Information Retrieval

2021-2-F1801Q110

Gabriele Ferrario - 817518 - g.ferrario@campus.unimib.it

Riccardo Pozzi - 807857 - r.pozzi@campus.unimib.it

Personalized Search Engine for microblog

Introduction

The goal of this project is to define a search engine that enables user profiling and allows users to perform both standard searches as well as advanced and personalized ones.

Starting from data obtained from the social network Twitter.com[1] it has been possible to create users' profiles starting from their personal tweets by extracting their main interests thanks to the Tf-idf metric from both the words and the hashtags they frequently use.

The implemented search engine allows the user to search for news tweets containing specific keywords or hashtags or a combination of both, providing additional advanced query modes.

[1] "Twitter." https://twitter.com/.

Indeed eventual errors in keywords, such as typographical ones, have been considered by the implementation of an approximate or fuzzy search mode, which is available to the user along with the synonyms expansion mode which, thanks to WordNet [2], is able to enrich the original query so that it matches synonyms too.

Furthermore, once an user is logged in, search results are refined according to his/her personal interests extracted as mentioned above.

Dataset

Dataset is composed by tweets directly collected from Twitter.com using the official Twitter API through tweepy[3] python library.

All of them have been saved directly into Elasticsearch[4] and they've been organized in two different indices: "usertweets" and "retrievalbase" which contain respectively tweets from users for profiling purposes and the ones that will be the targets of the search engine.

About users' profiles: six twitter accounts of english speaking people were considered; three among politicians:

- Barack Obama @BarackObama
- Boris Johnon @BorisJohnson
- Joe Biden @JoeBiden

and three among sportsmen:

- LeBron James @KingJames
- Raheem Sterling @sterling7
- Kevin Durant @KDTrey5.

All of these tweets have been then analyzed to obtain a profile for each user which is

[2] George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[3] "Tweepy." https://www.tweepy.org/.

[4] "Elastic." https://www.elastic.co/.

represented as a document inside the Elasticsearch index “users”.

About the retrieval base instead, several tweets were gathered from the following newspaper profiles:

- Wall Street Journal @WSJ
- BBC News (World) @BBCWorld
- BBC Sport @BBCSport
- New York Times @nytimes
- USA TODAY @USATODAY
- FOX Sports @FOXSports

and then saved into the “retrievalbase” Elasticsearch index.

Totally the number of collected tweets is 36.893; 17.551 of which are users’ tweets and the remaining 19342 represent the retrieval base instead.

Each of them is provided in JSON format and includes several attributes such as “id”, “full_text” and “user” as shown in *Figure 1*.


```
{
    "created_at": "Fri Jan 22 19:00:48 +0000 2021",
    "id": 1352692708448595968,
    "full_text": "Hank Aaron was one of the best baseball players ...",
    "truncated": false,
    "entities": {
        "hashtags": [
            {
                "text": "doyourjob"
            }
        ]
    },
    "user": {
        "id": 813286,
        "name": "Barack Obama",
        "screen_name": "BarackObama",
        "location": "Washington, DC",
        "description": "Dad, husband, President, citizen.",
        "profile_background_image_url_https": "https://a.twimg.com/b.png",
    },
    "lang": "en",
    ...
```

```
}
```

Figure 1: Example of Tweet where the most significant fields are indicated

The project's focus has been directed especially to "full_text" which represents the body of the tweet and to the hashtags present in the tweet. On average "full_text" contains 26 words, while regarding hashtags only one tweet out of four contains a hashtag.

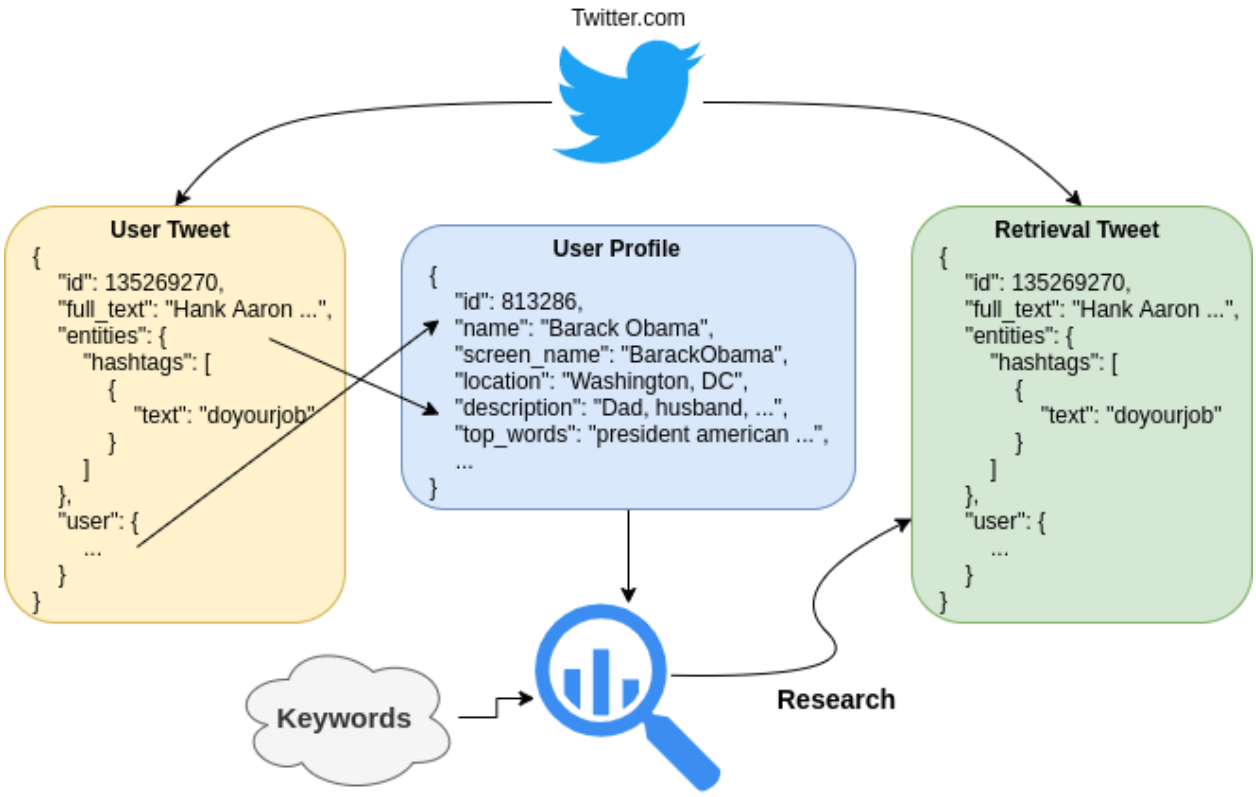

Figure 2: Data flow and data structure

Search Engine

The proposed search engine is based on Elasticsearch where the entire content of each tweet has been saved but only the necessary fields have been indexed; index mapping has been set to “static” and all the unneeded fields have been disabled while a correct mapping has been specified for the crucial ones.

“Full_text” is treated as text while hashtags as both keyword and text to allow exact search (keyword) as well as to take advantage of the text analyzer, which for instance enables case insensitive searches.

The chosen default analyzer for text fields is the built-in “english” featuring the following operations.

- standard tokenizer: tokenises using white spaces, hyphens and other characters
- english_possessive_stemmer: remove possessives (‘s)
- lowercase folding
- english_stop: removes stop words
- English_stemmer (Porter stemmer)

Another analyzer has been defined to enrich query with synonyms, called “english_synonym”, starting from the built-in “english” and adding a “synonym_graph” filter configured to use WordNet synonyms and positioned right after the lowercase folding. It has been used at search time only in order to allow the choice of whether to consider synonyms or not and at the same time avoiding to create bigger indices because of synonyms. The only acceptable downside is that by the fact the synonym expansion is issued at query time a little more computational effort is required.

```
original: "Hank Aaron was one of the best baseball players"
english: "hank aaron on best basebal player"
english_synonym: "aaron henri hank loui aaron aaron 1 i ac singl uniti
    matchless nonpareil on peerless unmatch unmatch unriv unrival an
    unitari on onli topper c charl outdo outflank trump scoop better best h
    best herbert best basebal game player”
```

Figure 3: an example showing output from english and english_synonym analyzers

User profiles are represented as documents inside the “users” index which contain basic information such as name, description and frequently used emojis along with more valuable ones like “top words” and “top hashtags”, respectively the top 30 most important words and the top 10 hashtags. This importance comes from Tf-idf score which outperformed the simpler approach based on the number of occurrences of each word: each user has been considered as a document containing all his tweets’ full_text and hashtags; full_texts have been then preprocessed through these steps:

- Lowercase folding
- Punctuation removal
- Hashtags removal (they’re treated separately)
- Removal of words composed by two or less characters, numbers, stop words, http links
- Emoji removal (they’re removed from full_text and then treated separately)
- Stemming (Porter stemmer: the same as the stemmer used when indexing)

After that Tf-idf value has been calculated for each word excluding the 5% most frequent ones considered as corpus-specific stop words.

The following example shows how Tf-idf improved the most relevant Barack Obama’s words by removing common used ones like “change”, “make” and “get”:

```
Top occurrences: "presid obama make chang american vote today year work get
time peopl climat help live senat whitehous watch one day job leader need
fight take america countri health read new'
"

Tf-idf: "obama presid american climat senat ofa whitehous leader economi
garland america potu plan court state add obamafound congress nation suprem
address elect organ gun econom worker equal joebiden weekli afford"
```

Figure 4: comparison of the extr most important words by number of occurrences and by Tf-idf

The search operation is performed against the “retrievalbase” index and it allows personalized search based on the user profile as well as different kinds of search modes which will be described in details in the next section.

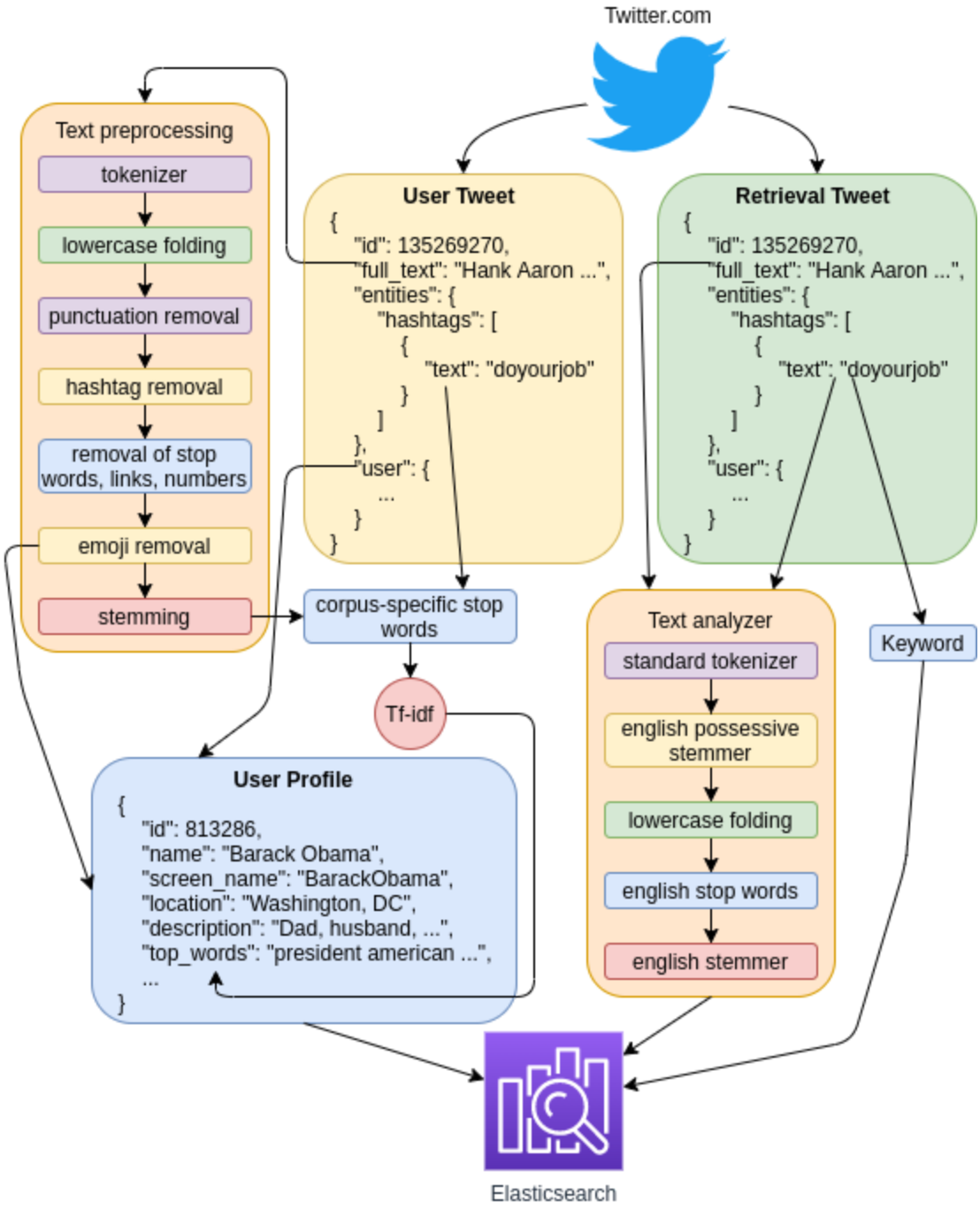

Figure 5: Diagram showing indexing flow on the right and profiling flow on the left.

Demonstration plan

The implemented search engine is available to the user through a simple React[5] based frontend user interface which directly access Elasticsearch APIs.

Two main search modes have been defined: a basic and an advanced one.

The basic one explicitly provides two input fields: one for querying the corpus or full_text of the tweet and another one for querying tweet's hashtags; combined use of both inputs is allowed, producing a conjunctive query (logical and) of the inputs in order to retrieve all and only the tweets matching both the corpus query and the hashtags one.

Under the hood multiple queries are issued inside an elasticsearch "must" (logical and) containing two "should" (logical or):

- A full text (match) query against corpus/full_text field in the first should
- Another full text (match) query against hashtags in the second should, which allows case insensitive matches and takes advantage of other steps done by the english analyzer such as stemming
- An exact (term) query against hashtags in the second should to privilege exact matches

Then more complex search features against the "full_text" can be added to the first should by toggling the corresponding switches:

- A fuzzy query with "fuzziness" set to "AUTO" allowing approximate results useful for example in case of typographical errors
- A synonym enhanced query which using the "english_synonym" analyzer is able to match keywords synonyms

[5] "React." https://reactjs.org/.

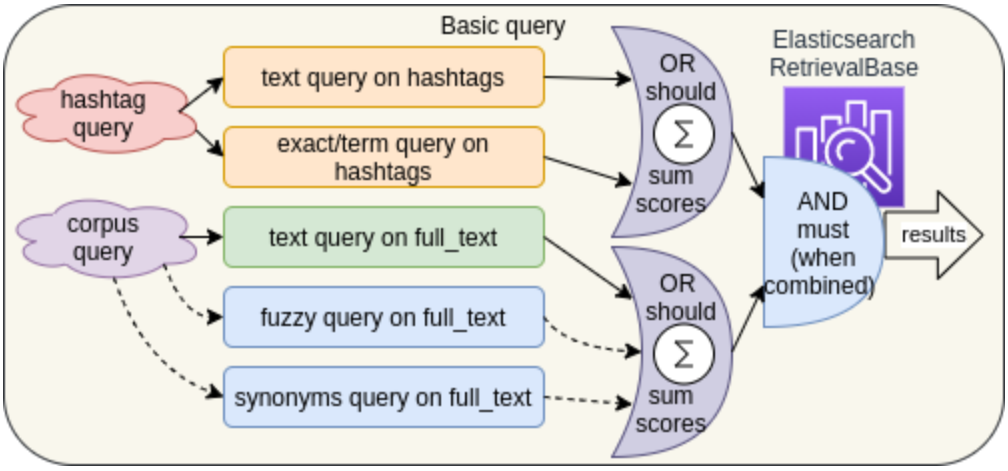

Figure 6: Basic query implementation schema.

This way better matches are privileged since what matches the first corpus query matches the additional queries too resulting in a score boost. Synonyms query has been further de-boosted because by increasing the number of words in the query it could influence too much the final scores with the risk of privileging a result containing multiple synonyms over an exact match.

The following figures show the results received from different examples of basic queries.

Figure 7: Basic query with corpus only.

Figure 7 shows a corpus only query where it's clearly observable the advantages given by the english text analyzer: the research is case insensitive thanks to lowercase step and plurals are found too thanks to stemming.

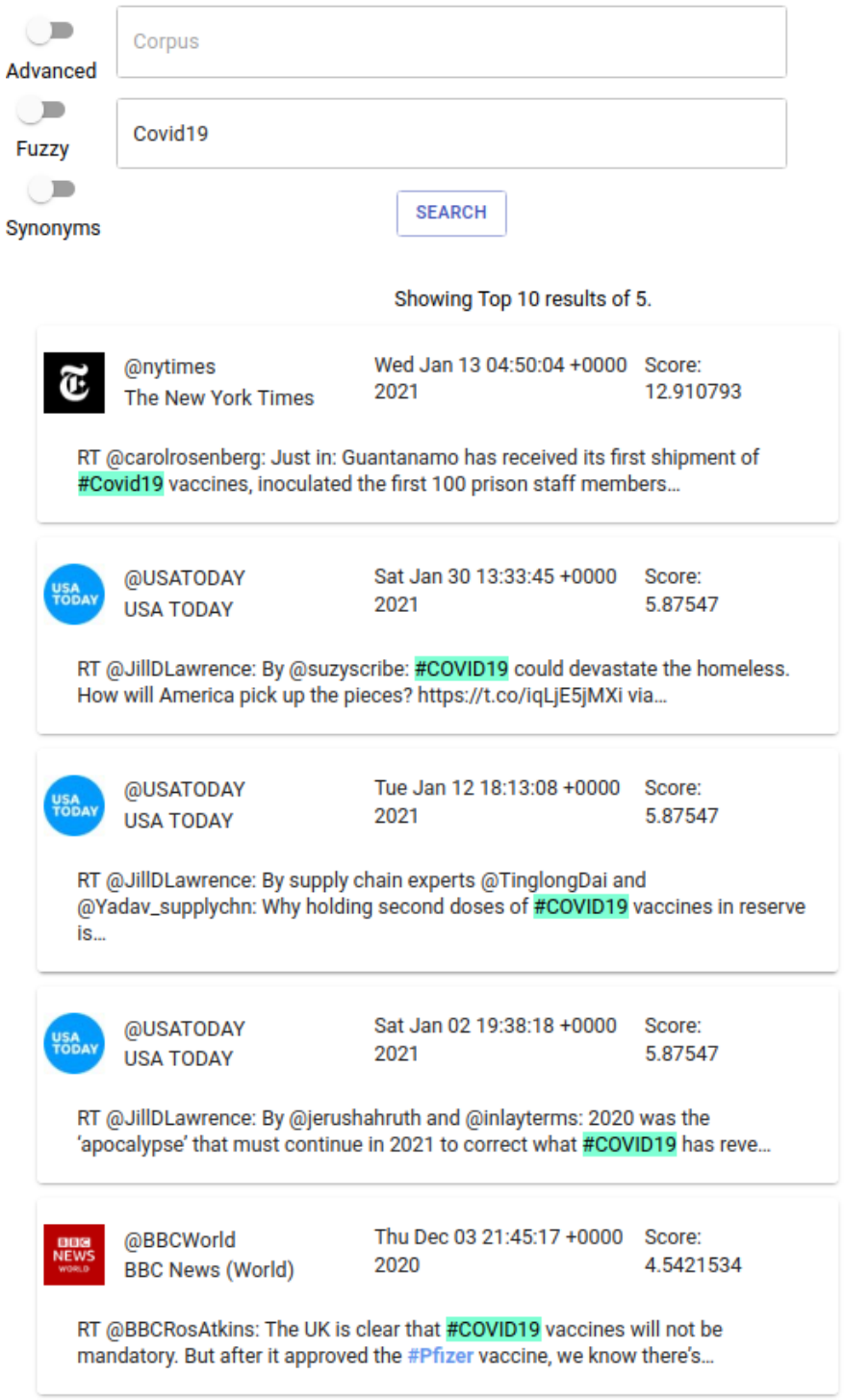

Figure 8: Basic query with hashtags only.

Figure 8 shows a hashtags only query where it's possible to note the effect of the two queries against hashtags. Indeed the exact match "Covid19" has about double the score of case insensitive matches as "Covid19" is matched by both hashtag subqueries.

Figure 9: Basic query combining corpus and hashtags.

Figure 9 shows a query which combines corpus and hashtags obtaining only tweets matching both.

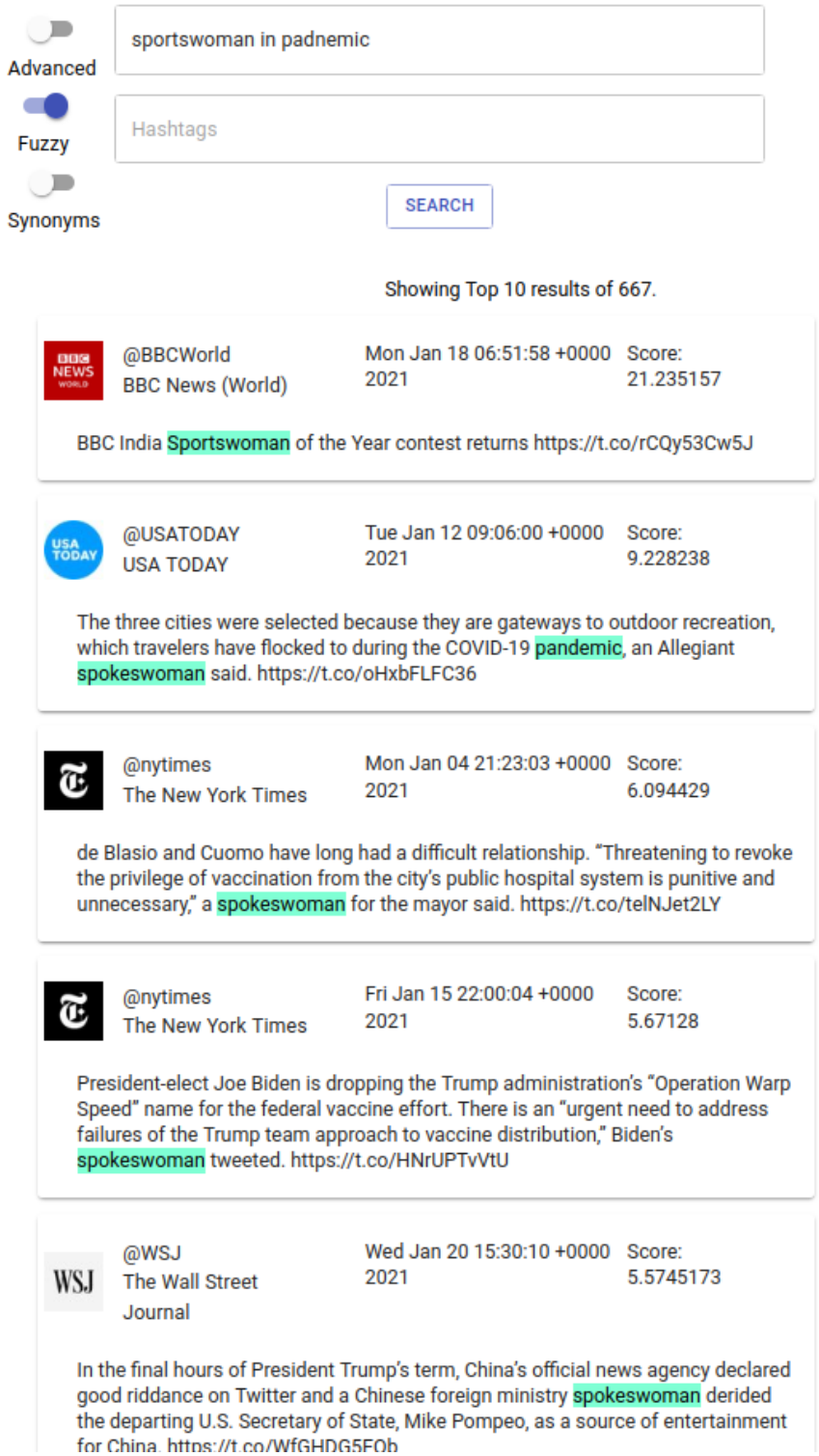

Figure 10: Fuzzy query.

Figure 10 is an example of a fuzzy query in which the keyword "pandemic" contains a typographical error. Nevertheless fuzziness allows the user to retrieve tweets matching "pandemic", and although no tweet matches completely the entire given query it's important to note the score of the first result which is over the double of the other results. This is due to the fact that it matches not only the fuzzy query but also the simpler full text one therefore the two queries scores are summed up.

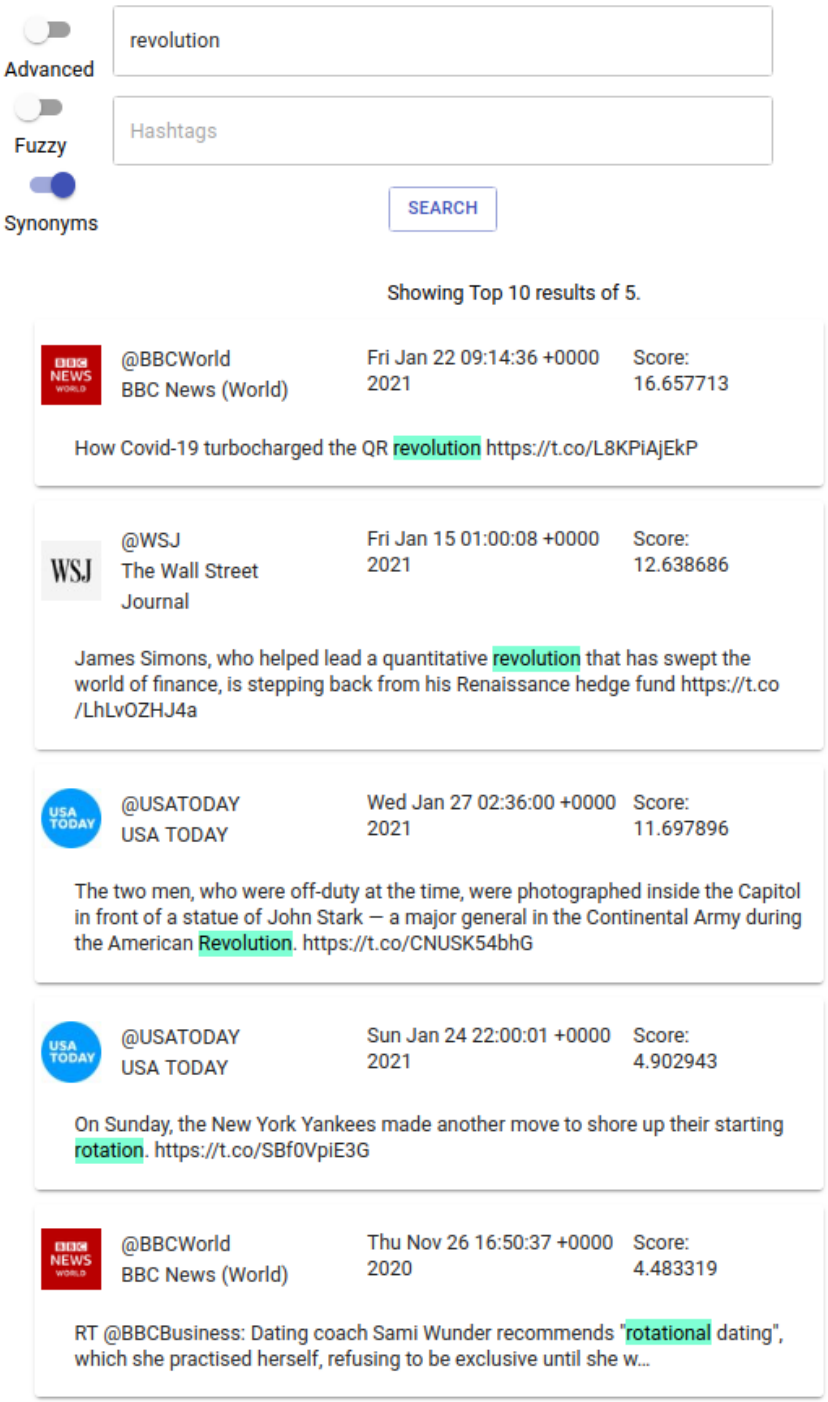

Figure 11: Synonym query

Figure 11 shows an example of a synonym enhanced query. The search for “revolution” related tweets brought back some tweets talking about rotations. Important is to note the difference in score between tweets about “revolution” (exact match) and the ones about “rotation” (synonym match).

The advanced query offers the user a single search bar which queries tweet's corpus or full_text field. Some tests were done to both search against full_text and hashtags using the same input query but, since hashtags are part of the full_text field, results matching hashtags were double boosted. The simpler query against full_text only proved to be more effective.

Furthermore the big difference with the basic query is that the user's profile, if logged in, is considered by the advanced query.

About the implementation, the input query has been privileged over the user profile so that results must mandatorily match the input query and user profile query is then used to boost matches closer to the user's personal interests.

Technically the input query is an elasticsearch "must" (results have to match the must) containing a "should" with a simple full text query against the corpus and eventually additional queries for fuzziness and synonyms expansion.

The profile query is instead a "should" with two queries:

- one that searches for user's profile top words in the full_text using the simplest "whitespace" analyzer which only divides the query in tokens as top words are already preprocessed
- and one which searches for the user's top hashtags in the hashtags field using text analyzer thus allowing case insensitive matches for instance.

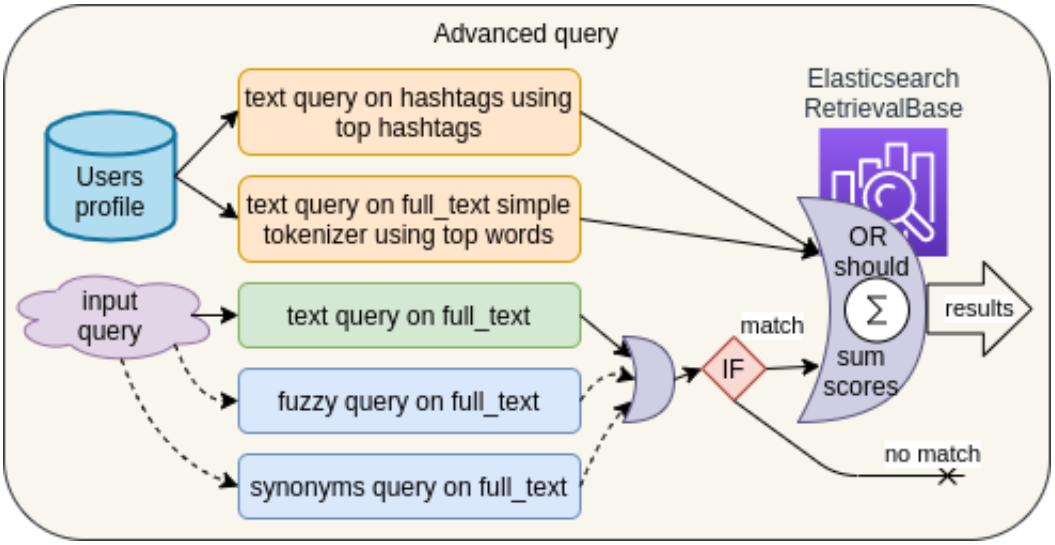

Figure 12: Advanced query implementation schema.

Additionally the possibility to “take a look” at what tweets are the closest to the user profile has been provided: in advanced query mode when a user is logged in and the input query is empty the search action shows results completely based on the user profile.

Follow some figures showing examples of the advanced query whose results are personalized according to the logged user.

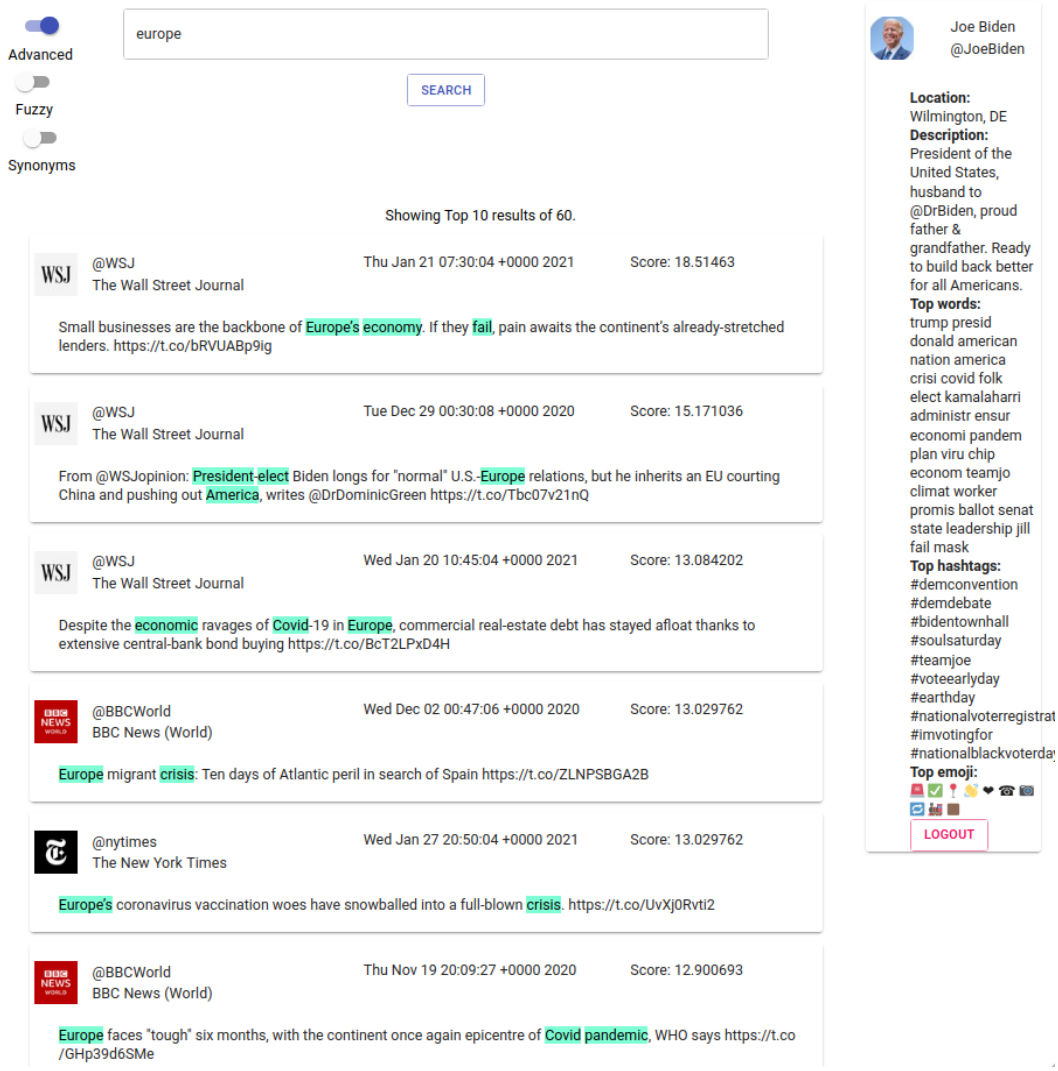

Figure 13: Joe Biden Personalized search for “europe”.

Figure 13 shows Joe Biden’s personalized results obtained by searching for “europe” and it’s possible to detect at least three different topics that are economy, crisis and pandemic which are somehow connected and the relationship between EU and US.

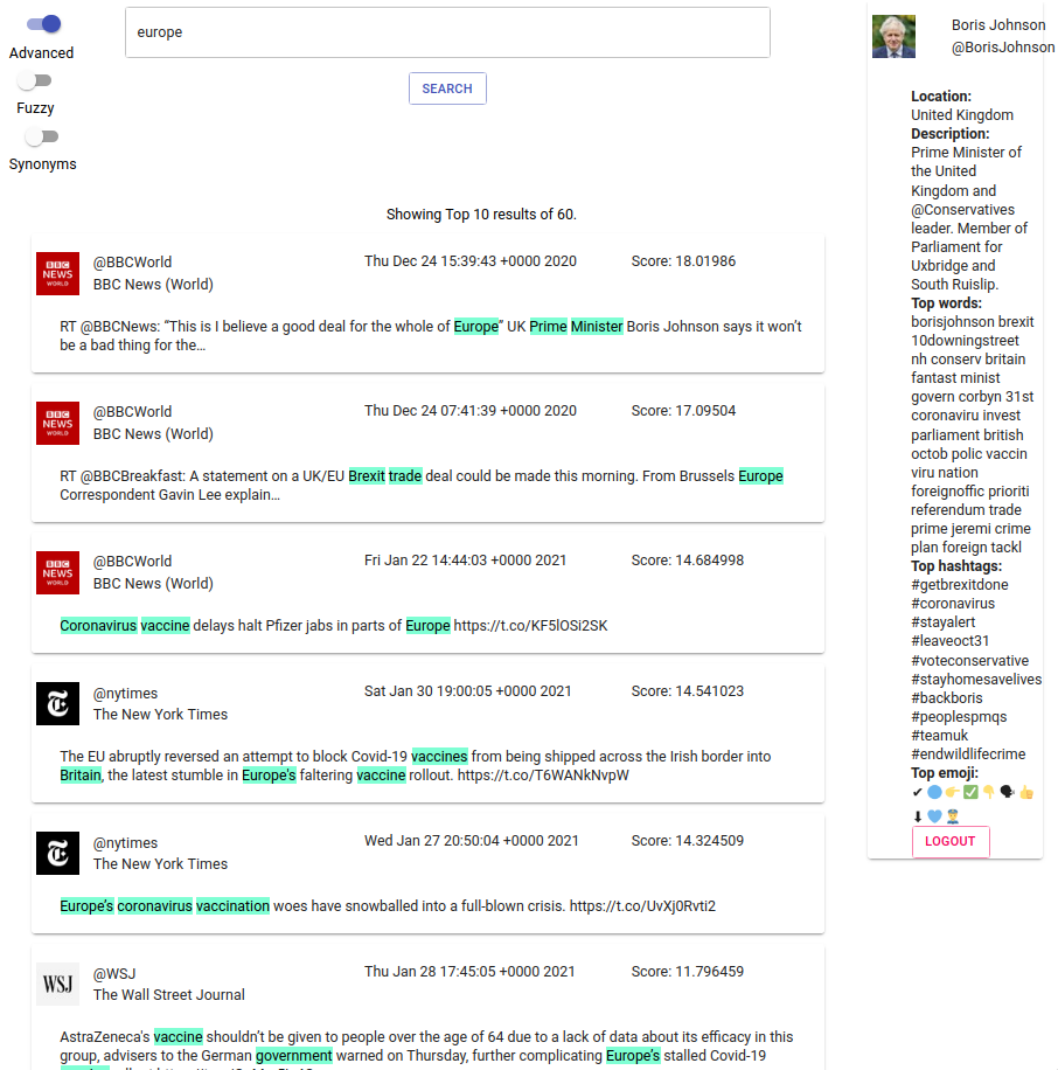

Figure 14: Boris Johnson Personalized search for "europe".

Figure 14 proposes the exact same search of Biden but impersonating Boris Johnson, whose personalized results can be summarized into the topics: Brexit and relationship between EU and UK and pandemic with a really strong focus given to the vaccines.

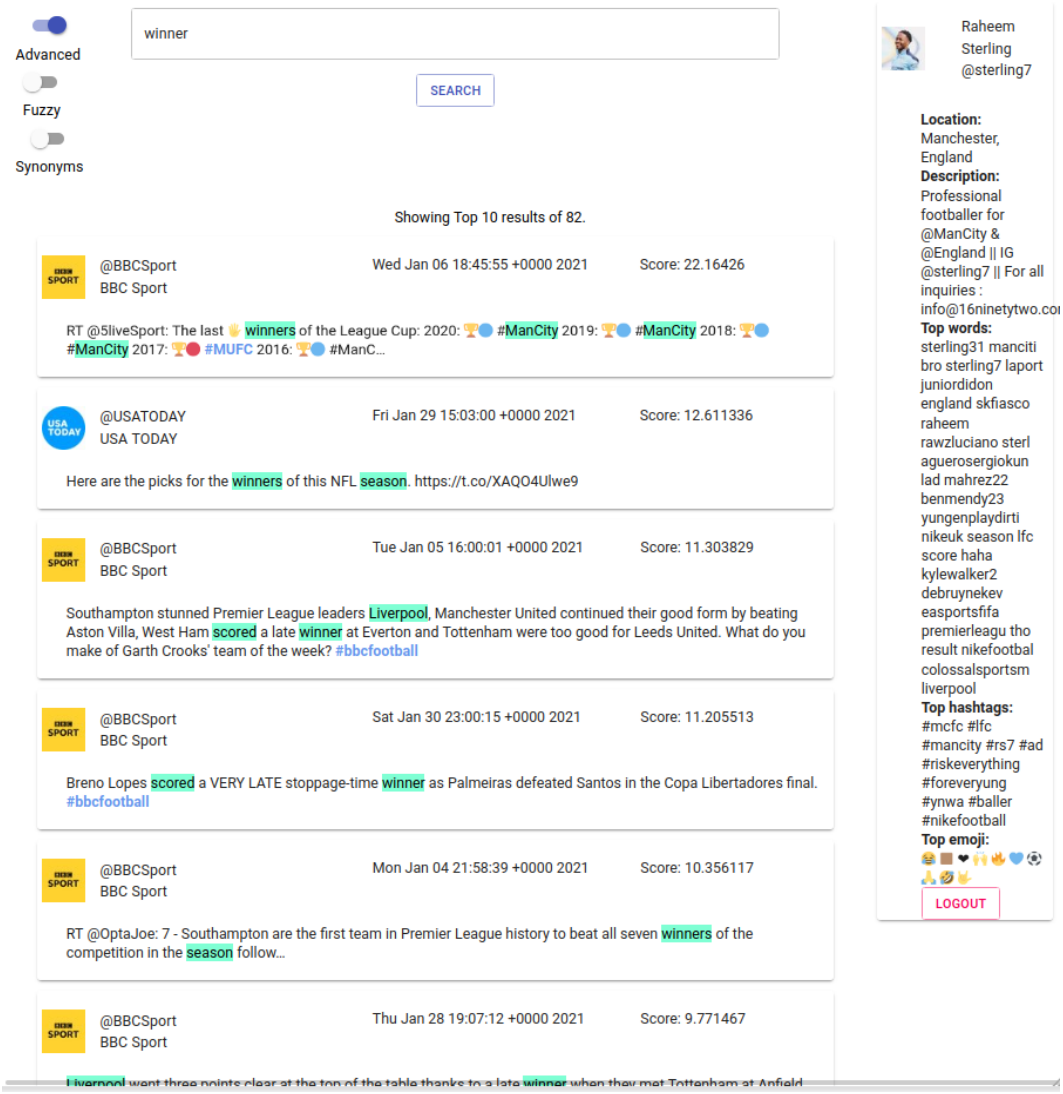

Figure 15: Raheem Sterling Personalized search for "winner".

Figure 15 contains Raheem Sterling personalized results for the query "winner" showing mostly sports related tweets, especially related to sports in the UK. Inside the first result it's possible to see some hashtags matching Sterling's user profile top hashtags.

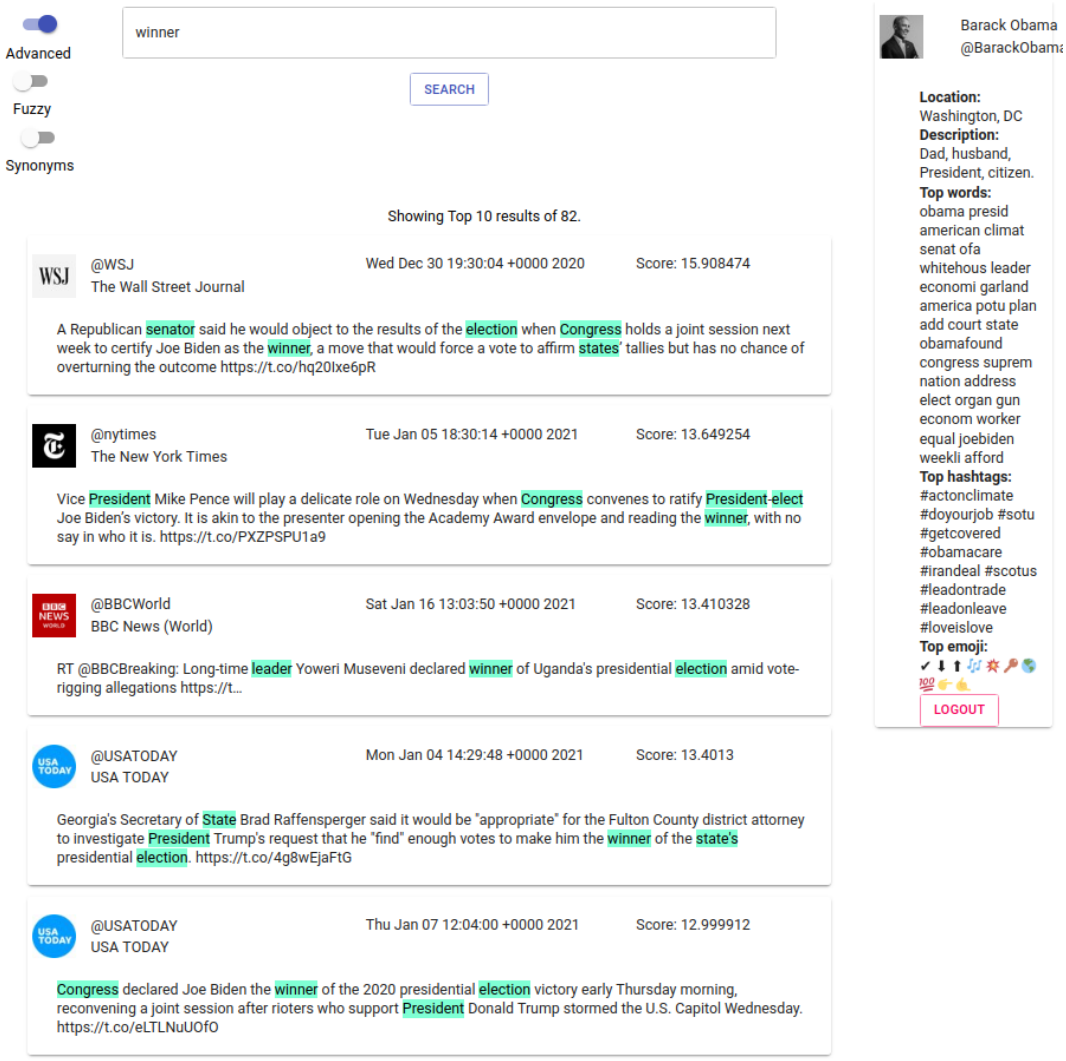

Figure 16: Barack Obama Personalized search for "winner".

Figure 16 proposes again the exact same query of Sterling but this time issued by Barack Obama, whose personalized results strongly deviate from Sterling's ones: it's easy to understand that "winner" for Obama takes a strong political meaning, in fact all the top 5 tweets talk about politics.