Horizon 2020

Understanding Europe's Fashion Data Universe

Demo for Trend Prediction

Deliverable number: D5.5

Version 0.1

Funded by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 699924

Project Acronym:	FashionBrain
Project Full Title:	Understanding Europe's Fashion Data Universe
Call:	H2020-ICT-2016-1
Topic:	ICT-14-2016-2017, Big Data PPP: Cross-sectorial and cross-lingual data integration and experimentation
Project URL:	https://fashionbrain-project.eu

Deliverable type	Report (R)
Dissemination level	Public (PU)
Contractual Delivery Date	31 December 2019
Actual Delivery Date	31 December 2019
Number of pages	21, the last one being no. 17
Authors	Ines Arous, Mourad Khayati - UNIFR Ying Zhang, Panagiotis Koutsourakis, Baptiste Kervaut, Martin Kersten - MDBS
Peer review	Alessandro Checco, Jennifer Dick - USFD

Change Log

Version	Date	Status	Partner	Remarks
0.1	01/06/2018	Draft	UNIFR	
0.2	18/11/2019	MDBS section		

Deliverable Description

This demo will show how a particular fashion trend (style) is detected on fashion time series over time. The prediction will be implemented as an operator in MonetDB. (*** @Ines: we probably should put the description of D5.5 from the DoW here? – Jennie ***)

Abstract

Predicting fashion trends is crucial for many fashion companies as it allows them to remain competitive in the market. Several prediction tools have been proposed in this context, but none of them can properly handle the irregularity and the incompleteness in fashion time series. Moreover, existing tools often can not perform multiple predictions at a time, which helps to simultaneously balance the supply and demand of multiple products. In this deliverable, we present two demo applications in the areas of (fashion) trend detection and prediction.

First, we describe a demo application for trend detection with the combined data processing and text analysis power of a relational database system (i.e. MonetDB) and a machine learning library (i.e. FLAIR) in handling large scale unstructured data (i.e. ∼3TB of 800+ million tweets). This demo is built using the in-database text analysis integration stack reported in deliverable D2.3 "Data Integration Solution".

Then, we describe a tool to predict the upcoming fashion trends. Our tool overcomes the limitations of existing prediction tools and make it possible to perform multiple predictions on irregular fashion series that contain missing values. This deliverable will benefit from the results obtained in D2.4 and D5.3.

Table of Contents

List of Figures **1**

1 Introduction **2**
1.1 Motivation of the Deliverable 2
1.2 Scope of the Deliverable 3

2 Trend Mining **4**
2.1 Background 4
2.2 Data set description 5
2.3 Demo implementation 6
2.4 Results 10

3 Trend Prediction **14**

4 Conclusions **15**

Bibliography **16**

List of Figures

2.1 the FashionBrain Integrated Architecture (FaBIAM) as presented in D2.3 “Data Integration Solution”. The in-database text analysis integration stack is marked by the dashed box. 4
2.2 Demo workflow 7
2.3 Parallel execution of FLAIR jobs 10
2.4 Number of tweets sent and number of topics discussed 11
2.5 Discussions about “fashion” versus “fashion week” per day 12
2.6 Popularity of fashion brands 12
2.7 Popularity of pop artists 13

1 Introduction

(*** @Ines: this deliverable is about demos. The current text is written more in the style for a technical report deliverable. Maybe you want to rephrase some text to shift the focus into describing a demo. I've already adjusted the text for the MDB part into a description of a demo. I'll leave your text alone :) – Jennie ***)

1.1 Motivation of the Deliverable

Time series prediction techniques have been extensively investigated in various domains including signal processing [10], speech processing [9] and market analysis [8, 13, 6]. These techniques have been also used in the fashion industry to predict upcoming trends [11, 4, 12]. Accurate fashion trends prediction is of a great importance to retailers since it allows them to design their production cycle and remain competitive in the market.

Prediction techniques for fashion data can be classified into three main classes. The first class includes basic statistical methods such as linear regression, exponential smoothing, Bayesian analysis [7], as well as more sophisticated ones such as ARIMA, SARIMA and SVR [1]. These methods are simple to implement and are highly efficient. The second class includes artificial neural networks methods such as the Elman recurrent neural networks [15], back-propagation neural networks [17] and Long-short term memory (LSTM) [5]. These network methods achieve better performance than purely statistical methods but require a much longer time. Finally, a number of research works proposed hybrid prediction methods that combine statistical and NN-based techniques, e.g., ARIMA-ANN [16], SARIMA-NN [14] and SVR-NN [3]. The aforementioned prediction techniques fail to achieve a good performance on incomplete and highly irregular time series [2, 7], which are two main properties characterizing time series in the fashion domain.

In this deliverable, we first identify fashion trends by performing an extensive trends analysis using a combined database and machine learning technologies. In this deliverable, we first a demo application for (fashion) trend detection with the combined data processing and text analysis power of a relational database system (i.e. MonetDB) and a machine learning library (i.e. FLAIR) in handling large scale unstructured data (i.e. 3TB of 800 million tweets). This demo is built using the in-database text analysis integration stack reported in deliverable D2.3 "Data Integration Solution".

Then, we describe a new online prediction tool that accurately predicts trends in fashion time series. Through our tool, users will observe prediction of multiple time

series and will be able to test it on fashion time series with different properties.

1.2 Scope of the Deliverable

This deliverable (D5.5) is part of WP5 in which we perform fashion analysis using time series data. D5.5 extends the work in D5.4, where we classify fashion products based on their fashion features and D5.3, where we predict users' fashion preferences. In this deliverable, we focus on the detection of (fashion) trends, and the prediction of the fashion time series using their underlying correlations.

For trends detection, D5.5 showcases the in-database machine learning technology introduced in D2.3 on a large Twitter data set. For trends prediction, D5.5 introduces a new method to predict the evolution of fashion trends accurately. Also, D5.5 contributes to the research challenges about Analytics for Business Intelligence (i.e., Challenge 9: Time Series Analysis) and the Core Technologies about the execution layer (i.e., CT2: Infrastructures for scalable cross-domain data integration and management).

For the trends detection demo, we have used a Twitter data set, which contains 800+ million tweets in the period Jul. 01 - Dec. 31, 2014. The information of each tweet is stored in one JSON object. All tweets from the same day are stored one file. In total, the data set contains 187 JSON files with ∼1 billion JSON objects[1] which take ∼3TB on disk.

To evaluate our prediction tool, deliverable D5.5 uses the datasets provided by Zalando and described in D2.2. These datasets correspond to the sales in two different regions. Each dataset contain 3 main columns: the timestamp, the category of items and the corresponding sale of the category with respect to the timestamp. Similarly to the process used in D5.4, these datasets are transformed to time series where each sale's category represent one time series. The resulting dataset contain 950 time series where the timestamps range from March 2013 to March 2015 with a granularity of 1 day.

[1] Some JSON objects denote the deletion of a tweet. Those were discarded after the preprocessing.

2 Trend Mining

In this section, we describe a demo application we have built for trend detection in Twitter text. This demo is built using the in-database text analysis integration stack reported in deliverable D2.3 "Data Integration Solution". We leverage the data management and analysis power of the relational database system MonetDB[1], and the natural language processing power of the machine learning library FLAIR[2] to jointly analyse the topics discussed in a large Twitter data set to detect trends. In our analysis, we focus on fashion related trends.

2.1 Background

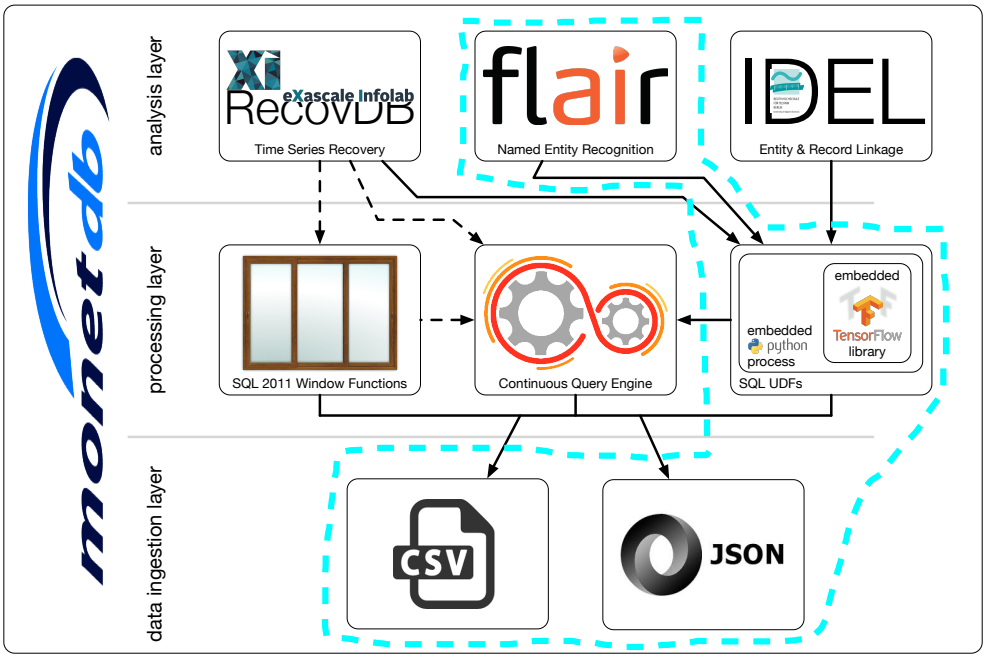


Figure 2.1: the FashionBrain Integrated Architecture (FaBIAM) as presented in D2.3 "Data Integration Solution". The in-database text analysis integration stack is marked by the dashed box.

[1] https://www.monetdb.org
[2] https://github.com/zalandoresearch/flair

Traditionally, DBMSs only have limited text processing ability, such as substring search and regular expression search; while the machine learning community has much more advanced text analysis tools which are capable of, for instance, understanding the meaning of the text, such as "wedding dress" vs. "dress for wedding". To combine the strength of both worlds, we have presented a FashionBrain Integrated Architecture (FaBIAM) in D2.3 "Data Integration Solution", as shown in Figure 2.1. Through embedded SQL user defined functions (UDFs) written in, e.g., Python[3], we can enrich the analytical features of MonetDB with various advanced technologies.

In D2.3 "Data Integration Solution" and D2.4 "Time Series Operators for MonetDB", we have demonstrated the main features of all components of FaBIAM using small synthetic data sets. In the context of task T5.4 "Prediction of Fashion Trends", we have built a demo to showcase the features of the in-database text analytics integration stack (as marked by the dashed-box in Figure 2.1) using a large read-world Twitter data set.

This chapter is further divided as follows. First, in Section 2.2, we give a brief description of the data set. Second, in Section 2.3, we describe how the demo is implemented. Finally, in Section 2.4, we present several fashion trends we have detected in this Twitter data set using our integrated MonetDB-FLAIR stack.

2.2 Data set description

The Twitter data set we have used for this demo contains tweets collected in the period of Jul. 01 - Dec. 31, 2014, which is roughly 1% of all tweets sent in this period. All information of each tweet is stored in one JSON object. The table below shows an anonymised example of one such JSON object containing the information of one tweet:


```
{ "created_at":"Mon Jul 01 00:00:00 +0000 2014",
  "id":000000000000000000,
  "id_str":"000000000000000000",
  "text":"RT @... : ...",
  "source":"<source URL>",
  "truncated":false,
  "in_reply_to_status_id":null,
  "in_reply_to_status_id_str":null,
  "in_reply_to_user_id":null,
  "in_reply_to_user_id_str":null,
  "in_reply_to_screen_name":null,
  "user":{
    "id":00000000,
    "id_str":"00000000",
    "name":"...",
    "screen_name":"...",
    "location":"...",
    "url":null,
    "description":"...",
    "protected":false,
    "verified":false,
    "followers_count":<int>,
    "friends_count":<int>,
    "listed_count":<int>,
    "favourites_count":<int>,
    "statuses_count":<int>,
    "created_at":"??? ??? ?? 00:00:00 +0000 ????",
    "description":"...",
    "protected":false,
    "verified":false,
    "followers_count":<int>,
    "friends_count":<int>,
    "listed_count":<int>,
    "favourites_count":<int>,
    "statuses_count":<int>,
    "created_at":"??? ??? ?? 00:00:00 +0000 ????",
    "utc_offset":<int>,
    "time_zone":"...",
    "geo_enabled":false,
    "lang":"??",
    "contributors_enabled":false,
    "is_translator":false,
    "profile_background_color":"000000",
    "profile_background_image_url":"<URL>",
    "profile_background_image_url_https":"<URL>",
    "profile_background_tile":false,
    "profile_link_color":"000000",
    "profile_sidebar_border_color":"000000",
    "profile_sidebar_fill_color":"000000",
    "profile_text_color":"000000",
    "profile_use_background_image":false,
    "profile_image_url":"<URL>",
    "profile_image_url_https":"<URL>",
    "profile_banner_url":"<URL>",
```


[3] https://www.monetdb.org/blog/embedded-pythonnumpy-monetdb

```
  "utc_offset":<int>,
  "time_zone":"...",
  "geo_enabled":false,
  "lang":"??",
  "contributors_enabled":false,
  "is_translator":false,
  "profile_background_color":"000000",
  "profile_background_image_url":"<URL>",
  "profile_background_image_url_https":"<URL>",
  "profile_background_tile":false,
  "profile_link_color":"000000",
  "profile_sidebar_border_color":"000000",
  "profile_sidebar_fill_color":"000000",
  "profile_text_color":"000000",
  "profile_use_background_image":true,
  "profile_image_url":"<URL>",
  "profile_image_url_https":"<URL>",
  "default_profile":false,
  "default_profile_image":false,
  "following":null,
  "follow_request_sent":null,
  "notifications":null
},
"geo":null,
"coordinates":null,
"place":null,
"contributors":null,
"retweeted_status":{
  "created_at":"??? ??? ?? 00:00:00 +0000 ????",
  "id":000000000000000000,
  "id_str":"000000000000000000",
  "text":"...",
  "source":"<URL>",
  "truncated":false,
  "in_reply_to_status_id":null,
  "in_reply_to_status_id_str":null,
  "in_reply_to_user_id":null,
  "in_reply_to_user_id_str":null,
  "in_reply_to_screen_name":null,
  "user":{
    "id":000000000,
    "id_str":"000000000",
    "name":"...",
    "screen_name":"...",
    "location":"...",
    "url":"<URL>",
    "default_profile":false,
    "default_profile_image":false,
    "following":null,
    "follow_request_sent":null,
    "notifications":null
  },
  "geo":null,
  "coordinates":null,
  "place":null,
  "contributors":null,
  "retweet_count":<int>,
  "favorite_count":<int>,
  "entities":{
    "hashtags":[{"text":"...","indices":[<int>,<int>]}],
    "trends":[],
    "urls":[],
    "user_mentions":[],
    "symbols":[]
  },
  "favorited":false,
  "retweeted":false,
  "possibly_sensitive":false,
  "filter_level":"???",
  "lang":"??"
},
"retweet_count":0,
"favorite_count":0,
"entities":{
  "hashtags":[{"text":"...","indices":[<int>,<int>]}],
  "trends":[],
  "urls":[],
  "user_mentions":[{
    "screen_name":"...",
    "name":"...",
    "id":000000000,
    "id_str":"000000000",
    "indices":[<int>,<int>]
  }],
  "symbols":[]
},
"favorited":false,
"retweeted":false,
"possibly_sensitive":false,
"filter_level":"...",
"lang":"??"
}
```


In total, there are ∼1 billion JSON objects, of which there are 800+ million "actual tweets" (the remaining JSON objects contain information about deleted tweets; they are discarded after the pre-processing).

The tweets of each day are stored in one file. In total, the data set contains 184 JSON files with each file is ∼16GB. The whole data set takes ∼3TB on disk.

2.3 Demo implementation

Figure 2.2 shows the steps in which this demo has been implemented. For this work, we have used machines both from our Scilens database computing cluster[4], as well as several AWS GPU instance[5]. The majority of the work was done using the "Bricks" machines, each contains an Intel Xeon E5-2650 CPU with 32 cores, 256 GB RAM, 5.4 TB HDD. Some of the Bricks machines also has GPU(s), e.g. "Bricks16" has two nVIDIA GeForce GTX 1080 Ti GPUs with 12 GB on-card Memory[6].

[4] https://www.monetdb.org/wiki/Scilens-configuration-standard

[5] We did not make much use of the AWS instances as we have initially planned, because the GPU instances did not give us speedup for the FLAIR jobs and they have too few RAM

[6] Note that this is not a performance benchmark. Nevertheless, we denote the hardware information and later in this section, some execution times just for the record.

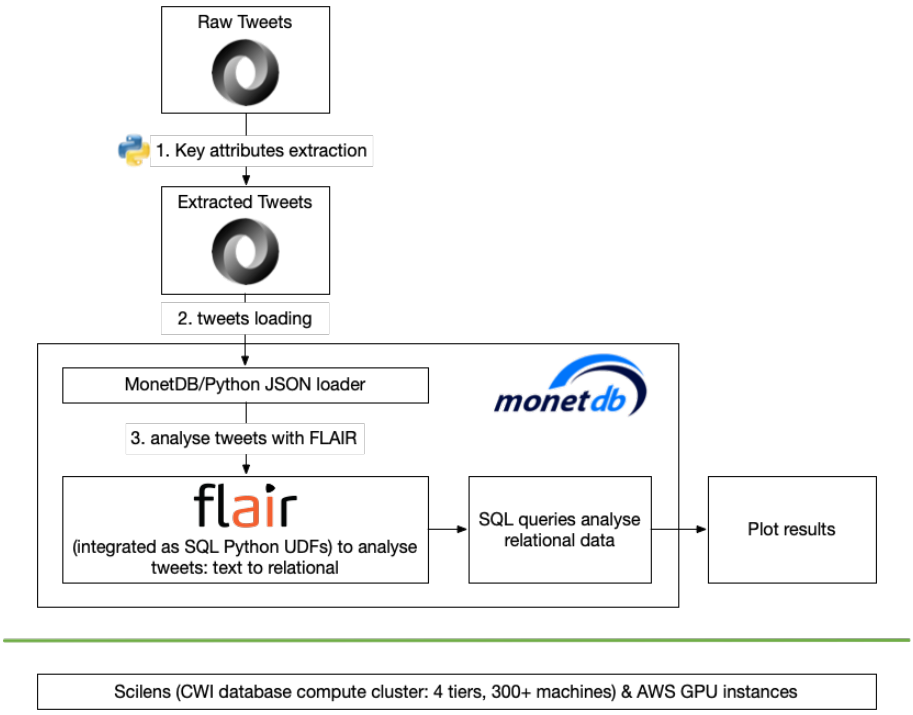


Figure 2.2: Demo workflow

Step 1: key attributes extraction

Because we are mainly interested in using FLAIR to analyse the Twitter texts in this demo, and because we do not want to include personal information due to privacy issues, we first extracted a small number of key attributes from the raw Twitter data (as shown above in Section 2.2) for our further analysis:

Tweet_ID	Date	Language	Retweet_ID	Text

Therefore, we have implemented a Python script, which reads the raw Twitter data from each file, extracts the required attributes, and outputs the values of the attributes as JSON objects, whose format is supported by the MonetDB/Python JSON loader function[7], into a file. The extraction takes 6∼7 minutes per file, hence, ∼20 hours in total. The results files are stored under the names `tweets_{MM}_{DD}.json`.

Initially, we extracted all "actual tweets". However, when we noticed that running FLAIR takes a long time (more on this in "Step 3" below) and many tweets contains languages not supported by FLAIR (e.g. Arabic languages and Asian languages), we decided to only extract tweets in German, English, Spanish, French, Italian, Dutch and Portuguese. This reduces the amount of tweets used in the FLAIR analysis with ∼50%.

[7] https://www.monetdb.org/blog/monetdbpython-loader-functions

The Python script was needed, because at the time when this project was conducted (i.e. summer 2019), MonetDB did not support the loading of the Twitter data in its raw JSON format. After the summer project, we continued to improve MonetDB's JSON support. By the time of this writing (i.e. end of Nov 2019), MonetDB is able to directly load the raw Twitter JSON data.

Step 2: tweets loading

In this step, we loaded all `tweets_{MM}_{DD}.json` files into MonetDB and did some post-loading processing of the data.

First, we created a Python `LOADER` function:

```
CREATE LOADER tweets_loader(filename STRING) LANGUAGE PYTHON {
    import json

    f = open(filename)
    _emit.emit(json.load(f))
    f.close()
};
```

Then the JSON files are loaded one by one:

```
sql>CREATE TABLE tweets FROM LOADER tweets_loader('<path-to>/tweets_json/tweets_07_01.json') ;
operation successful
sql>COPY LOADER INTO tweets FROM tweets_loader('<path-to>/tweets_json/tweets_07_02.json') ;
operation successful
...
sql>COPY LOADER INTO tweets FROM tweets_loader('<path-to>/tweets_json/tweets_12_30.json') ;
operation successful
sql>COPY LOADER INTO tweets FROM tweets_loader('<path-to>/tweets_json/tweets_12_31.json') ;
operation successful
```

The "Date" strings from the raw Twitter data are not recognised by MonetDB as timestamps. So, after the initial loading, those "Date" strings are stored in a character large object (CLOB) column, which occurs more disk storage and is less efficient in query processing than when the values are stored as binary timestamps. Therefore, as a final step in data loading, we cast the "Date" strings into timestamps:

```
ALTER TABLE tweets ADD COLUMN "TimeStamp" TIMESTAMP;
UPDATE tweets SET "TimeStamp" = STR_TO_TIMESTAMP("Date", '%a %b %d %H:%M:%S +0000 %Y');
ALTER TABLE tweets DROP COLUMN "Date";
```

On average, loading one `tweets_{MM}_{DD}.json` file takes ∼25 seconds. Converting the "Date" strings took ∼17 minutes. In total, this step took ∼1.5 hour. The final result of this step is a table called "tweets" with ∼400 million tuples table with a disk size of ∼68 GB:

```
sql>\d tweets
CREATE TABLE "sys"."tweets" (
    "Tweet_ID"   BIGINT,
    "Language"   CHARACTER LARGE OBJECT,
    "Retweet_ID" CHARACTER LARGE OBJECT,
    "Text"       CHARACTER LARGE OBJECT,
    "TimeStamp"  TIMESTAMP
);
sql>SELECT COUNT(*) FROM tweets ;
+-----------+
```

```
| L2         |
+============+
| 404578963  |
+------------+
1 tuple
sql>SELECT (SUM("count" * typewidth) + SUM(heapsize))/1024/1024/1024
more>FROM sys.storage() WHERE table = 'tweets';
+------+
| L5   |
+======+
|   68 |
+------+
1 tuple
```

Step 3: Analyse tweets with FLAIR

Finally, we use FLAIR[8] to process the tweet texts, i.e. extract the topics discussed in them. As shown in Figure 2.1, FLAIR was integrated into MonetDB as embedded SQL Python UDFs[9], which was described in detail in D2.3 "Data Integration Solution".

We have experienced with various FLAIR models. First, we trained our own models with the help of the Flair tutorials[10]. We trained a Named-Entity Recognition (NER) model and a multilingual Part-of-Speech (POS) model. The accuracy of the results were low, because tweets only contain short texts and are extremely messy with a lot of grammatical errors and shorthands. Then, we used two models provided by Flair: a multilingual NER model and a multilingual POS model, which gave us better results. In particular, the pre-trained POS-multi model was quite good in tagging of nouns and proper_nouns.

The code snippet below shows one such SQL Python UDFs, which uses the named-entity recognition (NER) model (line 11 - 14) on the input `s STRING`. It returns the `entities` found by FLAIR together with the timestamp associated with the input string and some auxiliary information of this entity (line 30). Finally, we apply this function on the text from the `tweets table` and store the results in a new table `entities` (line 32).

```
 1 CREATE FUNCTION flair_entity_tags (ts TIMESTAMP, s STRING)
 2 RETURNS TABLE(ts TIMESTAMP, entity STRING, tag STRING, start_pos INT, end_pos INT)
 3 LANGUAGE python
 4 {
 5   from flair.data import Sentence
 6   from flair.models import SequenceTagger
 7   import numpy
 8
 9   # Make sentence objects from the input strings
10   sentences = [Sentence(sent, use_tokenizer=True) for sent in s]
11   # load the NER tagger
```

[8] A. Akbik, D. Blythe and R. Vollgraf. Contextual String Embeddings for Sequence Labeling, in Proceedings of the 27th International Conference on Computational Linguistics, pages 1638–1649, August, 2018. Santa Fe, New Mexico, USA.

[9] https://www.monetdb.org/blog/embedded-pythonnumpy-monetdb

[10] https://github.com/zalandoresearch/flair

```
12   tagger = SequenceTagger.load('ner')
13   # run NER over sentences
14   tagger.predict(sentences)
15
16   tss = []
17   entities = []
18   tags = []
19   start_poss = []
20   end_poss = []
21
22   for idx,sent in numpy.ndenumerate(sentences):
23     for e in sent.get_spans('ner'):
24         tss.append(ts[idx[0]])
25         entities.append(e.text)
26         tags.append(e.tag)
27         start_poss.append(e.start_pos)
28         end_poss.append(e.end_pos)
29
30   return [tss, entities, tags, start_poss, end_poss]
31 };
32 CREATE TABLE entities AS SELECT * FROM flair_entity_tags((SELECT "TimeStamp", "Text" FROM tweets));
```

Even though we have worked with colleagues from Zalando research to speed up FLAIR, applying the FLAIR models on this amount of tweets sequentially is still a very time consuming task which can take weeks. Fortunately, multiple Bricks machines have GPUs[11]. So, eventually, we decided to divide the `tweets` table into several smaller pieces and run multiple FLAIR UDFs in parallel, as shown in Figure 2.3. We grabbed all Bricks machines that were still available in August 2019 with three of them having 2 GPUs and one having 1 GPU. On each GPU, we could run 2 FLAIR functions. In this way, we were able to speed up the FLAIR execution time with a factor of 14.

By using the FLAIR models, we identified in total ~1.7 billion topics discussed in the tweets of this Twitter data set. All topics are united into a view called `all_topics` that only contains the triple `(dte, hr, topic)` to denote the date, hour-of-day of a topic.

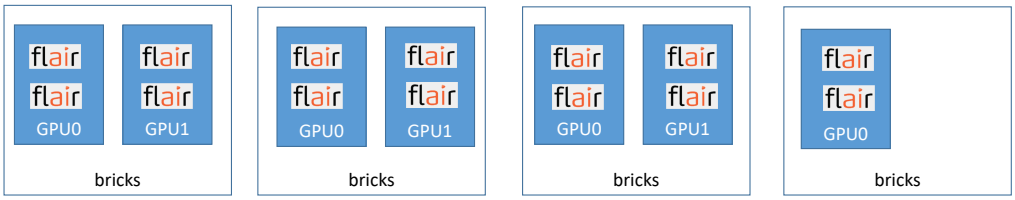


Figure 2.3: Parallel execution of FLAIR jobs

2.4 Results

With the query processing power of an analytical DBMS and the text analysis power of a multilingual machine learning library, we can gain a lot of information from this

[11] https://www.monetdb.org/wiki/Bricks

Twitter data set. Below, we limit the presented results to several fashion related topics without involving any personal information (of Twitter users).

(a) per day (b) per hour-of-day

Figure 2.4: Number of tweets sent and number of topics discussed

Trend 1: how active are we on Twitter?

Figure 2.4(a) and Figure 2.4(b) show the number of tweets sent and number of topics discussed per day or per hour-of-day, respectively.

The amount of activities on Twitter is more or less stable in this period. The spike on November 9th is possibly caused by several popular events happening at the same time, e.g. some news about Justin Bieber and the plethora of fans campaigning to get the girl group Fifth Harmony the top prize in the 2014 MTV European Music Awards (see also discussion below about Figure 2.7(a) and Figure 2.7(b)). Unfortunately, we cannot find any good explanation for the negative spike in December 2014.

Figure 2.4(b) shows that Twitter users are most active from 5 o'clock in the afternoon until 3 o'clock in the morning, and lest active earlier in the morning. This is as what we would expect.

Trend 2: how often do we discuss about fashion?

Figure 2.5 shows that the interest in fashion among the Twitter users is fairly stable over time. Some of the discussions about fashion is devoted to discussing the fashion week, a fashion event organised several times per year in different locations for a whole week. Throughout the year, there is a small amount of tweets about fashion week. However, during the week of an event, this topic obvious attracts more attention, such as shown by the peak in September 2014, which was when the New York Fashion Week was hold.

Trend 3: which brand should I buy to be more trendy?

Figure 2.6(a) and Figure 2.6(b) show the popularities of several luxury fashion brands and sport brands, respectively.

According to this data set, Gucci is far more popular than Prada and Burberry. Also, the popularity of the three brands barely overlaps. The amount of interest in Gucci fluctuates considerably over time, while the interests in Prada and Burberry

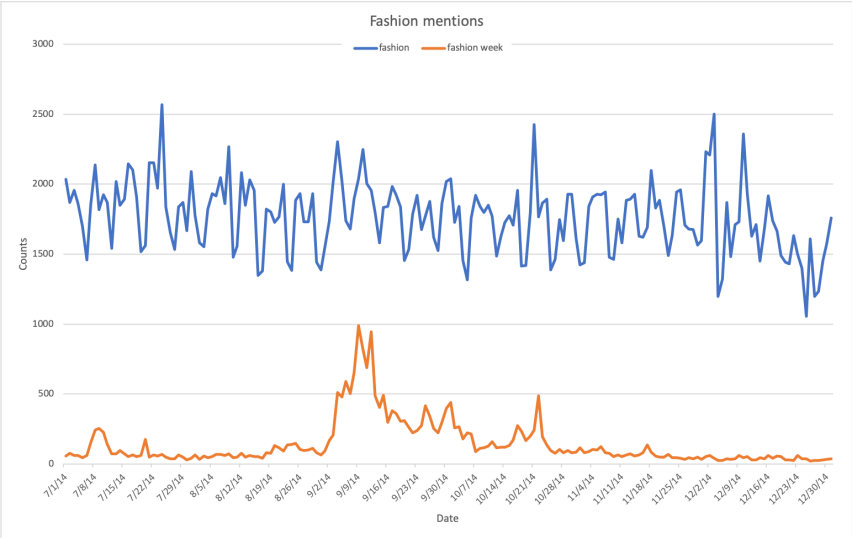
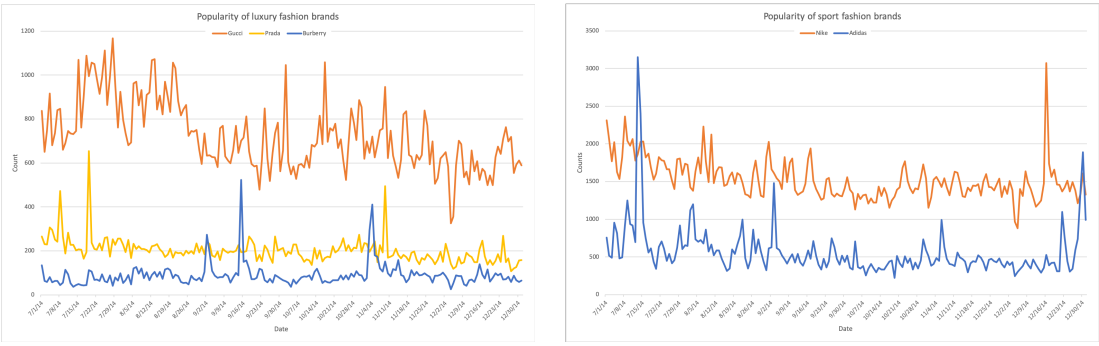


Figure 2.5: Discussions about "fashion" versus "fashion week" per day



(a) Luxury brands (b) Sport brands

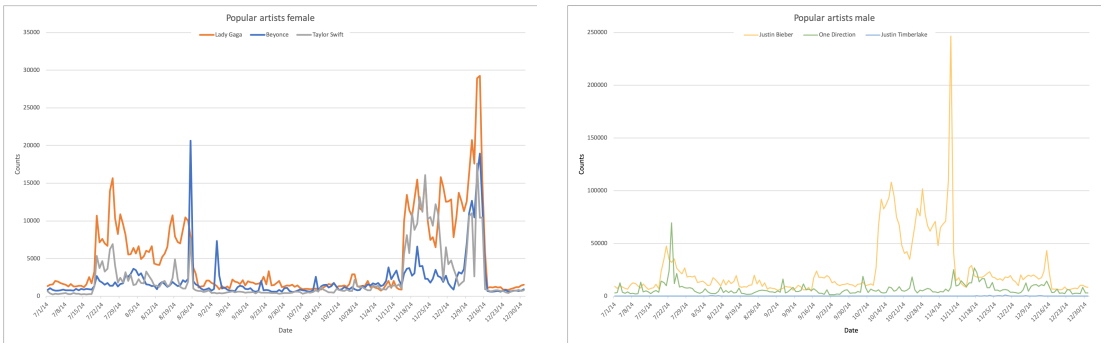
Figure 2.6: Popularity of fashion brands

are fairly stable. Spikes in the graph, e.g. those of Prada and Burberry, are generally caused by marketing events. For instance, on Sep. 15th, it was the Full Burberry Prorsum Womenswear S/S15 Show; while on Nov. 19th, the Pradasphere exhibition started in HongKong.

Figure 2.6(b) shows that Nike is generally more popular than Adidas. However, with some strong market, Adidas some times attracts more attention than Nike. For instance, on Jul. 13th, Adidas release the adiZero Prime Boost in the US; and on Dec. 30th, Adidas released the T-MAC 3 "Chinese New Year".

Trend 4: who might have the biggest influence?

By studying the popularity, we can get an idea of how much impact a famous person can have when they do something (i.e. find influencers). Figure 2.7(a) and Figure 2.7(a) show the popularity of several female and male music artists,



(a) Female artists (b) Male artists

Figure 2.7: Popularity of pop artists

respectively. From these graphs, we can observe several things. First, the amount of interest the female artists attract was largely affected by famous music event. In Figure 2.7(a), the peak in July - August goes with the 2014 MTV video music awards; while the peak in November goes with the 2014 American music awards. However, according to Figure 2.7(b), the interests in male artists do not show such clear correlation (or maybe they are a bit overshadowed by the news around Justin Bieber).

Second, we already know that Beyonce has not been very active in the recent years, so the social media seems to be a bit quiet around her. However, one live performance of her at the 2014 MTV Video Music Awards on Aug. 25th drove her popularity right through the platform formed by Lady Gaga. Clearly, Beyonce still has a lot of fans. From this, we can probably conclude that as an artist/influencer, Beyonce is a hibernating giant. Once she is back to action, we can expect huge impact from her.

Finally, by comparing the scale of the y-axis of both figures, one can see that the male artists attract by far more attention from the Twitter users than the female artists. This is probably because there is a huge amount of teenage girls on Twitter, which drives the popularities of teenage artists and bands. The huge spike in Justin Bieber's graph was probably caused by that on November 8th, several pictures of Justin Bieber and Hailey Baldwin were put online.

3 Trend Prediction

(*** MK: UNIFR part ***)

4 Conclusions

Bibliography

[1] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., San Francisco, CA, USA, 1990. ISBN 0816211043.

[2] Mu-Yen Chen and Bo-Tsuen Chen. A hybrid fuzzy time series model based on granular computing for stock price forecasting. *Information Sciences*, 294: 227–241, 2015.

[3] Xuejun Chen, Shiqiang Jin, Shanshan Qin, and Laping Li. Short-term wind speed forecasting study and its application using a hybrid model optimized by cuckoo search. *Mathematical Problems in Engineering*, 2015, 2015.

[4] Tsan-Ming Choi, Chi-Leung Hui, and Yong Yu. Intelligent time series fast forecasting for fashion sales: A research agenda. In *2011 International Conference on Machine Learning and Cybernetics*, volume 3, pages 1010–1014. IEEE, 2011.

[5] Daniel Hsu. Time series forecasting based on augmented long short-term memory. *arXiv preprint arXiv:1707.00666*, 2017.

[6] Bin Li and Steven Chu Hong Hoi. *Online portfolio selection: principles and algorithms*. Crc Press, 2018.

[7] Na Liu, Shuyun Ren, Tsan-Ming Choi, Chi-Leung Hui, and Sau-Fun Ng. Sales forecasting for fashion retailing service industry: a review. *Mathematical Problems in Engineering*, 2013, 2013.

[8] Marc Nerlove, David M Grether, and Jose L Carvalho. *Analysis of economic time series: a synthesis*. Academic Press, 2014.

[9] Lawrence Rabiner and RW Schafer. Digital speech processing. *The Froehlich/Kent Encyclopedia of Telecommunications*, 6:237–258, 2011.

[10] José Luis Rojo-Álvarez, Manel Martínez-Ramón, Mario de Prado-Cumplido, Antonio Artés-Rodríguez, and Aníbal R Figueiras-Vidal. Support vector method for robust arma system identification. *IEEE transactions on signal processing*, 52(1):155–164, 2004.

[11] Roberto Sanchis-Ojeda, Daragh Sibley, and Paolo Massimi. Detection of fashion trends and seasonal cycles through the analysis of implicit and explicit client feedback. In *KDD Fashion Workshop*, 2016.

[12] Pawan Kumar Singh, Yadunath Gupta, Nilpa Jha, and Aruna Rajan. Fashion retail: Forecasting demand for new items. *arXiv preprint arXiv:1907.01960*, 2019.

[13] Ruey S Tsay. Financial time series. *Wiley StatsRef: Statistics Reference Online*, pages 1–23, 2014.

[14] Fang-Mei Tseng, Hsiao-Cheng Yu, and Gwo-Hsiung Tzeng. Combining neural network model with seasonal time series arima model. *Technological forecasting and social change*, 69(1):71–87, 2002.

[15] Jie Wang, Jun Wang, Wen Fang, and Hongli Niu. Financial time series prediction using elman recurrent random neural networks. *Computational intelligence and neuroscience*, 2016, 2016.

[16] Li Wang, Haofei Zou, Jia Su, Ling Li, and Sohail Chaudhry. An arima-ann hybrid model for time series forecasting. *Systems Research and Behavioral Science*, 30(3):244–259, 2013.

[17] Chih-Chieh Young, Wen-Cheng Liu, and Wan-Lin Hsieh. Predicting the water level fluctuation in an alpine lake using physically based, artificial neural network, and time series forecasting models. *Mathematical Problems in Engineering*, 2015, 2015.