Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 737–744

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

Detection of Hate Speech Text in Hindi-English Code-mixed Data

Sreelakshmi k[a], Premjith B[a], Soman K.P[a]

[a]*Center for Computational Engineering & Networking (CEN)*
Amrita School of Engineering, Coimbatore,
Amrita Vishwa Vidyapeetham, India

Abstract

Social media sites like Twitter, Facebook, being user-friendly and a free source, provide opportunities to people to air their voice. People, irrespective of the age group, use these sites to share every moment of their life making these sites flooded with data. Apart from these commendable features, these sites have down side as well. Due to lack of restrictions set by these sites for its users to express their views as they like, anybody can make adverse and unrealistic comments in abusive language against anybody with an ulterior motive to tarnish one's image and status in the society. So it became a huge responsibility for the Government and these sites to identify this hate content before it disseminates to mass. Automatic hate speech detection faces quite a lot of challenges due to the non-standard variations in spelling and grammar. Especially for a country like India with huge multilingual and bilingual population, this hate content would be in code-mixed form which makes the task demanding. So our paper projects a machine learning model to detect hate speech in Hindi-English code-mixed social media text. The methodology makes use of Facebook's pre-trained word embedding library, fastText to represent 10000 data samples collected from different sources as hate and non-hate. The performance of the proposed methodology is compared with word2vec and doc2vec features and it is observed that fastText features gave better feature representation with Support Vector Machine (SVM)-Radial Basis Funcrion (RBF) classifier. The paper also provides an insight to the researchers working in the field of code-mixed data that character level features provide best result for code-mixed data.

Keywords: Code-mixed, Support Vector Machine, fastText, Hate speech, Natural Language Processing

1. Introduction

Over the last ten years along with the growth of population, the number of social media users have also increased extensively. Social media sites like Twitter and Facebook open opportunities for people to express and share their thoughts globally which led to the flooding of text data in these sites. While these social media sites provide platforms

* Corresponding author. Tel.: +91-8075968467
E-mail address: ammaslakshmy@gmail.com

10.1016/j.procs.2020.04.080

for people to interact with each other and share news, they also have a dark side. The tremendous usage of these platforms has led to the promulgation of hateful and offensive content resulting in cyber violence.

Hate speech is a form of verbal or non-verbal communication expressing prejudice and aggression. It is defined as an act of belittling a person or community based on their gender, age, sexual orientation, race, religion, nationality, ethnicity etc., [1], [2]. These hate speech contents lead to lot of violence and the social media companies like Twitter, Facebook etc., face lot of criticism for the increasing offensive content in their sites [3]. Not only human being, the hate content can corrupt the chatbots as well. Microscoft's chatbot 'Tay' which was developed to engage people through casual and playful conversation started using filthy language, which it learned from the conversation with people. The chatbot was unable to understand and avoid the hate content. So the detection of hate speech in tweets and social media sites has important applications in Chatbot building, content recommendation, sentiment analysis etc.

India being a diverse country in terms of its culture and language has a huge population using code-mixed language in social media. Around 44% of the Indian population speak Hindi. So the usage of Hindi-English code-mixed language is very high in Twitter and Facebook. It is mainly seen among bilingual and multilingual communities. Code-mixing is the usage of certain words, phrases or morphemes of one language in other language [4], [18]. Two such hate and non-hate texts with their translations are given below

Text-1: Hate:
Look ye politicians suvar jaise baithe rahte hain sirf money ke liye kaam karte hain. They don't care about public
Translation:
Look these politicians are sitting like pigs and they only work for money. They don't care about public.
Text-2: Non-hate:
tumhara SC,ST logon ko koi fark nahi padega. Iss reservation ki wajah se so many OC people losing their chances.
Translation:
You SC,ST people don't feel any difference. Many OC people losing their chances because of this reservations.

Both the texts use few English words like 'look', 'politicians', 'reservation' along with Hindi words and also the Hindi words are written using roman script. Text-1 contains hate-full meaning and text-2 non-hate-full. The linguistic complexity as well as the non-standard variations in grammar, spelling and translation of language makes the detection of hate and offensive content in social media a difficult task. Lot of works have been done in hate speech detection from tweets in English. But with the rise in the usage of regional languages in social media the need for researchers to find hate speech from tweets has increased.[5].

This paper contributes a machine learning model which does a binary classification of Hindi-English tweets to hate and non-hate. The model outperformed the existing works in performance. In this work pre-trained models such as fastText and bilingual embedding were used as features for the classification. FastText is Facebook's pre-trained model which is a one-hidden layer neural network. The two main applications of fastText is text representation and text classification. It considers character n-grams as the smallest unit for vector representation which makes it useful to get word vectors for out of vocabulary words in comparison to word2vec[6]. Satyajit et.al [7] has developed Bilingual word embeddings by training a word2vec model for 255,309 Hindi-English tweets. These trained word embeddings were used as text representation for our tweets. Grabbing the semantics of the whole sentence can contribute more to the identification of hate content which led to the use of Gensim's [19] Doc2vec feature. Doc2vec gives vector representations for a sentence or a document irrespective of its length. Different machine learning algorithms such as Support vector machine, [16] Random forest [17] were used as classifiers [15].

The paper proceeds as follows. Section 2 provides a brief description about the existing works done in the area of hate speech detection. Section 3 provides details about the dataset used for the experiments. Section 4 accords a description of the proposed methodology. Section 5 provides the details of the experiments and results and the paper is concluded in the section 6.

2. Related work

Several works have been done for the detection of hate, offensive and aggressive contents from tweets in English. But due to challenges in language and the lack of availability of dataset, there are very less works done in hate speech

detection from Hindi-English code-mixed tweets. The first work on hate speech detection from Hindi-English tweets was done by Aditya et.al [8]. They collected around 4575 Hindi-English tweets which were annotated by two linguists and were validated by calculating the inter annotation agreement using Cohen's Kappa coefficient. The data was then preprocessed and features like punctuation count, emoticon count, word n-gram, character n-gram, word2vec of lexicon words were extracted. These features created a fat matrix. So they used a dimensionality reduction technique such as chi square to reduce the size of the feature extracted fat matrix. SVM and Random forest was used for classification which gave accuracies of 71.7% and 66.7% respectively.

The second work on this data was done by Santosh et.al [9]. They conducted two sets of deep learning experiments namely sub-word level LSTM model and Hierarchical LSTM model with attention based on ph onemic sub-words. Sub-word level LSTM model gave an accuracy of 69.8%. Hierarchical LSTM model with attention based on phonemic sub-words gave an accuracy of 66.6%. They also experimentally compared the performance of their model with the existing work.

Puneet et.al [10] did a work on classification of offensive tweet in Hinglish. Through this paper they introduced a new set of 3000 data which are labeled as Non-offensive, Abusive and Hate-inducing. The labelling was done by three annotators and the quality of the data was evaluated. A transfer learning approach was used where the LSTM and CNN models were trained on English offensive Tweet then with this weights the model was retrained using the Hindi-English tweets. Different word embedding techniques such as Glove, fastText and Twitter word2vec are used as features.

Piotr et.al [11] experimentally proved that the fastText is equally competent with the deep learning models as well as its computational speed is many orders higher than deep learning. They evaluated the efficiency of fastText by conducting two sets of experiments namely Sentiment analysis and Tag prediction. It was observed that fastText gave better accuracy when compared to the existing baseline approaches. They also made a comparison of training time between fastText and baseline methods. The experiments proved that fastText has outperformed deep learning in terms of accuracy as well as computational time. Igor et.al [12] developed a CNN model for sentiment analysis. It made use of facebook's fastText word embedding as word representation for the task and the classification was done using deep learning. From the experimental results it was observed that CNN outperformed all the baseline approaches.

Satyajit et.al [7] developed a domain specific word embedding from 255,309 Hindi-English tweets having hate and non-hate content which were collected using Twitter API. They used gensim's word2vec algorithm to train the word embedding model. Using these embeddings as features they conducted experiments on the dataset provided in the basepaper [8]. Deep learning algorithms such as CNN-1D, LSTM and BiLSTM were used for classification. Among the three, 1D-CNN performed well with an accuracy of 82.62% followed by BiLSTM with 81.48% and LSTM with 80.21% accuracies.

Duyu et.al [14] proposed a method for sentiment classification of tweets from word embeddings. Most of the normal word representations only capture the syntactic content ignoring the sentiments. So in their paper they developed a sentiment specific word embedding which is capable of capturing sentiments from sentences. The embedding was developed on 10 million dataset collected from Twitter which were not annotated. The performance of these embeddings was tested on the SemEval 2013 data. The classification was carried out using different deep learning algorithms. The performance proved the importance of domain specific word embedding for sentiment classification.

Ankita Rane et.al [13] proposed a method for sentiment classification of Twitter data for US Airline Service Analysis. The work made use of Doc2vec algorithm for the representation of sentences as vectors. Further, different machine learning algorithms such as Decision Tree, Random Forest, SVM, Logistic Regression, Gaussian Naïve Bayes, K- Nearest Neighbour, AdaBoost are used for classification.

3. Overview of dataset

This section provides a detailed description of the dataset used for experimentation. The dataset consists of a total of 10000 texts which are equally divided to two classes namely hate and non-hate. Table 1 provides the dataset details.

The dataset is formed by collecting tweets from 3 different sources. The three sources are given below

- First set of data is taken from the baseline paper [8]. They collected the data using Twitter API and was annotated by two linguists who have knowledge in both Hindi and English.

Table 1. Detailed dataset description

Classes	Number of texts in each class
Hate	5000
Non-hate	5000

- Second set of data is grabbed from the paper 'Classification of Offensive Tweets in Hinglish Language' [10]. In this the dataset was labeled as Non-Offensive Abusive and Hate-inducing but for our current task the 3 class data is relabeled as 2 class i.e, Abusive and Hate-inducing classes are labeled as hate and non-offensive as non-hate.
- The third set of data is collected from a shared task HASOC.

4. Proposed Methodology

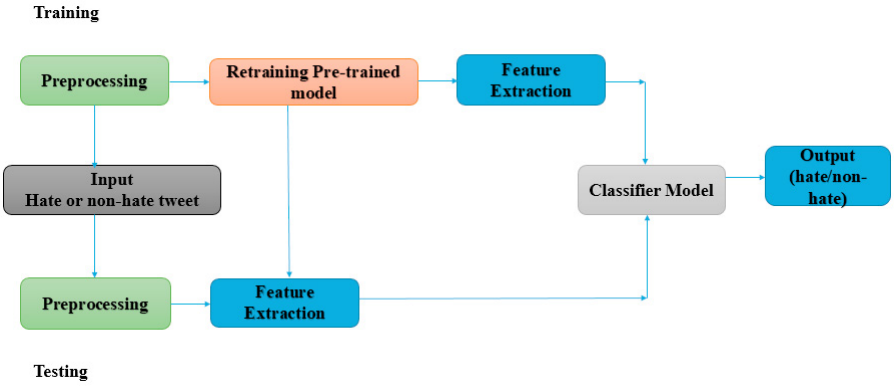

Fig. 1. Proposed Methodology

This section explains in detail the proposed methodology used for the detection of hate and non-hate tweets. Fig. 1 shows the flow chart of the proposed methodology. The detailed explanation of the proposed methodology is as follows:

- **Pre-processing:** As our dataset consists of tweets, there will be lot of emoticons, punctuation, special characters, URLs, hashtags and usernames. So the first step is pre-processing which involves removing
  - URL's
  - Usernames, Hashtags
  - emoticons
  - punctuation marks
  - unwanted characters
  - erasing extra white spaces from each sentence
  - converting all words to lower case

  This data pre-processing is done using regular expressions in python.
- **Retraining the model:**The cleaned data is then used for retraining the pre-trained model. In our work, we made use of two pre-trained model namely fastText and domain specific word embedding. In the case of fastText, each sentence along with the label is passed to the pre-trained model. Sentences are modified to ' __label__<L> <Text>' format where L is the class label and Text is the tweet to be classified.
  Eg: __label__<Hate> <Look ye politicians suvar jaise baithe rahte hain sirf money ke liye kaam karte hain. They don't care about public>

For retraining the domain specific word embedding, each sentence is tokenised and passed to Gensim's word2vec command which has an option to train.

- **Feature Extraction:**The most complex part of a NLP task is grabbing the right feature. Any deep learning or machine learning algorithm can only take number as input, so it is very important to convert the textual data to vectors. For this, the vector representation of each word is obtained from the retrained model in the case of fastText and domain specific word embedding. Where as for doc2vec the vector representation for a whole sentence is obtained. This forms the feature vector for the task.
- **Classification:**The obtained feature vectors were passed to the classifier models which predict whether a sentence is hate inducing or not. For our experiments we considered the classifiers such as linear SVM, SVM-RBF [16]and Random Forest [17]. Finally the performance of the classifier was evaluated by finding the parameters such as accuracy, precision, recall, F1-Score.

5. Experiments and Results

This section explains in detail the experiments conducted to evaluate the proposed methodology. In our paper we carried out 3 sets of experiments.

5.1. Experiment 1

Understanding the whole semantics of the sentence makes the task of classification easier. Hence, we decided to use doc2vec (implemented in Python library called Gensim) feature for classification. It has two types of training methods CBOW and skip-gram. Doc2vec is a method that converts sentences or paragraphs to vectors regardless of the sentence length. As it converts the whole sentence to a single vector it has the capability to capture the sentence semantics and it will also hold the word ordering.

For our experiment, we fixed the hyper-parameters of doc2vec with a vector length of 300 and the window size as 5 and minimum count as 1 and training was done using CBOW. Doc2vec of each sentence for the whole dataset was taken which formed a feature matrix of size 10000×300. This feature matrix is then fed to machine learning algorithms such as SVM and Random Forest [16], [17] for the classification. The performance of the classification models were evaluated using 10-fold crossvalidation and the evaluation parameters for this experiment is given in Table 2.

Table 2. Performance metrix for experiment 1

Machine learning algorithm	Accuracy (%)	Precision	Recall	F1-Score
SVM -linear	0.6375	0.6375	0.6375	0.6374
SVM-RBF	0.613	0.6150	0.613	0.6112
Random Forest	**0.6415**	**0.6415**	**0.6415**	**0.6414**

5.2. Experiment 2

It is evident from the results of experiment 1 that the doc2vec feature extracted from the dataset was not adequate for detecting the sentences with hate contents. As the embeddings generated by doc2vec are unable to give promising results, we took fragments of a sentence as our features to generate word level embeddings. Motivated by the paper [7], we used domain specific embedding for the feature extraction. This features were learned using the word2vec algorithm which is implemented in Gensim Python package. The whole dataset was tokenized and fed to word2vec model which takes distinct words and forms vector representation of specified length for it.

For our experiment we fixed the hyper parameters such as the vector length to be 300 and the window length as 5. Vector representation of each sentence was formed by concatenating the vector representation for each word. Since each sentence length is different, we took the mean of the words vectors of each sentence. This formed a feature

matrix of size 10000 × 300. Classification was carried out using machine learning algorithms such as SVM and Random Forest [16], [17]. Table 3 provides the details of the experimental results.

Table 3. Performance metrix for experiment 2

Machine learning algorithm	Accuracy (%)	Precision	Recall	F1-Score
SVM -linear	0.7281	0.7288	0.7281	0.7278
SVM-RBF	**0.7511**	**0.7517**	**0.7511**	**0.7509**
Random Forest	0.7267	0.7267	0.7267	0.7266

From the results obtained it is perceptible that word2vec has outperformed the doc2vec features in the detection of sentences with hate contents. And the performance is also comparable to the existing works done in this area. Since the detection accuracy is not satisfactory, we decided to compute the features in terms of the finer segments of a sentence which is the feature extraction at the character level.

5.3. Experiment 3

The previously experimented doc2vec and word2vec algorithms have a disadvantage that they can't handle out of vocabulary words. So most of the researches pertained to NLP make use of pre-trained models like fastText which were learned by training over a very huge corpus. Among these pre-trained models, fastText considers character n-grams as the smallest unit. FastText is a free open-source library developed by Facebook Research Team for potent sentence classification and for efficient learning of word representation. Unlike word2vec, fastText considers each word as a chunk of characters. For example the word vector 'apple' is a sum of the n-grams phrases such as '<ap', 'app', 'appl', 'apple', 'apple>', 'ppl', 'pple', 'pple>', 'ple', 'ple>', 'le>'. This makes the fastText capable of handling out of vocabulary words.

Since our data is Hindi-English, for our experiments we considered english pre-trained fastText model which we retrained using supervised learning option. FastText provides an option for incremental learning. Each preprocessed data was converted to ' __label__<L> <Text>' format where L is the class label and text is the tweet to be classified and we retrained the fastText model. After retraining we obtain a .bin and a .vec file. The .vec files contain only the aggregated word vectors. The .bin files in addition contains the model parameters, and crucially, the vectors for all the n-grams. Each text is tokenised and the vector representation for each word in a sentence is taken and concatenated. Each vector is of length 300. Since each sentence is of different length, the mean of the word vector representations were taken. This formed a feature matrix of size 10000 × 300. The detailed results for the experiment is shown in Table 4.

Table 4. Performance metrix for experiment 3

Machine learning algorithm	Accuracy (%)	Precision	Recall	F1-Score
SVM -linear	0.8144	0.8147	0.8144	0.8143
SVM-RBF	**0.8581**	**0.8586**	**0.8581**	**0.8580**
Random Forest	0.7834	0.7848	0.7834	0.7831

We used machine learning algorithms such as SVM and Random Forest for classification. Experiments were conducted using other classifiers also and SVM and RF obtained the best results. The proposed classification approach was compared with the recent paper by Aditya et.al [8]. They used Random Forest and SVM for the classification of hate speech with handmade features. In order to analyse the efficacy of our proposed approach, we implemented the same set of classification algorithms with new features for the detection of hate speech sentences.

From the three tables, it is noticeable that fastText features gave the highest accuracy of 85.81% using SVM- RBF as classifier followed by word2vec with 75.11% accuracy using SVM-RBF and then doc2vec with an 64.15% accuracy using Random Forest. The subword level information used by fastText in generating the feature representation for

words helped to achieve features better than word2vec and doc2vec for this task. The hyper-parameters used for the classifier with fastText features are shown in Table 5.

Table 5. Classifier parameters

Parameters	Kernel	C	Degree
Values	RBF	1000	2

Table 6. Details of the existing works

Author	Classifier	Accuracy(%)
Aditya et.al	SVM_RBF	71.7
Aditya et.al	RF	66.7
Santosh et.al	Sub-word level LSTM model	69.8
Santosh et.a	Hierarchical LSTM model with attention based on phonemic sub-words	66.6
Satyajit et.al	CNN-1D	82.62
Satyajit et.al	BiLSTM	81.48
Satyajit et.al	LSTM	80.21
Proposed methodology	SVM-RBF	85.81

Table 6 provided the details of the existing works done in the area of hate-speech detection from code-mixed Hindi-English Tweets. Further details about the features and dataset are provided in the related works. All these works were done on 4500 tweets introduced in the paper "A dataset of hindi-english code-mixed social media text for hate speech detection" by Aditya et.al [8]. But this 4500 sentences is inadequate for experimentation and also it provides insufficient statistical power. So, we collected and cleaned Hindi-English code-mixed hate speech datasets from different sources and conducted the experiments. From the above Table 6 it is evident that the proposed methodology out performed all the existing works in terms of performance by using pretrained embedding such as fasttext as feature.

6. Conclusion

Day by day, along with the increase in the usage of social media, the amount of hate content in them is also increasing drastically, resulting in lot of cyber violence and offenses. It is, therefore, highly imperative to detect the hate content in social media sites like Twitter and Facebook, before it reaches the mass. But this detection becomes a challenging task, as people tend to use code-mixed language. As majority of the Indian population speak Hindi, the need for Hindi-English code-mixed hate speech detection has risen. In this paper, a methodology using Facebook's pre-trained library fastText, is proposed to classify the tweets as hate and non-hate. The performance of the proposed methodology in hate speech detection is also compared with word2vec and doc2vec algorithms. Observing the performance evaluation parameters, it is evident that the proposed methodology is able to identify majority of the hate tweets. The experimental results confirm that the proposed methodology has outperformed all the existing works done in this area. We can also make an important conclusion from the results that character level features provide more information than word and document level for code-mixed text classification. Further, in future, the proposed method will be extended to the finer classification of hate tweets to offensive, abusive and profane.

References

[1] Al-Hassan, Areej, and Hmood Al-Dossari. (2019) "Detection of hate speech in social networks: asurvey on multilingual corpus." *Computer Science Information Technology (CS IT)* **9** (2) :8.

[2] Blaya, Catherine. (2019)"Cyberhate: A review and content analysis of intervention strategies." *Aggression and violent behavior* **45**: 163-172.

[3] Biere, Shanita, Sandjai Bhulai, and Master Business Analytics. (2018) "Hate Speech Detection Using Natural Language Processing Techniques". Diss. Master's dissertation, Vrije Universiteit Amsterdam.
[4] Ranjan, Prakash, Bharathi Raja, Ruba Priyadharshini, and Rakesh Chandra Balabantaray. (2016) "A comparative study on code-mixed data of Indian social media vs formal text." *2nd International Conference on Contemporary Computing and Informatics (IC3I) IEEE*: 608-611
[5] Bali, Kalika, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. (2014) ""I am borrowing ya mixing?" An Analysis of English-Hindi Code Mixing in Facebook." *Proceedings of the First Workshop on Computational Approaches to Code Switching* : 116-126
[6] Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov.(2017)"Enriching word vectors with subword information." *Transactions of the Association for Computational Linguistics 5* : 135-146.
[7] Satyajit Kamble , and Aditya Joshi. (2018) "Hate Speech Detection from Code-mixed Hindi-English Tweets Using Deep Learning Models." *arXiv preprint arXiv:1811.05145*
[8] Aditya Bohra, Deepanshu Vijay, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. (2018) "A dataset of hindi-english code-mixed social media text for hate speech detection." *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media* :36-41
[9] Santosh, T. Y. S. S., and K. V. S. Aravind. (2019) "Hate Speech Detection in Hindi-English Code-Mixed Social Media Text." *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data ACM* : 310-313.
[10] Mathur, Puneet, Ramit Sawhney, Meghna Ayyar, and Rajiv Shah. (2018) "Did you offend me? classification of offensive tweets in hinglish language." *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)* :138-148
[11] Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. (2016) "Bag of tricks for efficient text classification." *arXiv preprint arXiv :1607.01759*
[12] Santos, Igor, Nadia Nedjah, and Luiza de Macedo Mourelle. (2017) "Sentiment analysis using convolutional neural network with fastText embeddings." *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)* : 1-5
[13] Santos, Igor, Nadia Nedjah, and Luiza de Macedo Mourelle. (2017) "Sentiment analysis using convolutional neural network with fastText embeddings." *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)* : 1-5
[14] Tang, Duyu, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin.(2014) "Learning sentiment-specific word embedding for Twitter sentiment classification." *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*:1555-1565
[15] Premjith, B., K. P. Soman, and M. Anand Kumar. (2018) "A deep learning approach for Malayalam morphological analysis at character level." *Procedia computer science 132*: 47-54.
[16] Soman, K. P., R. Loganathan, and V. Ajay. (2019) "Machine learning with SVM and other kernel methods". *PHI Learning Pvt. Ltd*
[17] Premjith, Bhavukam, Kutti Padannayil Soman, and Prabaharan Poornachandran. (2019) "Amrita CEN@ FACT: Factuality Identification in Spanish Text." *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2019). CEUR Workshop Proceedings, CEUR-WS, Bilbao, Spain*
[18] Remmiya Devi, G., P. V. Veena, M. Anand Kumar, and K. P. Soman. (2016) "AMRITA-CEN@ FIRE 2016: Code-mix entity extraction for Hindi-English and Tamil-English tweets." *CEUR Workshop Proceedings* :304-308.
[19] Radim Rehurek, Petr Sojka. (2010) "Software Framework for Topic Modelling with Large Corpora." *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* :45-50.