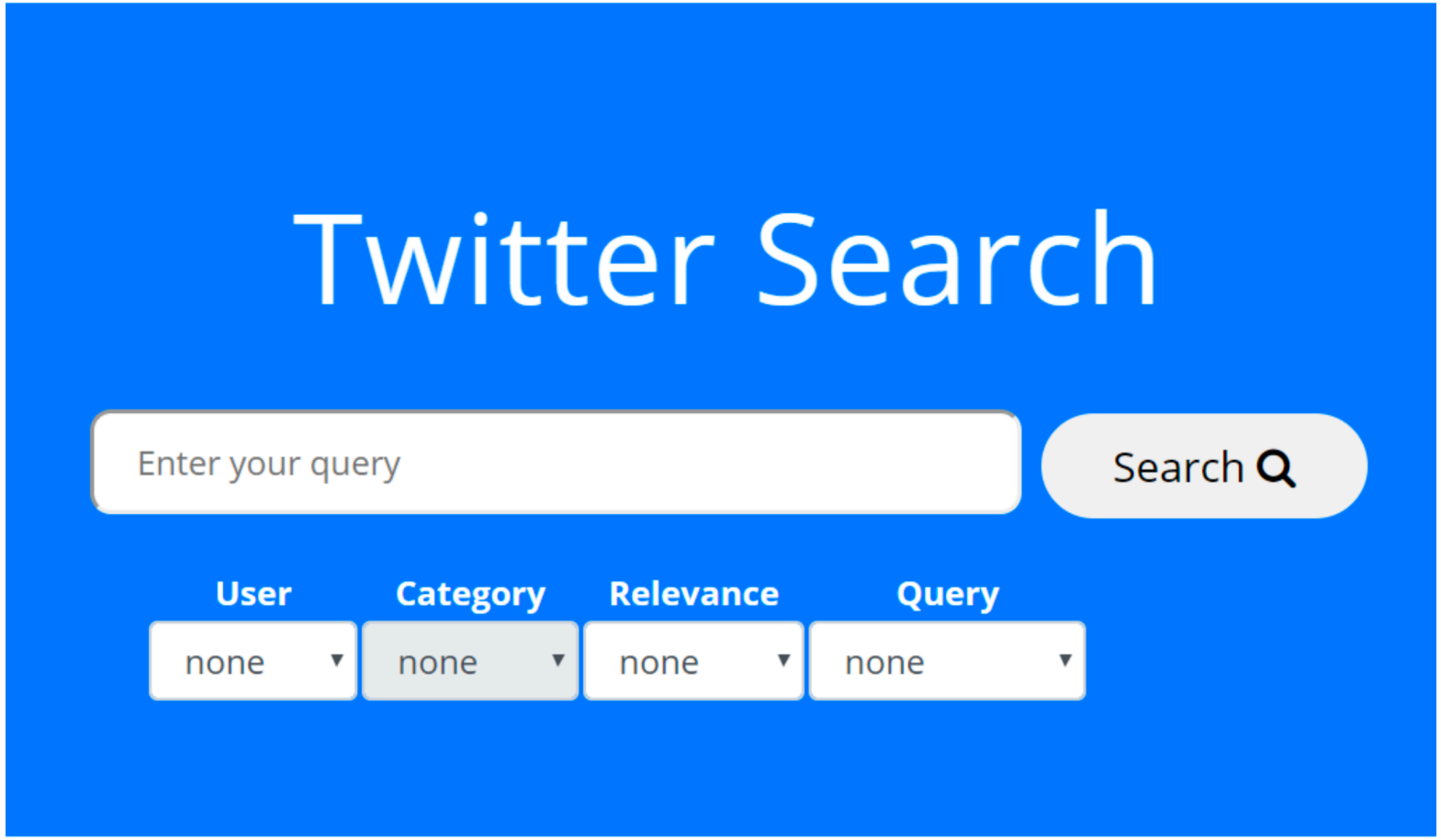
Web Application

- To create the web application we used SpingBoot with Thymeleaf.
- The web app consists mainly of two pages, one for simple search and one for personalized search, in order to access advanced search there is an options button.
- In the personalized search section you can customize the search combine:
  - User profile
  - Category
  - Relevandce
  - Query expansion.
- The search results are displayed with the url, the number of likes and retweets, the date and time and obviously the content of the tweets.

Search Engine



Personalized Search



Page of Results

Results

https://twitter.com/AlluSirish/status/1219563843136786433

:6091 : 514 *21-01-2020 11:14*

Congrats Dad on getting the "Champions of Change" award for exemplary contribution to cinema from the Govt of India. It's a proud moment to see you and former president Shri Pranab Mukherjee in the same frame. https://t.co/In6rmZZLuJ

https://twitter.com/utdxtra/status/1218336166178631680

:4641 : 354 *18-01-2020 01:55*

Silas (Sporting manager): "We have to understand the ambition of the player, who prefers to play in a league superior to ours. There has been talk about the English league, but who wouldnâ€™t like to play there? Bruno deserves everything." #mufc [Record, @Sport_Witness]

https://twitter.com/ManCity/status/1218833950996606978

:1557 : 98 *19-01-2020 10:53*

🌟 @aguerosergiokun hailed as 'the Premier League's best-ever striker'\n🏆 Legend backs City for Champions League glory\n💙 Cushing's debt to trio of stalwarts...\n\n🔵 #mancity \n\nhttps://t.co/MheGa38kT4