Sentiment analysis on tweets using ClowdFlows platform

Olha Shepelenko
Computer Science
University of Tartu
Tartu, Estonia
Email: olha89@ut.ee

***Abstract*—In the research paper I am using social media (Twitter) as a source of opinions about pre-defined topics. Then, I apply sentiment analysis, semantic search and mining approaches in order to extract opinions about the pre-defined topics and classify them for the purpose of insights discovery. Moreover, all the methods adopted in this investigation is integrated into ClowdFlows platform that allows to construct and execute workflows in real-time.**

I. Introduction

Nowadays, mankind is living in an era where information and communication technologies has evolved so rapidly. In addition, social media plays a big role in our daily life; because, it is a place where people share their thoughts, experience, and opinion. This type of information is a powerful tool in the modern world and it is everywhere; however, it should be properly collected and analyzed in order to make sense of it. One possible form of information is opinion (sentiment). It could be used in different ways, for different purposes. For instance, it was always important for people to have additional opinion about the product or service in order to make a final decision whether to buy this product/service or not. However, today we are not limited to ask friends or relatives opinion any more, we can easily explore peoples opinions towards things we are interested in through the Web. Thanks to high availability of social media resources such as social networks, blogs, forums, and online reviews, we have an opportunity to collect raw data for opinion mining.

Nevertheless, searching opinion sources and monitoring them in the Internet can be difficult; since every source can have large volume of opinions. Frequently, sentiments are masked in long posts (discussions) that makes detection of opinions hard. Hence, we can see necessity of applying sentiment analysis, also known as opinion mining. Research in sentiment analysis has a significant impact on Natural Language Processing, but it also influences the other areas that are affected by peoples opinion (social, political, management sciences, economics).

II. Related work

This section presents an overview of ClowdFlows platform and its features.

A. Architecture of ClowdFlows platform and technologies used

ClowdFlows platform consists of two parts. Execution of the workflows happens on the server side and the workflows construction is made via web application through the browser. The architecture of the ClowdFlows system accessed by multiple users is depicted in Fig. 1 [1].

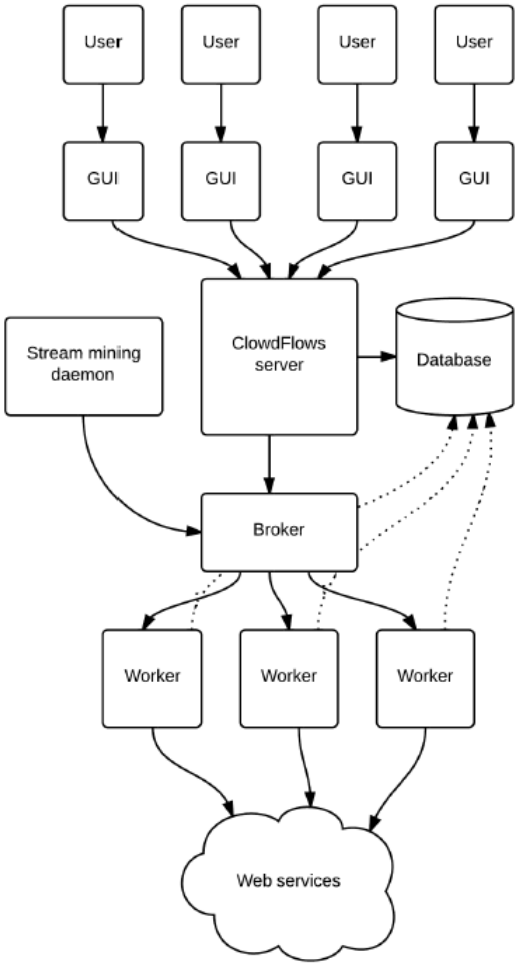

Fig.1 - The architecture of ClowdFlows platform

Python using the Django framework was applied for implementing the server side. HTML and JavaScript were

used for developing the client side (GUI). ClowdFlows can work with various databases. It is possible to install ClowdFlows on multiple machines, this is enabled by applying the RabbitMQ messaging server and broker and Django implementation of Celery for transmitting asynchronous tasks between servers. PySimpleSoap library is utilized for enabling consumption of Web services.

Moreover, authors describe in details the distinctive features of ClowdFlows platform, such as a web-service consumption module, a real-time processing module and possibility of workflows sharing publicly. Modular design makes usage of the platform much easier and makes possible to extend packages.

B. The graphical user interface

Workflows are constructed via GUI that follows the visual programming paradigm. This approach makes easier the representation of complicated processes into a spartial arrangement of building units (widgets). Workflow is constructed using widgets that are connected with each other. GUI has the widget repository and canvas. Adding widgets to the canvas happens by drag and drop.

Widgets can be four types: regular, visualization, interactive, workflow control widgets (i.e. a widget that contains a workflow [1]).

Canvas allows to add, connect and delete widgets/connections as well as issue commands in order to execute widgets/workflows. Asynchronous HTTP POST request is forwarded to the server when the user does an operation. The operation is validated on the server and a success or error message with additional information is passed to the user interface (the clients browser) formatted in JSON or HTML [2].

Above the canvas you can find tollbar that allows to to save, copy, and delete workflows. Notice that workflow can be shared with other users, it this case your workflow gets specific URL.

C. The Workflow Execution Engine

All executable widgets have to be executed by execution engine, this should be done in the correct order. Hence, the main purpose of the execution engine is the absence of widgets in executable or running state. Widgets are processed by engine that assigns various execution states to them. There are 5 types of the states: executable, pending, running, finished, or failed.

Authors implemented engine twice. First implementation is done in Python, second in JavaScript because of performance problems when a user wants to check the order of the executed widgets in real time. Engine that is implemented in JavaScript allows check the results of the execution of widget in real time, however Python implementation allows to check results when execution is totally completed.

In case when a workflow is running the engine constantly monitors if executable widgets are available and if it finds them, it executes such widgets. If widgets are independent (input of one widget does not depend on the output of another widget) and they have executable state simultaneously, they can be asynchronously executed in parallel. On the other hand, widget state mechanism will not allow simultaneous execution of dependent widgets.

D. Real-time data analysis

ClowdFlows system is able to do processing real-time data streams. When you are trying to execute data stream workflow you have to keep in mind that it is executed potentially infinite number of times, execution will last until the user stops it. For handling data streams authors changed engine to execute the workflow numerous times at arbitrarily small temporal intervals in parallel.

Stream mining daemon delegates the execution workflows. Tasks are issued to messaging queue by stream mining daemon, which can assign priority to execution. Workers take tasks from the queue and execute workflow.

Widgets for real-time processing have the internal memory (for storing information, i.e. the timestamp of the last processed instance) and halting mechanism.

Authors also developed several additional components:

- aggregation widget
- sliding window widget
- sampling widgets
- stream visualization widgets.

III. Data

As a dataset I am using sentence polarity dataset v1.0 introduced in Pang/Lee ACL 2005 [3]. Dataset represents movie reviews. It contains 5331 positive and 5331 negative sentences. The training set has 7996 reviews and testing 2666 reviews.

Movie reviews was chosen as a dataset because reviews comprise a wide range of human sentiments and capture a lot of adjectives appropriate for sentiment classification.

Training set is used for building supervised learning model based on Naive Bayes Classifier. Testing set is used for evaluating the accuracy of the given classifier.

IV. Algorithm for classification of sentiments

Sentiment analysis, also called opinion mining, is the field of study that analyzes peoples opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes [4].

Sentiment analysis and opinion mining mainly focuses on opinions which express or imply positive or negative sentiments.

Several approaches for sentiment analysis are known [5]:

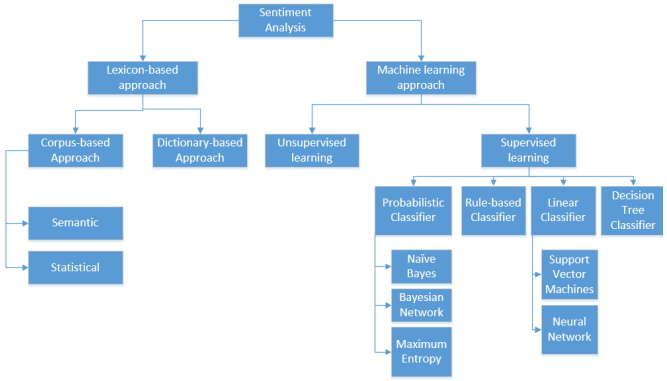

Fig.2 - Sentiment analysis approaches

In this research paper I am using a machine learning method and applying the Naive Bayes Classifier.

A. Naive Bayes Classifier

Naive Bayes Classifier is a simple probabilistic model based on the Bayes theorem:

$$P(label|features) = \frac{P(label) * P(features|label)}{P(features)} \quad (1)$$

where *P(label | features)* – posterior probability of a particular features belong to a label,
P(label) – prior probability of a features occur in label,
P (features| label) – conditional probability of encountering a feature among features belonging to a label,
P (features) – prior probability of feature in corpus features.

In this research paper, we assume that there are only two classes of label: positive or negative. Lets make naive assumption that is given a label, the features are independent of each other. The assumption does not influence the accuracy of the text a lot but speeds up classification algorithm applicable for the problem.

$$P(features|label) \approx P(f_1|label) * P(f_2|label) * \ldots \\ \ldots * P(f_n|label) = \prod_{i=1}^{n} P(f_i|label) \quad (2)$$

$$P(label|features) = \frac{P(label) * \prod_{i=1}^{n} P(f_i|label)}{P(features)} \quad (3)$$

where f_i – individual features.
The purpose of classification is to define to what label the feature belongs, so we do not need the probability, but we want to define the most probable label. Bayesian classifier uses the maximum a posteriori estimation to determine the most probable label $label_{map}$:

$$label_{map} = argmax_{label \in L} \left[\frac{P(label) * \prod_{i=1}^{n} P(f_i|label)}{P(features)} \right] \quad (4)$$

Denominator can be ignored because it will be the same for positive and negative labels. So we can rewrite expression for $label_{map}$:

$$label_{map} = argmax_{label \in L} \left[P(label) * \prod_{i=1}^{n} P(f_i|label) \right] \quad (5)$$

In the above equation, we multiply many conditional probabilities. In order to avoid floating point underflow we will apply the logarithm product property: *log(xy)= log x+ log y*, so instead of multiplying probabilities we are adding logarithms of probabilities. Due to logarithm function is monotonic, applying it to both parts of the expression will only change its numeric value, however not the parameters in which maximum is reached. Hence the label that has the highest log probability score is still the most likely:

$$label_{map} = argmax_{label \in L} \left[P(label) + \sum_{i=1}^{n} P(f_i|label) \right] \quad (6)$$

Estimation of *P(label)* and *P(f_i | label)* is performed on the training data. The prior probability can be estimated as:

$$P(label) = \frac{N_{label}}{N} \quad (7)$$

where N_{label} – number of features that belongs to the respective label,
N – total number of features in the training data (positive and negative).

Conditional probability *P(f_i| label)* can be estimated in several way, we are using multinomial Bayes model:

$$P(f_i|label) = \frac{F_{ilabel}}{\sum_{i' \in V} F_{i'label}} \quad (8)$$

where $F_{i\ label}$ – number of occurrences of *i*-th feature in the training data with respective label, including repetitions of the features,
V – dictionary that contains all unique features for particular label.

Applying the above formula we can encounter with a problem. It can happen that on the classification stage the feature has occurred, however the same feature has never occurred in training data, in this case None is assigned as a frequency of this feature $F_{i\ label}$, hence *P(f_i | label)* is also None.

Another solution for this problem is Laplace smoothing. The idea is to add one to the frequency of each feature:

$$P(f_i|label) = \frac{F_{ilabel} + 1}{\sum_{i' \in V}(F_{i'label} + 1)} = \frac{F_{ilabel} + 1}{|V| + \sum_{i' \in V} F_{i'label}} \quad (9)$$

Using above formulas we can obtain final formula for Bayesian classification:

$$label_{map} = argmax_{label \in L} \left[\log \frac{N_{label}}{N} + \right. \\ \left. + \sum_{i=1}^{n} \log \frac{F_{ilabel}}{\sum_{i' \in V} F_{i'label}} \right] \quad (10)$$

Or the following formula if we apply Laplace smoothing:

$$label_{map} = argmax_{label \in L} \left[\log \frac{N_{label}}{N} + \right. \\ \left. + \log \frac{F_{ilabel} + 1}{|V| + \sum_{i' \in V} F_{i'label}} \right] \quad (11)$$

All necessary information for training and applying the Naive Bayes classifier is provided above. The algorithm is represented in Figure 2 (adapted from [6]).

Algorithm 1 MUTLINOMIALNB

procedure TRAINMULTINOMIALNB(L, F)
  $V \leftarrow ExtractVocabulary$(F)
  $N \leftarrow CountFeatures$(F)
  for each $label \in L$ **do**
    $N_{label} \leftarrow CountFeatures$(F)
    $prior[c] \leftarrow N_{label}/N$
    $text_{label} \leftarrow ConcatTextOfAllFeatInLabel$(F,label)
    for each $f \in V$ **do**
      $F_{ilabel} \leftarrow CountFrequencyOfFeat(text_{label}, f)$
      for each $f \in V$ **do**
        $condprob[f][label] \leftarrow \frac{F_{ilabel}}{\sum_{i' \in V} F_{i'label}}$
      end for
    end for
  end for
  return $V, prior, condprob$
end procedure

procedure APPLYMULTINOMIALNB(L, V, $prior$, $condprob$)
  $W \leftarrow ExtractFeatures$(V)
  for each $label \in L$ **do**
    $score_{label} \leftarrow logprior[label]$
    for each $f \in W$ **do**
      $score_{label} += logcondprob[f][label]$
    end for
  end for
  return $argmax_{label \in L} score[label]$
end procedure

Fig. 3 - Naive Bayes algorithm (multinomial model): training and testing

V. TRAINING AND TESTING SYSTEM

In this section I will explain how system works when we are training and testing data.

As a training data we have two separate files: one file contains reviews with a positive sentiment, second with a negative. Next step is building frequency distribution of all words, in other words algorithm has to find occurrences of all the words as well as occurrences within each label (positive, negative). Then algorithm counts number of positive and negative words, and also total number of the words. After completion this stage we have dictionary that contains word score for each word in the document. The following step is filtering words based on score, so we want to use only the most informative word for training the model. All these retrieved parameters are called classification model, having the model allows us to apply the Naive Bayes Classifier for training data.

After training is done we want to check how system is accurate. To do this we have to use our test data: classify test features and evaluate metrics of the system, such as accuracy, prediction and recall. On the figure 4 you can find system for training and testing looks like.

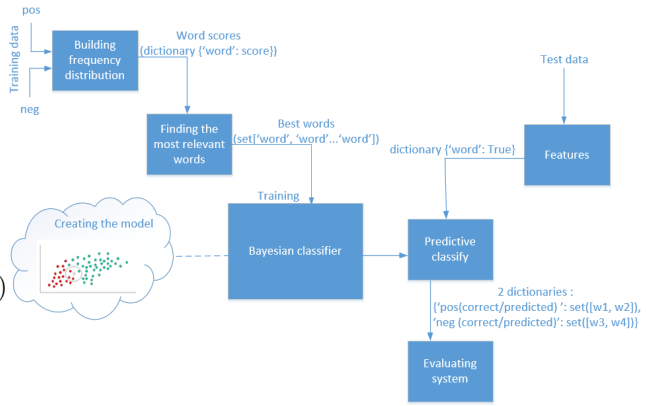

Fig.4 - Training and testing system

VI. INTEGRATION WITH CLOWDFLOWS PLATFORM

In this section I will explain how to construct a workflow and how to integrate sentiment analysis technique into Clowdflows platform.

A. Design of the workflow

Lets define what kind of widgets we need to use for designing our workflow.

First of all we need a stream input that will collect tweets from the Twitter based on specific query. ClowdFlows widget repository contains widget called Twitter, it connects to the Twitter through Twitter API. You have to set several parameters to execute this widget, namely, specify the query based on which you want to select ingoing tweets and also you need to enter the credentials for the Twitter API (consumer key, consumer secret, access token, access token secret), optionally you can provide geocode. The widget can operate in streaming and non-streaming context. Twitter widget extracts the latest results of the query when it is executed. In case we are working with streaming workflows, widgets internal memory keeps the ID of the latest tweet, ID is sent to the Twitter API, this procedure ensures extracting only unique tweets.

Second of all we need sampling widget that will filter tweets based on language, so it will reject all non-English tweets. The heart of the system is the widget that will carry out sentiment analysis, in this research work Naive Bayes Classifier is used. For training the algorithm a labeled dataset was used, algorithm is implemented in Python. For using classifier in the ClowdFlows platform it is exposed as a Python function.

After completion sentiment analysis we can split our steam of tweets into positive and negative tweets. In order to catch particular amount of tweets we are using sliding window with size (widget parameter) equals to the number of tweet we want to get. We need to do this because visualization widgets (Positive/Negative words, Positive/Negative Word Cloud) merely show the last data obtained as an input parameters for these widgets, so for instance specifying size to 500, we will get 500 of the last tweets.

Another visualization window is Sentiment graph. It is line chart that depicts four different dependencies over time: total volume, volume of positive tweets, volume of negative tweets and difference between positive and negative tweets.

To conclude, using ClowdFlows GUI we have designed workflow for sentiment analysis on tweets. New widget for sentiment analysis was created and added to the streaming repository. Widgets were chosen from the repository and added to the canvas by drag and drop, then connection between corresponding widget was added. Constructed workflow is represented on figure 5.

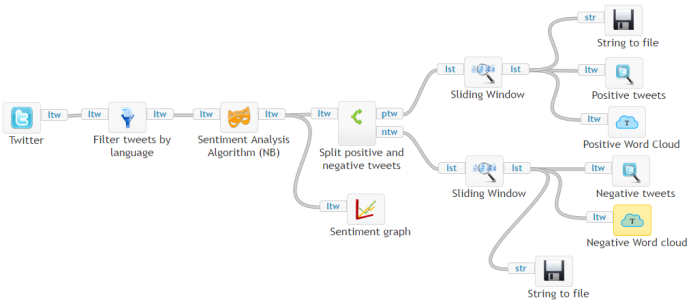

Fig. 5 Twitter sentiment analysis workflow

B. Integration with Clowdflows platform

To integrate an algorithm for example sentiment analysis into the clowdflows platform I can apply two methods:

- implement widget as a Python function and include it in a ClowdFlows platform
- manually import widget through GUI as a WSDL Web service.

Despite the fact that it is easier to use Web service import, I am using more stable method – implementing widget [7].

It is impormant to notice that, creating a widget is allowed when ClowdFlows platform is deployed locally. It allows to generate sceleton code for widget (Python function).

To create a widget you have to go to the page that allow to add new widget: http://127.0.0.1:8000/admin/workflows/abstractwidget/add/ [8], where widgets attributes have to be specified:

- name of the widget
- action (name of the Python function)
- description of the widget
- category to which widget belongs (in my case it is streaming)
- visualization view (Python function that is a view that will render a template)
- there is also more optional fields.

Once all necessary attributes is entered, you have to add the inputs and outputs. Here you have to provide the name of the input/output, description, variable (provides how the data will be accessed in the python function that is executed when the widget runs), type. You also can add flags to the inputs: required, parameter, multi (whenever a connection is added to this input another input will be created on the fly that accepts the same data).

Eventually, you have to open library.py file locally and add your function, that takes a dictionary as an input and returns a dictionary as an output.

VII. Results

For training Naive Bayes Classifier it was used 7996 instances, for testing 2666. Testing set contains 1333 positive and 1333 negative words.

TABLE I
Estimation results

	positive	negative	total
positive	1268	65	1333
negative	0	1333	1333

Given confusion matrix shows that negative words were recognized correctly in all cases (100%), but positive words were detected correctly in 95% cases.

Below you can find 10 most informative words:

engrossing = True	pos : neg	=	17.0 : 1.0
quiet = True	pos : neg	=	15.7 : 1.0
mediocre = True	neg : pos	=	13.7 : 1.0
absorbing = True	pos : neg	=	13.0 : 1.0
portrait = True	pos : neg	=	12.4 : 1.0
refreshing = True	pos : neg	=	12.3 : 1.0
inventive = True	pos : neg	=	12.3 : 1.0
flaws = True	pos : neg	=	12.3 : 1.0
triumph = True	pos : neg	=	11.7 : 1.0
refreshingly = True	pos : neg	=	11.7 : 1.0

For example, there is 17 to 1 chance to recognize ”engrossing” as positive word.

VIII. Conclusion and further work

This paper presents sentiment analysis technique that uses Naive Bayes Classifier and integration approach of this algorithm into Clowdflows platform that is developed for construction and execution workflows, including real-time data.

I have made short overview of ClowdFlows platform, presented algorithm for training and testing data for sentiment analysis and explained workflow construction and integration with ClowdFlows. The results of work delivered in Section VII.

As future perspective, I will conduct more experiments with different datasets, depending on evaluation system metrics

I will decide about necessity of further sentiment analysis algorithm improvement.

REFERENCES

[1] Janez Kranjc, Vid Podpecan, Nada Lavrac, *Real-Time Data Analysis in ClowdFlows*. IEEE International Conference on Big Data, 2013.

[2] Janez Kranjc, Jasmina Smailovic, Vid Podpecan, Miha Grcar, Martin Znidaric, Nada Lavrac, *Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the ClowdFlows platform*. Information Processing and Management 51, 2015.

[3] Bo Pang and Lillian Lee, *Movie Review Data. Sentence polarity dataset v1.0* . url: https://www.cs.cornell.edu/people/pabo/movie-review-data/, 2005.

[4] Bing Liu, *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers, 2012

[5] Walaa Medhat, Ahmed Hassan, Hoda Korashy, *Sentiment analysis algorithms and applications: A survey*. Ain Shams Engineering Journal (2014) 5, 10931113, 2014.

[6] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schtze, *An Introduction to Information Retrieval*. Cambridge University Press Cambridge, England, 2009.

[7] Janez Kranjc, Roman Orac, Vid Podpecan, Nada Lavrac, Marko Robnik- Sikonja, *ClowdFlows: Online workflows for distributed big data mining*. Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.07.018

[8] Janez Kranjc *Widget and Abstract Widget*. url: https://github.com/xflows/clowdflows/wiki/widget