DATA WAREHOUSES
AND ANALYTICAL PROCESSING

# Apache Drill

Piotr Jówko, Wojciech Mikołajczyk, Bartosz Wysocki

Poznań 2017 r.

# Table of Contents

# 1. Project description

The purpose of this project is to explore Apache Drill technology. We wanted to check functionality of this framework for analysis of big NoSql data. As part of the project, we analyzed various queries and configured the environment to be able to work on single node. Then we made analysis of some standard functionalities, like querying, grouping and joins. The source of our data were tweets, which we extracted from twitter api by search terms based on two keywords: Trump, Clinton.

## 1.1. Apache Drill

Apache Drill is an open-source software framework that supports data-intensive distributed applications for interactive analysis of large-scale datasets. Drill supports Hadoop, MongoDB, HBase and a few cloud storage systems. Drill is able to scale to 10,000 servers or more and to process petabytes of data and trillions of records in seconds. A single query can join data from multiple datastores. Drill's datastore-aware optimizer automatically restructures a query plan to leverage the datastore's internal processing capabilities.

Features:

- Schema-free JSON document model similar to MongoDB and Elasticsearch, without requiring a formal schema to be declared
- Industry-standard APIs: ANSI SQL, ODBC/JDBC, RESTful APIs
- Extremely user and developer friendly
- Pluggable architecture enables connectivity to multiple datastores

Queries in Apache Drill are written in ANSI SQL. List of sql commands is available in: https://drill.apache.org/docs/supported-sql-commands/

Select command support similar clauses like sql counterpart. List of supported clauses is available in: https://drill.apache.org/docs/select/

Apache Drill also have own data types listed here: https://drill.apache.org/docs/supported-data-types/

Apache Drill have many embedded functions listed here:

https://drill.apache.org/docs/sql-functions/

https://drill.apache.org/docs/sql-window-functions/

Apache Drill uses Apache Licence 2.0.

Apache Drill is still in Developments phase and many features are still not implemented.

# 2. Apache Drill installation

Apache Drill supports embedded and distributed mode. In embedded mode Apache Drill works only on local computer. Embedded mode allows for quick installation without any configuration tasks. In distributed mode drill works on multiple nodes. Distributed mode requires Zookeeper quorum. There is no information in documentation about distributed mode on Windows. Probably this isn't supported.

## 2.1. Installation on Windows in Embedded mode

1. Install Oracle JDK 8 or newer and set JAVA_HOME and PATH environment variables to proper directory where java was installed.
2. Download and extract Apache Drill: https://drill.apache.org/download/
3. Go to apache drill/bin directory.
4. Open cmd.exe and execute command:

```
sqlline.bat -u "jdbc:drill:zk=local"
```

Now Apache Drill works in embedded mode. This means that you perform queries only on local computer. You can execute drill queries in console or go to http://localhost:8047/query and execute queries from there.

## 2.2. Installation on Ubuntu in Embedded mode

1. Install Oracle JDK 8 or newer.
2. Open terminal and run:

```
wget http://apache.mirrors.hoobly.com/drill/drill-1.10.0/apache-drill-1.10.0.tar.gz
```

3. Extract drill with command:

```
tar -xvzf <.tar.gz file name>
```

4. Go to bin folder and run drill-embedded.

## 2.3. Installation on Ubuntu in Distributed mode

To install Apache Drill in distributed mode it is required to have Running Oracle JDK and a ZooKeeper quorum. Then it is possible to configure and install Drill by following these steps:

1. Install Apache Drill like on Ubuntu in embedded mode (don't run drill-embedded)
2. Configure conf/drill-override.conf file

```
drill.exec:{
 cluster-id: "<mydrillcluster>",
 zk.connect: "<zkhostname1>:<port>,<zkhostname2>:<port>,<zkhostname3>:<port>"
}
```

Example from https://drill.apache.org/docs/installing-drill-on-the-cluster/

3. Start Drillbit (Drill daemon) on each node

```
bin/drillbit.sh start
```

We met some problems with running Apache Drill in distributed mode. ZooKeeper configuration went wrong and drill nodes didn't connect with each other, but they were running in standalone mode. For this reason we have skipped testing Drill in distributed mode.

# 3. Tweets

## 3.1. Tweets downloading

Downloading of tweets requires creating a developer account in the twitter api. We have created a program that uses streaming api to extract tweets. To get tweets from stream, you need to specify query by with they will be extracted. Tweets appear in streaming api after short time when they were created by users. Consumer_key, consumer_secret, access_token, access_token_secret are required to access twitter api. You can generate them on your twitter developer account: https://apps.twitter.com . Below is our code used to download tweets:

```
import tweepy
from tweepy
import OAuthHandler
from tweepy.streaming
import StreamListener
from tweepy
import OAuthHandler
from tweepy
import Stream

consumer_key = 'some_consumer_key'
consumer_secret = 'some_consumer_secret'
access_token = 'some_access_token'
access_token_secret = 'some_access_token_secret'

class StdOutListener(StreamListener):

  def on_data(self, data):
  print(data)
return True

def on_error(self, status):
  print(status)

# auth
for twitter
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

listener = StdOutListener()

def get_from_stream():
  while True:
    try:
    stream = Stream(auth, listener)
stream.filter(track = ['trump', 'clinton'])
except:
    continue

get_from_stream()
```

For more information about twitter api, you could visit twitter official documentation: https://dev.twitter.com/docs

## 3.2. Tweet structure

Tweets used in our analysis have JSON format. Single tweets contains hundreds of different properties. Below we listed sample of tweet, with contains only most important properties on with analysis was performed.

```
{
  "created_at":"Thu Apr 06 06:59:29 +0000 2017",
  "text":"RT @Ziggy_Daddy: Trump blamed Obama for use of chemical weapons in #Syria, but Buzzfeed reports
pentagon officials believe #Assad did it to\u2026",
  "source":"\u003ca href=\"http:\/\/twitter.com\/download\/android\" rel=\"nofollow\"\u003eTwitter for
Android\u003c\/a\u003e",
  "user":{
    "location":null,
    "time_zone":null,
    "lang":"en",
  },
  "lang":"en",
}
```

Property created_at contains date on with tweet was created. text contains text of the tweet. Source contains html link to source of the tweet(for example android, ipad, twitter.com). lang contains information about tweet and user language. location is user specified text field, in with he writes where he lives. time_zone stores text information about time zone of user. Detailed information about tweet structure can be found in official api overview: https://dev.twitter.com/overview/api

# 4. Tweets analysis

Analysis was performed on the one local computer(drill in embedded mode). Data was stored locally in JSON files. We used sql queries to perform analysis. We started with basic queries and after that we performed more advanced queries with grouping and joining. At the end, we created some visualisations of results.

## 4.1. Basic queries

**Count example**

```
SELECT COUNT(*) as total_tweets from dfs.`path_to_catalog_with_tweets`;
```

Time in embedded mode: 1 row selected (122.459 seconds), 20,2GB

```
+---------------+
| total_tweets  |
+---------------+
| 3455550       |
+---------------+
```

**Average tweet length**

```
SELECT avg(length(t.text)) as `tweet_length` from dfs.`path_to_catalog_with_tweets` t;
```

Time in embedded mode: 1 row selected (124.234 seconds), 20,2GB

```
+------------------------+
|      tweet_length      |
+------------------------+
| 119.78667145190614                |
+------------------------+
```

**Selecting tweets between specified date**

Apache Drill supports different date formats and some date functions, but date format used in tweets is not compatible with existing functions. We had to extract date manually. The following query takes the day, hour, minute, and second of the date when tweet was created. Then this string is converted to Integer. Date can not be empty to avoid exception NumberFormatException. The result number is in ddhhmmss format. The following example retrieves the date and the content of the tweet from March 13, between 16:45 and 16:55.

```
alter session set `drill.exec.functions.cast_empty_string_to_null`=true;

SELECT t.created_at, t.text
FROM dfs.`path_to_catalog_with_tweets` t
where t.created_at IS NOT NULL

and substring(t.created_at, 5, 3) like 'Mar'

and cast(concat(
substring(t.created_at, 9, 2),
substring(t.created_at, 12, 2),
substring(t.created_at, 15, 2),
substring(t.created_at, 18, 2)) as INT) <= 13165500
and cast(concat(

substring(t.created_at, 9, 2),
substring(t.created_at, 12, 2),
substring(t.created_at, 15, 2),
substring(t.created_at, 18, 2)) as INT) >= 13164500
```

## 4.2. Grouping

**Average tweet length grouped by language**

```
SELECT avg(length(t.text)) as `tweet_length`, t.lang
from dfs.`path_to_catalog_with_tweets` t
where t.lang is not null group by t.lang order by `tweet_length` desc;
```

Time in embedded mode: 57 rows selected (114.119 seconds), 20,2GB

```
+-----------------------+-------+
|      tweet_length      | lang  |
+-----------------------+-------+
| 138.0                 | pa    |
| 136.0                 | ka    |
| 129.9094              | th    |
| 128.82875             | lt    |
| 128.0                 | or    |
| 124.54545             | ta    |
| 121.47067             |en     |
| 120.5                 | gu    |
| 118.47352             | it    |
| 118.22656             | ur    |
| 117.98173             | sl    |
| 117.27272             | bn    |
| 116.36516             | ko    |
| 116.03885             | fr    |
| 115.97178              | es     |
| 115.62651              | tr     |
| 115.24889              | de     |
| 114.04705              | ar     |
| 113.6                  | kn     |
| 113.25                 | km     |
| 111.90271              | nl     |
| 111.58990              | ja     |
| 111.44444              | bg     |
| 109.69686              | fi     |
| 109.51202              | pt     |
| 108.85714              | te     |
| 108.77645              | ru     |
| 108.37121              | fa     |
| 106.17472              | pl     |
| 106.14285              | sr     |
| 106.08439              | sv     |
| 105.21724              | hi     |
| 105.01980              | el     |
| 104.5                  | my     |
| 104.48930              | vi     |
| 103.99787              | ro     |
| 99.412698              | hu     |
| 99.0                   | ckb    |
| 98.571428              | cs     |
| 97.294117              | iw     |
| 96.666666              | ne     |
| 96.5                   | mr     |
| 95.730909              | da     |
| 95.262793              | no     |
| 94.2                   | ml     |
| 94.0                   | am     |
| 92.718805              | in     |
| 91.663430              | zh     |
| 90.642201              | et     |
| 84.768115              | lv     |
| 80.153846              | uk     |
| 75.471698              | is     |
| 67.955752              | cy     |
| 65.814250              | und    |
| 63.469879              | tl     |
| 60.581081              | eu     |
| 48.974550              | ht     |
+------------------------+-------+
```

### Tweet count grouped by location

```
SELECT t.`user`.location as location, count(*)
FROM dfs.`path_to_catalog_with_tweets` t
group by t.`user`.location order by count(*) desc;
```

Time in embedded mode: 208,065 rows selected (153.63 seconds), 20,2GB

First 25:

```
+---------------------------+----------+
|          location         |  EXPR$1  |
+---------------------------+----------+
| null                   | 1219065  |
| United States          | 98783    |
| USA                    | 35644    |
| California, USA        | 26153    |
| Washington, DC         | 22892    |
| Florida, USA           | 20847    |
| India                  | 19743    |
| Texas, USA             | 18722    |
| New York, NY           | 15360    |
| New York, USA          | 14316    |
| Los Angeles, CA        | 13859    |
| Estados Unidos         | 12631    |
| London, England        | 11334    |
| New York               | 10720    |
| Chicago, IL            | 9247     |
| Pennsylvania, USA      | 9054     |
| Texas                  | 8404     |
| Canada                 | 8213     |
| London                 | 8073     |
| North Carolina, USA    | 8019     |
| Michigan, USA          | 7861     |
| New Jersey, USA        | 7583     |
```

**Tweet count grouped by time zone and tweet source(devices)**

```
SELECT t.`user`.time_zone as country, t.source, count(*)
FROM dfs.`path_to_catalog_with_tweets` t
group by t.`user`.time_zone, t.source having t.`user`.time_zone is not null;
```

Time in embedded mode: 11,358 rows selected (127.134 seconds), 20,2GB

Most of the results were blogs and unpopular mobile apps. On single node with condition "count(*) > 100" were only 785 rows. Sources were stored in html link with directed to source website.

Advanced grouping:

For now advanced operators are not supported. It is planned to add such features in future releases: https://issues.apache.org/jira/browse/DRILL-3962

## 4.3. Joins and sets

**Simple join with countries.json**

```
SELECT t.lang, c.code
from dfs.`path_to_catalog_with_tweets` t
JOIN dfs.`path_to_file_countries` c
ON LOWER(t.lang) = LOWER(c.code);
```

Time in embedded mode: 219,896 rows selected (124.282 seconds), 20,2GB

**Join with grouping**

```
SELECT t.lang, c.code, count(*)
from dfs.`path_to_catalog_with_tweets` t
JOIN dfs.`path_to_file_countries` c
ON LOWER(t.lang) = LOWER(c.code)
group by t.lang, c.code;
```

Time in embedded mode: 39 rows selected (308.711 seconds), 50,5GB

**Union example**

```
SELECT t.lang, t.text
FROM dfs.`path_to_catalog_with_tweets` t
where t.lang is not null and t.lang = 'pl'
UNION
SELECT t.lang, t.text
FROM dfs.`path_to_catalog_with_tweets` t
where t.lang is not null and t.lang = 'fr';
```

Time in embedded mode: 58,870 rows selected (622.649 seconds), 50,5GB

**Union and grouping**

```
SELECT t.lang, count(*)
FROM dfs.`path_to_catalog_with_tweets` t
where t.lang is not null and t.lang like 'p%'
group by t.lang
UNION
SELECT t.lang, count(*)
FROM dfs.`path_to_catalog_with_tweets` t
where t.lang is not null and t.lang like 't%'
group by t.lang;
```

Time in embedded mode: 8 rows selected (306.956 seconds)

```
+-------+---------+
| lang  | EXPR$1  |
+-------+---------+
| pt    | 45055   |
| tr    | 33811   |
| th    | 3247    |
| tl    | 3772    |
| ta    | 28      |
| pa    | 1       |
| te    | 16      |
| pl    | 3426    |
+-------+---------+
```

**Intersect and Except**

We also wanted to test other operators on sets, but they are not supported yet. There are plans to create such features in future releases:

https://issues.apache.org/jira/browse/DRILL-4092

## 4.4. Results visualisations

To create results visualisation, we inserted query results to table and exported to csv format. Then we created charts in Microsoft Excel 2007. All this queries were performed on single node.

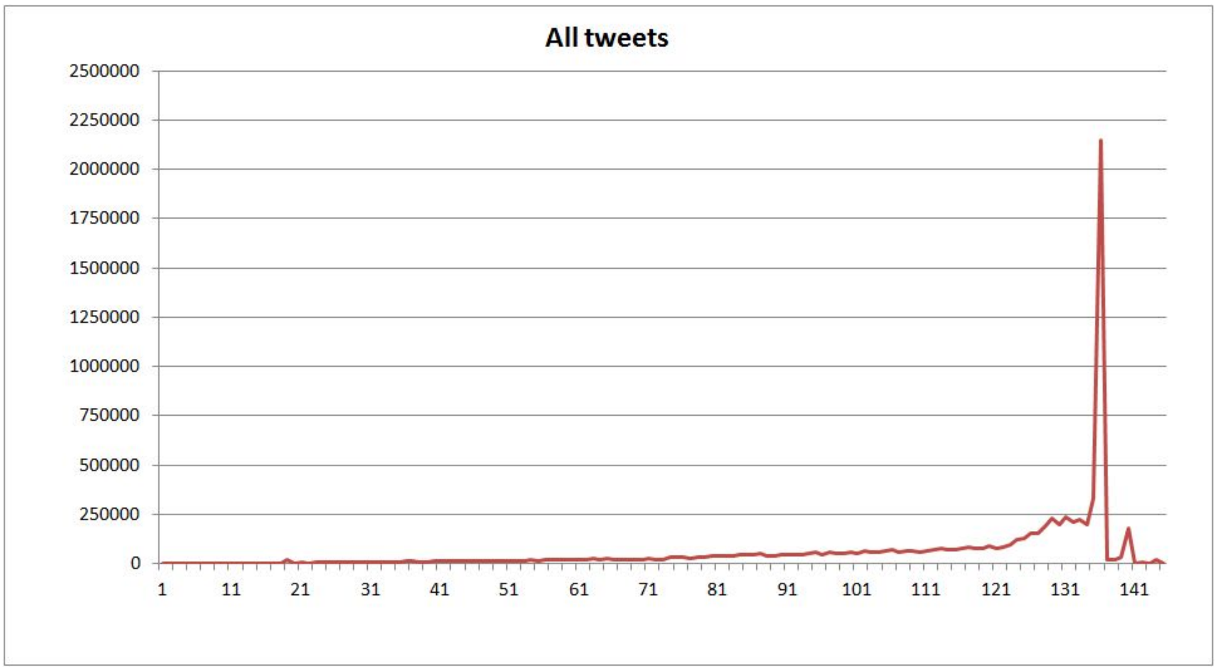
It is worth to notice that "Apache Drill cannot insert, update, or delete data that currently exists on HDFS". In addition, when creating a table, checksum is created and manual removal from the file (where the created table is located) prevents the table from being used. To add or remove records to a table, you need to save them to a new table with the data from the old table.

Results with queries are available below:

**Average tweet length**

```
use dfs.tmp;
alter session set `store.format`='csv';
create table dfs.tmp.licz as SELECT length(t.text) as `liczba znakow`, count(t.text) as
liczba_tweetow from dfs.`path_to_catalog_with_tweets` t group by length(t.text);
```

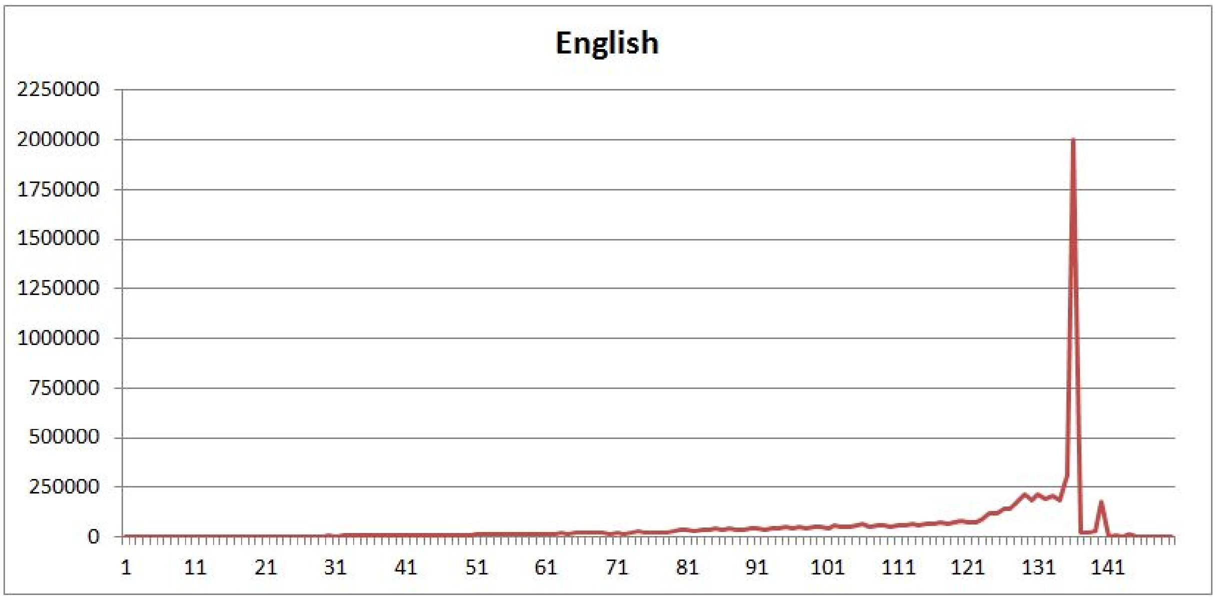
There were 8430480 tweets in dataset.



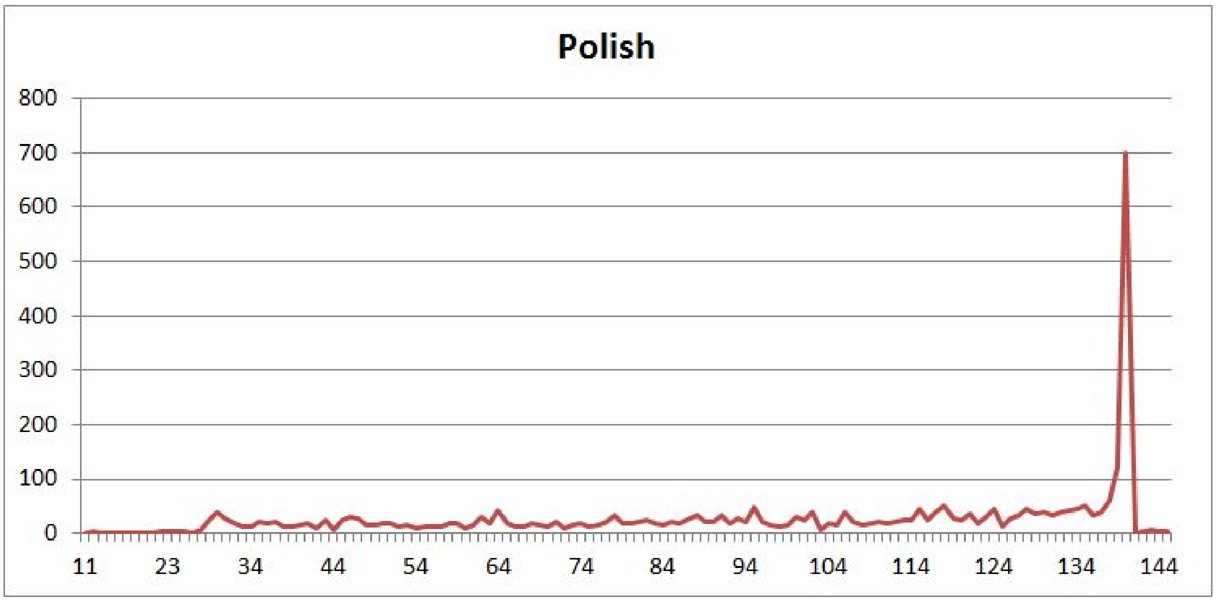
**Average tweet length grouped by language**

```
use dfs.tmp;
alter session set `store.format`='csv';
create table dfs.tmp.licz_lang as SELECT avg(length(t.text)) as `liczba znaków`, t.lang,
count(t.text) from dfs.`path_to_catalog_with_tweets` t where t.lang is not null group by
t.lang, length(t.text) order by `liczba znaków` desc;
```

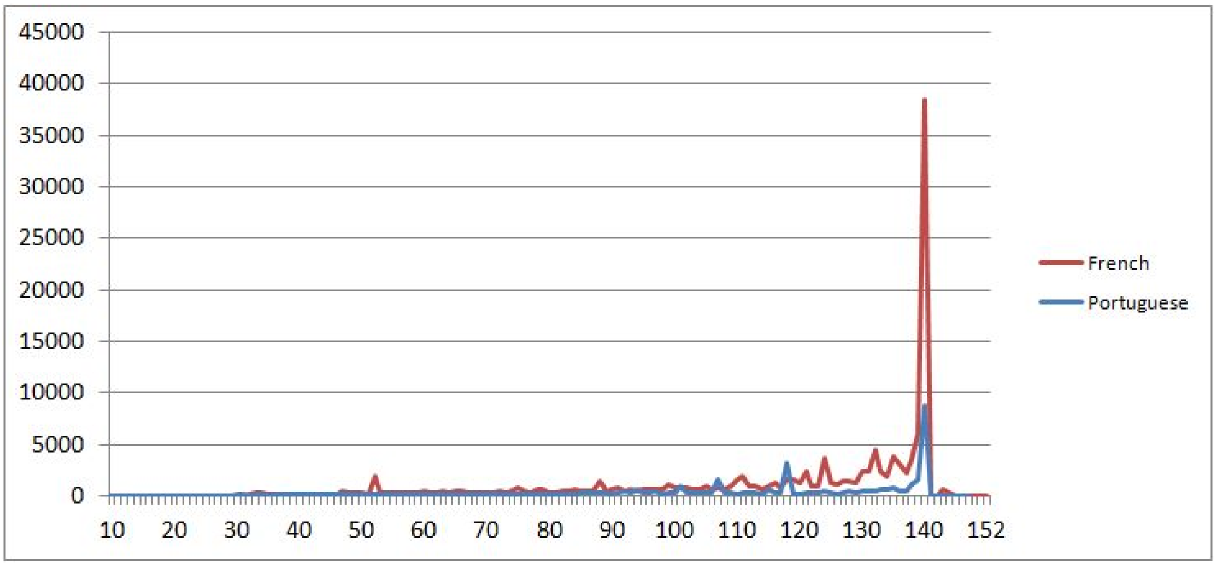
For english language there were 7639506 tweets(about 90,618% of data set).



For polish language there were only 3426 tweets.



For french language there were 139685 tweets(1,657% of data set) and for portuguese 45055 tweets(0,534% of data set).



While twitter allows up to 140 characters in tweet, there are some tweets that exceed this value due to special character encoding. The recorder had 428 characters. All graphs show that users have tried to use the maximum number of characters available. Over two million tweets (25.5% of the dataset) had exactly 140 characters. 4542672 tweets had 130-145 characters (53.9% of the data set). The shortest tweets were 5 characters (146 tweets).

## Discretized sources

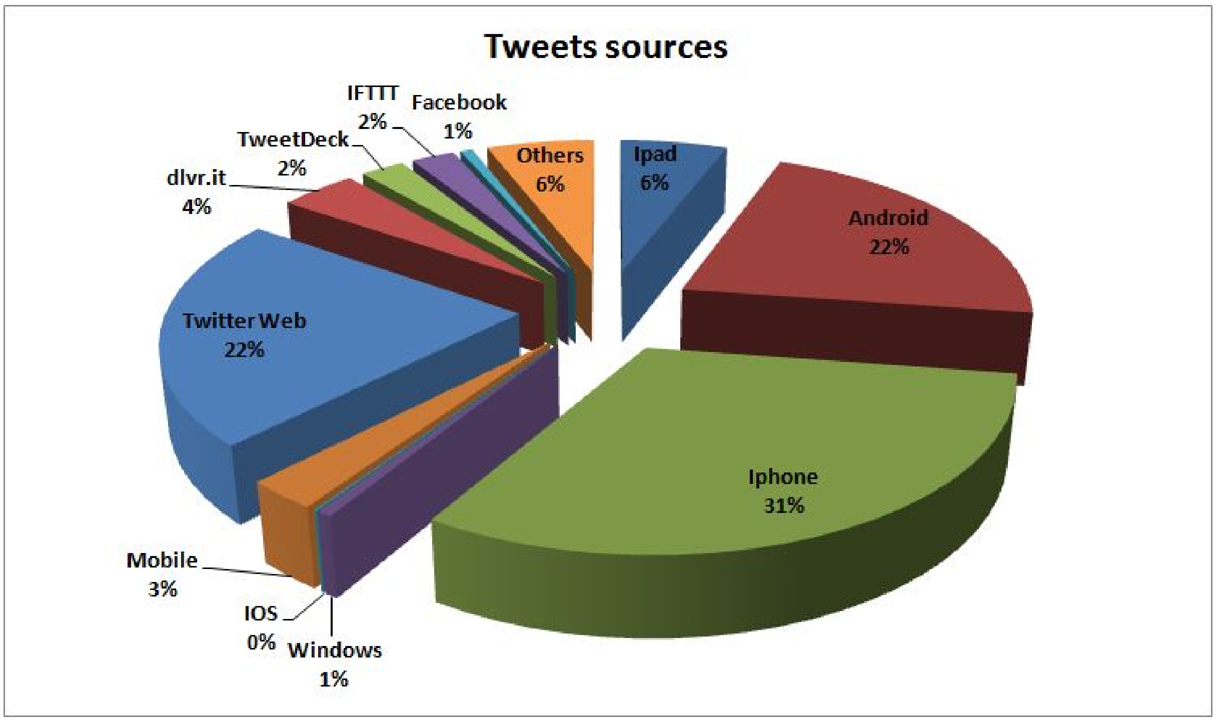
There are several thousands sources of tweets in our dataset (mainly blogs and small websites). We discretized the values of the source. First we created a group query that inserted the results to a separate table:

```
use dfs.tmp;
alter session set `store.format`='json';
create table dfs.tmp.zrodla (zrodlo, liczba) as SELECT t.source as `zrodlo`, count(*) as
`liczba` FROM dfs.`path_to_catalog_with_tweets` t group by t.source order by `liczba`
DESC;
```

Next, we performed queries like 'pattern' to extract data related to the most popular sources.For example:

```
SELECT t.zrodlo, t.liczba from dfs.tmp.zrodla t
where t.zrodlo like '%ipad%';
```

The result of these queries is the following pie chart:



## 5. Summary

Apache drill allows to perform sql queries on NoSql data without defining schema. It offers embedded mode, with can be used to quickly learn how to perform queries on this platform. It is also free and open source. But for now, there is still many unimplemented features, with could be used in more advanced queries. It is also hard to install Apache Drill in distributed mode, because Drill changes quickly and some parts of documentation look outdated.