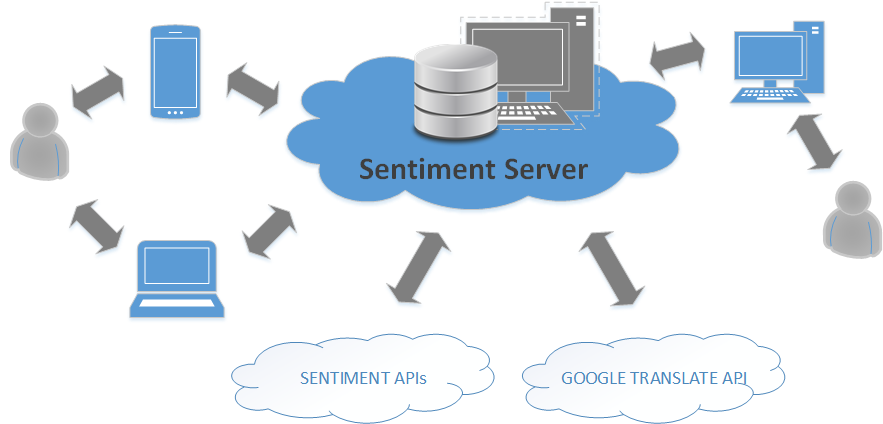
# Framework

## Design

Our system consists of web based application with responsive screen layout which can be opened using any web browser. This way, our app is not limited to any platform. High level design of system can be seen in Figure



Figure

System consists of a Web Server hosted by Digital Ocean. We have created Ubuntu 14.04.4 virtual machine with 512MB of RAM with 1 CPU, 20 GB SSD Disk Storage, Nginx 1.4.6, Gunicorn 17.5 on which we have installed MySQL server, with following characteristics:

* + **DBMS:** MySql 5.5+
  + **character set:** utf8mb4
  + **collate:** utf8mb4\_unicode\_ci
  + **database name:** sentiment\_db

In sentiment\_db we have stored provided database dump containing 100K rows, as well as added new columns to support our analysis.

Our application interacts with several external APIs for sentiment analysis. These APIs are open source sentiment analysis projects. The following table shows used sentiment analysis APIs.

| **API** | **Database column** | **Used language** |
| --- | --- | --- |
| [vivekn](http://sentiment.vivekn.com/api/text/) | sentiment\_api1 | original |
| [text-processing](http://text-processing.com/docs/sentiment.html) | sentiment\_api2 | original |
| [vivekn](http://sentiment.vivekn.com/api/text/) | sentiment\_api1\_en | english |
| [text-processing](http://text-processing.com/docs/sentiment.html) | sentiment\_api2\_en | english |
| [indico](http://text-processing.com/docs/sentiment.html) | sentiment\_api3 | english |
| [indico HQ](http://text-processing.com/docs/sentiment.html) | sentiment\_api4 | english |

Aside of the sentiment analysis APIs, we have used Google Translate API in order to check successful rate of APIs when provided data has been translated to English language.

Conceptual Design



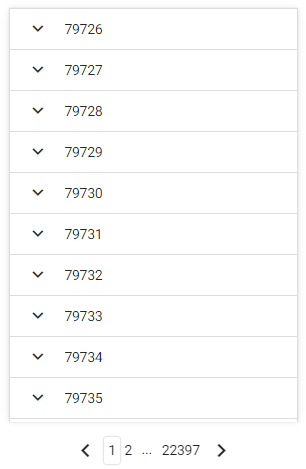
For this project we will use model-view-controller (MVC) architectural pattern (see. Figure 2). MVC pattern splits application on three logical components and defines interaction among them. This separation of responsibilities allows flexibility down the road. Model stores data that is retrieved according to commands from the controller and displayed in the view. View generates an output presentation to the user based on changes in the model. While controller sends commands to the model to update model’s state, it can also send commands to its associated view to change the view’s presentation of the model.

## Implementation

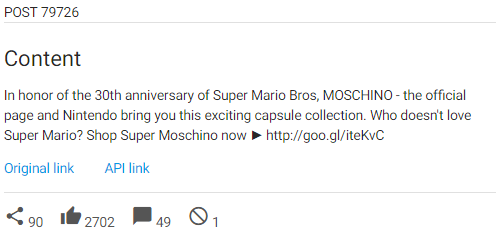
## User Interface

User interface has main landing page, which contains listed posts from the given database dump, current selected post with its details such as content of the post, link towards the original content, link toward Django API post details, statistics about number of shares, likes, comments and spam comments. On the right side of the main page pie charts are shown, representing results of investigating real sentiment of comments related to current selected post, under it pie chart is shown with statistics related to chosen sentiment API to make the comparison more visible for the user. Under the post details fragment list of comments related to the post is shown. Each comment fragment has its content shown, as well as results of sentiment analysis i.e sentiment API results versus real sentiment investigation results, spam detection status. Comment detail fragment has a link attached to it redirecting the user to Django API’s comment details page which shows retrieved data from the server in form of JSON string.

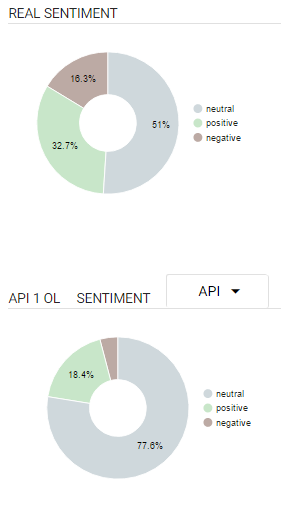
In the main page user is able to choose a post from the list of posts, change sentiment API in order to compare results of analysis of different sentiment APIs. Look of separate fragments of main page are shown in following figures.



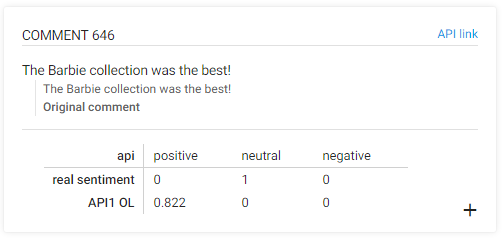
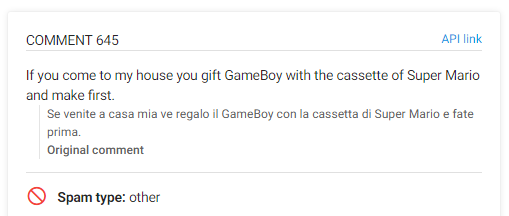
List of posts



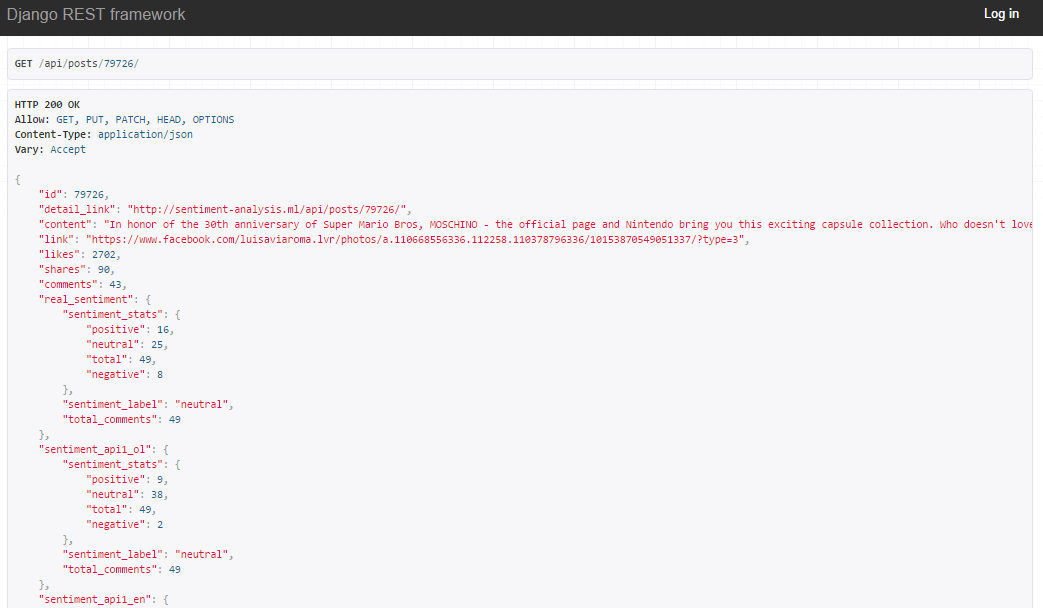
Post details fragment



Real sentiment vs api sentiment fragment



Comment details fragments



Django API JSON result representing post details page

Technical specification

Live site [sentiment-analysis.ml](http://sentiment-analysis.ml/) is installed on DigitalOcean VS with the following specs:

* 512MB of RAM with 1 CPU
* 20 GB SSD Disk Storage
* Ubuntu 14.04.4
* Nginx 1.4.6
* Gunicorn 17.5
* MySQL + 5.5.47

**Resources software:**

* [Django 1.9](https://www.djangoproject.com/)
* [Django jsonifeld 0.9.19](https://github.com/bradjasper/django-jsonfield)
* [Python lib Requests 2.9.1](http://docs.python-requests.org/en/master/)
* [Django REST framework 3.3.3](http://www.django-rest-framework.org/)

All resources are listed in requirements.txt. To install them all just use the pip install -r requirements.txt command.

**Database:**

* **DBMS:** MySql 5.5+
* **character set:** utf8mb4
* **collate:** utf8mb4\_unicode\_ci
* **database name:** sentiment\_db

**Resolved encoding issues:**

* Make sure to follow [this tutorial](https://mathiasbynens.be/notes/mysql-utf8mb4) to set encoding to utf8mb4 properly
* The server side phpMyAdmin issues were solved by harcoding encoding as seen [here](https://github.com/phpmyadmin/phpmyadmin/commit/fb30c148f22f3e1534cb216a5994660b09c7079)

**Table structure:**

The following table shows *relevant* details of database tables significant for the sentiment analysis. The full extend of the databases tables, including but not limited to: django authentication, followers etc, is not shown for the sake of simplicity.

| **Table name** | **Table columns** |
| --- | --- |
| im\_post | **id**, content, link, likes, shares, comments |
| im\_commento | **id**, idpost, content |
| im\_post\_sentiment | **id**, idpost, real\_sentiment, sentiment\_api<nb> |
| im\_commento\_sentiment | **id**, idcommento, language, english\_translation, spam, real\_sentiment, sentiment\_api<nb> |

Out of the table columns listed above, the following are in **json** format but stored as lontext:

* spam
* real\_sentiment
* sentiment\_api<nb>

**Resources APIs:**

The table maps used API's links to the column name in our database where the results of the API calls are stored

| **API** | **Database column** | **Used language** |
| --- | --- | --- |
| [vivekn](http://sentiment.vivekn.com/api/text/) | sentiment\_api1 | original |
| [text-processing](http://text-processing.com/docs/sentiment.html) | sentiment\_api2 | original |
| [vivekn](http://sentiment.vivekn.com/api/text/) | sentiment\_api1\_en | english |
| [text-processing](http://text-processing.com/docs/sentiment.html) | sentiment\_api2\_en | english |
| [indico](http://text-processing.com/docs/sentiment.html) | sentiment\_api3 | english |
| [indico HQ](http://text-processing.com/docs/sentiment.html) | sentiment\_api4 | english |

External API descriptions

**APIs**

[**Vivekn**](http://sentiment.vivekn.com/docs/api/)

**Database column:** sentiment\_api1 & sentiment\_api1\_en

You will receive a JSON response of the form:

{

"result": {

"sentiment": "Positive",

"confidence" : 73.422451

}

}

[**Text-processing**](http://text-processing.com/docs/sentiment.html)

**Database column:** sentiment\_api2 & sentiment\_api2\_en

**Label:** will be either pos if the text is determined to be positive, neg if the text is negative, or neutral if the text is neither pos nor neg.

**Probability**: an object that contains the probability for each label. neg and pos will add up to 1, while neutral is standalone. If neutral is greater than 0.5 then the label will be neutral. Otherwise, the label will be pos or neg, whichever has the greater probability.

{

"probability": {

"neg": 0.68846305481785608,

"neutral": 0.38637609994709854,

"pos": 0.31153694518214375

},

"label": "neg"

}

[**Indico**](https://indico.io/docs#sentiment_hq)

**Database column:** sentiment\_api3 & sentiment\_api4 (hq)

**Output:** This function will return a number between 0 and 1. This number is a probability representing the likelihood that the analyzed text is positive or negative. Values greater than 0.5 indicate positive sentiment, while values less than 0.5 indicate negative sentiment.

{

"results": 0.3468102081511113

}

External API issues & solutions

**General issues:**

1. a lot of comments are in italian rendering them misclassified as neutral
2. some comments (~17 of them) have no content i.e. content=''
3. hash tags detected as english but not translated

**Proposed solutions:**

1. language issue
   * classify by hand the language of the comment
   * classify automatically the language of the comment (find a lib/api)
     +   didn't predict well
     +  try google's [Translate API free trial](https://cloud.google.com/) and use the english translation of the comments to determine the sentiment
   * find & use sentiment api that supports languages other than english
     + again, try [indico's](https://indico.io/docs" \l "sentiment) sentiment and sentiment HQ apis
2. comments without content issue:
   * ignore
   * do a bit of data cleaning (i.e. delete them)

**API specific issues:**

**TextProcessingAPI:**

* cannot classify comments with no content for content='' getting response 400

User manual: the developed API

Other than being able to access the API via a browser, it can also be done from the command-line, using tools like curl. For e.g.

* To get a list of posts:

curl -u admin:sentiment1234 http://sentiment-analysis.ml/api/posts/ -H 'Accept: application/json; indent=2'

* To get single post detail:

curl -u admin:sentiment1234 http://sentiment-analysis.ml/api/posts/<id>/ -H 'Accept: application/json; indent=2'

* To get a list of comments:

curl -u admin:sentiment1234 http://sentiment-analysis.ml/api/comments/ -H 'Accept: application/json; indent=2'

* To get single comment detail:

curl -u admin:sentiment1234 http://sentiment-analysis.ml/api/comments/<id>/ -H 'Accept: a

User manual: scripts to analyze sentiment with external APIs

**Description**

Scripts in the scripts/python/ folder are used to determine a comment's sentiment by making a request to an external API and then store the result it in the **sentiment\_db** database. This can be done for a single comment providing its **id**, sequentially for all comments, and for comments whose **id** satisfies id > nb1 AND id <nb2.

**Note:** the predict\_sentiment\_of\_translated\_comments.py script analyses an English translation of the comment's content. The comments have been translated in English by [Google's Translate API](https://cloud.google.com/translate/v2/using_rest). If you by chance want to analyse comment's original content just use run\_batch\_api.py instead.

**Usage**

The scripts are to be invoked from the command line by specifying parameters

python predict\_sentiment\_of\_translated\_comments.py <params>

To find out which params the script accepts run the following command:

python predict\_sentiment\_of\_translated\_comments.py --help

Even though accepted parameters and their descriptions can be listed using the command above, but for the sake of completeness, here is the list:

* **-api <name>** specifies which api to use (*Required*)
* **-ideq <nb>** specifies the id of the comment we want to analyze (*Optional*)
* **-idgt <nb>** analyzes all comments that satisfy id > nb (*Optional*)
* **-idlt <nb>** analyzes all comments that satisfy id < nb (*Optional*)

***Note***: Params **-idgt** and **-idlt** can be used on their own or in combination, but they will be ignored should the **-ideq** param be specified

**Examples**

Make sure you're in the scripts/python/ folder or modify the commands accordingly.

To analyze and store sentiment of **all** comments run:

python predict\_sentiment\_of\_translated\_comments.py -api TextProcessingApi

# possible choices = {IndicoAPI, IndicoHqAPI, ViveknAPI,TextProcessingAPI}

To analyze and store sentiment of a **single** comment with id=5 run:

python predict\_sentiment\_of\_translated\_comments.py -api TextProcessingApi -ideq 5

# Expected response:

Using Text-processing API: http://text-processing.com/api/sentiment/

--------------------------------------------------

Comment\_id: 5

Content: Nice

Predicted sentiment: positive

-----------------------------

To analyze and store sentiment of all comments that satisfy id<5 run:

python predict\_sentiment\_of\_translated\_comments.py -api TextProcessingApi -idlt 5

To analyze and store sentiment of all comments that satisfy id>3 AND id<5 run:

python predict\_sentiment\_of\_translated\_comments.py -api TextProcessingApi -idgt 3 -idlt 5